Discovering Knowledge Abstractions For Sample Efficient Embodied Transfer Learning

Sasha Salter
Balliol College
University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2022

Acknowledgements

Supervisor I would like to greatly thank my supervisor Prof. Ingmar Posner for his continued support and guidance. His advice has been invaluable and paramount towards this thesis.

Collaborators This thesis would not exist without all the collaborators. First and foremost, I am immensely grateful for my collaborators at DeepMind for their continued advice, expertise and efforts during the collaboration and internship projects. Specifically, I thank Raia Hadsell and Martin Riedmiller for their high level and insightful guidance. I thank Nicolas Heess and Dhruva Tirumala for all the eye opening conversations had. Finally and most crucially, I am extremely appreciative of the continued heavy involvement and interest of Markus Wulfmeier and Dushyant Rao throughout my doctorate. This thesis has been heavily influenced by both. Beyond DeepMind, I am very grateful of Kristian Hartikainen's contribution towards the codebases used in this thesis. I additionally thank all additional collaborators listed below:

- APRiL: Dushyant Rao, Markus Wulfmeier, Raia Hadsell, Ingmar Posner
- TACO: Kyriacos Shiarlis, Markus Wulfmeier, Shimon Whiteson, Ingmar Posner
- APES: Kristian Hartikainen, Walter Goodwin, Ingmar Posner
- MO2: Dushyant Rao, Markus Wulfmeier, Dhruva Tirumala, Nicolas Heess, Martin Riedmiller, Raia Hadsell

DPhil Funding I am grateful for the funding of my DPhil via an EPSRC Studentship.

Computing Resources I would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out these projects (http://dx.doi.org/10.5281/zenodo.22558) and the use of Hartree Centre Resources via JADE HPC UK.

Reviewers I would like to thank the reviewers who provided constructive feedback on the publications that resulted from the work conducted for this thesis.

Examiners I would like to express my gratitude to Dr Ioannis Havoutis from the University of Oxford and Dr Yuke Zhu from the University of Texas at Austin for taking the time to examine this

thesis and for providing insightful feedback.

Family and Friends Finally, I would like to thank my family and friends for their support. I thank Theo Istrate for his continued friendship and positivity throughout this PhD. I thank Rasheed El-Bouri for always believing and supporting me throughout the doctorate. I thank my brothers, Nathan Salter Perez and Maximos Atkinson Perez for being there whenever I needed them. I thank my parents, Marcos Salter and Maria Atkinson Perez, and their partners, Olga Salter and Matthew Atkinson Perez, for their kindness, support, and love. Finally, I am ever grateful for my wife, Yi Song, for supporting me, emotionally and financially, throughout and for being patient and always there to lean on.

Abstract

This thesis concerns sample-efficient embodied machine learning. Machine learning success in sequential decision problems has been limited to domains with a narrow range of goals, requiring orders more experience than humans. Additionally, they lack the ability to generalise to new related goals. In contrast, humans are continual learners. Given their embodiment and computational constraints, humans are forced to reuse knowledge (compressed abstractions of repeated structures present across their lifetime), to tackle novel scenarios in as sample-efficient and safe manner as possible. In robotics, similar traits are desired, given they are also embodied learners. Taking inspiration from humans, the central claim of this thesis is that **knowledge abstractions acquired from prior experience can be used to design domain-independent sample-efficient algorithms that improve generalisation across modular domains**. We refer to modular domains as Markov decision processes (MDPs) whose optimal policies can be obtained when reasoning and acting occurs over compressed abstractions shared across them. The challenge is how to discover these abstractions with minimal supervision and sample-efficiently. Additionally, for embodied machine learning it is important the approach supports continuous, potentially unbounded, state-action spaces. Adhering to these constraints, we first develop novel self- (Chapter 3) and weakly-supervised (Chapter 4) knowledge abstraction, domain adaptation, methods for zero-shot generalisation to unseen domains. We demonstrate their potential on robotic applications including sim2real transfer (Chapter 3) and generalisation using a human-robot command interface (Chapter 4). We continue by developing novel unsupervised knowledge abstraction, transfer learning, methods for sample-efficient adaptation to unseen domains (Chapters 5 and 6). We highlight their relevance in robotics and continual learning. We introduce a hierarchical KL-regularised RL approach based on novel theory behind the transferability-expressivity trade-off of abstractions (Chapter 5) and develop the first, to our knowledge, bottleneck-options approach adhering to the aforementioned embodied machine learning constraints (Chapter 6).

Contents

Chapter 1

Introduction

Recent deep reinforcement learning (RL) advancements have achieved impressive feats across a range of complex sequential decision making domains[95;143;144;169] formulated as Markov decision processes (MDPs)[13]. Nevertheless, this success has focused on training agents across a narrow range of goals and lack the ability to generalise to new related ones, even for simple RL domains[14]. Additionally, success has required orders more experience than humans. In contrast, humans are *continual learners*, continuously learning across a lifetime. Humans refine and re-apply knowledge they acquire across their life to safely and sample-efficiently adapt to unseen and diverse scenarios[127]. In its most ambitious formulation, *continual learning* tackles learning across a continuously changing, non-stationary, environment. In the RL context, these non-stationarities can occur at any point in time, across any environment component, such as non-stationary rewards if goals change, or non-stationary dynamics due to, for example in robotics, equipment wear and tear.

Embodied learners acting in the real world encounter similar constraints as humans, primarily the need for safe, sample- and computation-efficient learning and acting, in the presence of non-stationarity[29]. Therefore, the studies of *continual* and *human learning* are of importance to the robotics community as well. The most ambitious formulation of *continual learning* is an extremely challenging problem, with multiple hurdles that need overcoming, such as knowledge discovery and adaptation, catastrophic forgetting, context detection, and forwards and backwards transfer interference[68]. Tackling all these problems at once is a grandiose challenge. As such, multiple *continual learning* sub-fields have emerged studying individual challenges. This thesis tackles the sub-fields of *domain adapta-*

tion[68] and *transfer learning*[68] and how *knowledge discovery and adaptation* can aid them.

In both *domain adaptation* and *transfer learning*, the agent must learn over source domains such that it generalises to a target domain. Neither assume access to the target domain when training over the source domains. Therefore, both focus on *generalisation* to unseen domains, a key property for *continual* and *embodied learners*, for adaptability and robustness. For *domain adaptation*, the learner must generalise in a zero-shot fashion, without additional training on the target environment. In *transfer learning*, the agent is permitted additional training on, and interaction with, the target domain, but is expected to sample-efficiently (and safely in some cases) adapt. *Domain adaptation* is a field of particular importance to robotics as it concerns safety, being able to handle unseen scenarios as they present themselves (such as the first time a pedestrian walks in-front of a self-driving vehicle). *Transfer learning* is also relevant to robotics in scenarios where zero-shot generalisation is not possible but safety and sample-efficiency are still paramount. Additionally, *transfer learning* is closely related to *continual learning*, both continually learning in the presence of non-stationarity.

In this thesis, we take inspiration from humans as continual learners, and build on the intuition that the discovery of knowledge across sequential, related, domains can aid generalisation across them. We consider *domain adaptation* and *transfer learning* problems where the source and target domains share modular structure, defined in the next section, which if discovered can facilitate effective generalisation. We consider the findings in this thesis to contribute to both the *robotics* and *continual learning* communities.

1.1 Thesis Statement

The central claim of this work is that **state, action, and temporal abstractions can be used to design domain-independent sample-efficient algorithms that improve compositional generalisation to modular tasks and observations.**

State, action, and temporal abstractions We refer to *abstractions* as the compression of reoccurring structures deemed beneficial across a lifetime of learning, in line with Thrun and O'Sullivan[160]. *State-abstractions* compress the environment state-space into an alternate representation that is ideally beneficial for reasoning and generalisation, $S \rightarrow \tilde{S}$, (or observation-space, $O \rightarrow \tilde{O}$, in the partially

observable MDP setting[65]). *Action-abstractions* compress the agent's action-space, typically constraining available actions to those beneficial for the MDPs of interest, $A \rightarrow \tilde{A}$. S, O, A represent the original environment state, observation and action spaces while $\tilde{S}, \tilde{O}, \tilde{A}$ are their abstracted equivalents. *Temporal-abstractions*, typically framed as semi-MDPs[120] (SMPDs) (explicitly introduced in Section 2.3.2), support temporally abstract reasoning over decision making states, ideally constraining reasoning only over relevant decision states. As Li et al.[83] highlight, useful abstractions preserve crucial information for decision making over the abstracted MDPs of interest. Abstracted MDPs refer to the (S)MDPs, $\tilde{M}$, that result by abstracting the original MDPs', M, state, action, and/or decision making state spaces. For a formal definitition of MDPs, M, see Section 2.1.

Domain-independent sample-efficient algorithms We refer to *domain-independent algorithms* as those able to succeed on a variety of domains without requiring domain-specific tailoring, using the same definition as Cholodovskis Machado[20]; Naddaf[101]. *Sample-efficient* refers to the ability of the algorithm to succeed using as few environment samples as possible.

Compositional generalisation We refer to *generalisation* as the ability to generalise performance (obtain high return - see Section 2.1) from source MDPs, M_{source}, to unseen target MDPs, M_{targ}, in either a zero-shot manner, for *domain adaptation* approaches, or with additional training on the *target* domain, for *transfer learning* methods. The *target* MDPs may vary in reward, transition and observation functions, as well as state and action spaces, as discussed in Section 2.2. *Compositional generalisation* refers to *generalisation* to M_{targ} when reasoning (learning and acting) instead over abstracted target MDPs, $\tilde{M}_{targ}$, using state, action, and/or temporal abstractions discovered over M_{source}.

Modular tasks and observations Given a family of MDPs, they are said to differ in *tasks* if they have distinct reward functions. The same can be said for *observations* if the observation functions differ (see Section 2.1). Source and target MDPs (M_{source} and M_{targ} respectively) are said to be modular in structure if there exists shared abstractions across them that support optimal policies, π^* (achieving maximum theoretical return), when learning instead over the abstracted MDP equivalents, $\tilde{M}_{source}$ and $\tilde{M}_{targ}$. The question remains how to discover beneficial abstractions for *compositional generalisation* across *modular MDPs* using minimal supervision.

The above algorithmic traits are commonly considered core components for successful performance

in the fields of *domain adaptation*, *transfer learning*, and most ambitious of all, *continual learning*[108;23;49;68]. For embodied learners acting in a non-stationary environment, such as many robotic applications[29], data collection is costly necessitating *sample-efficiency*, and zero-shot *generalisation* is essential for robustness to such non-stationarities. We discuss forms of non-stationarities in Section 2.2. As such, the methods proposed in this thesis are practical with robotic applications and additionally contribute to the longer term vision of fully autonomous *continual learning*.

1.2 Approach

We support the thesis statement with different algorithms, theoretical justifications, and experimental results. This section provides a high-level overview and contextualisation of our approaches. Subsequent sections will delve deeper on both these fronts. To support sample-efficiency and flexibility of our frameworks, we build on *off-policy* methods for discovering and leveraging abstractions. These enable learning over agent-agnostic samples, in contrast to their on-policy counterparts that require samples from the current policy. As such, we maximise data reuse, learning over samples from a possibly outdated policy or expert. We employ *model-free actor-critic* RL as it supports continuous state-action spaces, common in many robotic applications, and has achieved significant success in the field[3]. Finally, we propose *supervision-efficient* methods for abstraction discovery to support practicality and ultimately autonomous continual learning. To this end, we explore weakly-, self-, and un-supervised approaches.

We continue by detailing the four algorithms introduced in this thesis. We first detail two weakly- and self-supervised *domain adaptation* algorithms for compositional generalisation. These algorithms are tailored for robotics, the first tackling *sim2real* modular observation generalisation; the latter concerning modular task generalisation using a *human-robot* command interface. Afterwards, we present two unsupervised *transfer learning* methods for compositional generalisation across modular tasks, and are thus closely aligned with autonomous *continual learning*.

In Chapter 3, we present a *state-abstraction* approach for *sim2real domain adaptation*. *Sim2real adaptation* is of interest in robotics as it enables safe training and deployment of embodied reinforcement learners. By training entirely in simulation, we avoid employing the unsafe trial-and-error RL paradigm in the real world. Our approach discovers *state-abstractions* achieving compositional

generalisation across modular image observations. Specifically, we focus on generalisation across distractors; object-level components of the observation space that an optimal policy should not depend on and should be invariant to. We leverage privileged information, available during simulation, but not deployment, to guide learning of object-level, observation abstractions that are invariant to and hence generalise across distractors. This is achieved by a novel self-supervised attention-based mechanism suppressing distractors. Compared with alternate *sim2real* approaches, our abstractions improve generalisation to domains with unseen distractors, key for robust *sim2real* deployment.

In Chapter 4, we introduce a *domain adaptation* method, leveraging *temporal-abstractions*, for compositional generalisation across modular tasks. Our approach uses weak-supervision to learn grounded, interpretable, *temporal-abstractions* corresponding to sub-tasks. Specifically, during training the user provides the learner with a list of labels indicating which ordering of sub-tasks occurred. During transfer, a new list can be provided and the agent will execute the novel sub-task ordering. As such, our approach provides a *human-robot* communication interface for generalisation across sub-task orderings. We introduce a novel connectionist temporal classification (CTC)[39] and options[156] inspired approach that simultaneously aligns labels with the location of sub-tasks within the demonstration and learns the corresponding controllers. By coupling these processes, we improve segmentation, controller, and generalisation performance. Additionally, by optimising over a distribution of alignments, we improve sample-efficiency, key in robotics. Finally, we perform commensurately with fully supervised approaches, provided with sub-task alignment, whilst requiring significantly less annotation effort, which can be costly.

In Chapter 5, we introduce our first *transfer learning* approach, that leverages unsupervised *action-abstractions*, for compositional generalisation across modular tasks. We build off powerful, probabilistic, variational methods given their empirical success[86;93;94;159;162;163]. These methods employ information asymmetry[36] (IA), between architectural modules, to influence which abstractions are learnt. Nevertheless, the literature rarely ablates over IA, usually chosen on intuition, despite its heavy influence. We introduce theory behind the transferability-expressivity trade-off for any given abstraction, and the crucial role IA plays. Given these insights, we combine *hierarchical* and *KL-regularised*[159] RL for their individual benefits, imposing hard and soft compositional constraints on transferred behaviours. We employ *hierarchy* to transfer multi-modal behaviours that generalise

favourably at the expense of expressivity, and *priors* for directed behaviours that do not generalise. Combined, we benefit from both directed and generalisable behaviours. On the transfer domain, we outperform unsupervised methods that lack *hierarchy*, *priors*, or correct *IA* choice. The theory and expressive framework we introduce provide one small step towards *continual learning*.

In Chapter 6, we present a *transfer learning* method, focused on unsupervised *temporal-abstractions*, for compositional generalisation across modular tasks. We build on the options framework[156] for discovering these abstractions. In-line with the traditional options literature[91;145;150;53;124], we seek options that terminate at *bottleneck states* (states most frequently visited when considering the shortest distance between any two states in an MDP[150]). *Bottlenecks* have been shown beneficial for planning over. Nevertheless, traditional methods do not support *embodied continual learning*. They are either not off-policy[124] necessary for sample-efficiency, or are restricted to discrete state- and/or action-spaces[91;91;145;150;53]. We introduce a framework with these properties. We build on Hindsight Off-Policy Options (HO2)[177], a highly performant sample-efficient off-policy framework that, given any trajectory, back-propagates gradients through the dynamic programming inference procedure to robustly train all options end-to-end. Our *bottleneck options* not only improve exploration, but also credit-assignment and value estimation[155]. Altogether, our options improve policy performance on the transfer domain when compared to their non-bottleneck counterparts. In short, our framework enables *bottleneck option* discovery in a *continual learning* and *robotics* friendly manner.

1.3 Contributions

Work during this DPhil has led to a number of papers. We categorise these into published papers, papers under submission, and additional papers.

1.3.1 Published Papers

These papers have been published at peer-reviewed conferences and contribute to this thesis.

- Chapter 3, S. Salter et al. "Attention-Privileged Reinforcement Learning". In: *Robotics: Science and Systems Workshop on Structured Approaches to Robot Learning for Improved Generalisation (RSS SARL Workshop) (2020)*[132].

- Chapter 3, S. Salter et al. "Attention-Privileged Reinforcement Learning". In: *Conference on Robot Learning (CoRL) (2020)*[133].
- Chapter 4, K. Shiarlis et al. "TACO: Learning Task Decomposition via Temporal Alignment for Control". In: *International Conference on Machine Learning (ICML) (2018)*[140].

1.3.2 Papers Under Submission

These papers are currently, or soon to be, under submission at peer-reviewed conferences and contribute to this thesis.

- Chapter 5, S. Salter et al. "Priors, Hierarchy, and Information Asymmetry for Skill Transfer in Reinforcement Learning". In: *Conference on Robot Learning (CoRL) (2022)*[134].
- Chapter 6, S. Salter et al. "MO2: Model-Based Offline Options". In: *Conference on Lifelong Learning Agents (CoLLAs) (2022)*.

1.3.3 Additional Papers

These papers do not contribute to this thesis and have already been published at a peer-reviewed conference.

- Y. Wu et al. "Imagine That! Leveraging Emergent Affordances for 3D Tool Synthesis". In: *International Conference on Machine Learning Workshop on Object-Oriented Learning (ICML) (2020)*[173].

1.4 Dissertation Layout

This document contains eight chapters. Chapter 2 provides a high level overview of related background and prior research contextualising this thesis with relation to *continual learning*. Chapters 3 to 6 cover each novel abstraction approach for compositional generalisation. Each chapter delves deeper into related works, problem formulation, and background theory. Claims made throughout are supported by empirical evidence. Chapters 3 and 4 consider *domain adaptation* methods, while Chapters 5 and 6 concern *transfer learning*. We conclude with Chapter 7, summarising our findings and contributions, and discuss limitations and future research directions. Finally, Chapter 8 includes derivations, additional analysis, and details required to reproduce every experiment.

Chapter 2

Background

This chapter aims to contextualise the contributions presented in Chapters 3 to 6 with respect to continual (reinforcement) learning. Continual learning is a very ambitious problem with equally broad definitions. There exist several literature reviews aiming to provide a concrete problem definition, highlight hurdles that have to be overcome, and categorise related fields and methods[108;23;49]. This chapter closely adheres to Khetarpal et al.[68] as it specifically covers continual reinforcement learning, rather than its supervised learning counterpart. We start by defining the typical reinforcement learning paradigm. We continue by relating it to continual reinforcement learning and related fields. Finally, we present a taxonomy of continual learning approaches, specifying which assumptions they make, which continual learning problems they tackle, and how this thesis contributes to them.

2.1 Reinforcement Learning

Notation: In this chapter, capital letters refer to random variables and lower case is used for values for those variables as well as scalar functions. In later chapters we break this convention to match closely related works, but make it clear when we do so.

Sequential decision making (SDM) problems consider the setup where an agent must make decisions over multiple time-steps to achieve some goal. Reinforcement learning is a specific formulation of SDM where at every time-step an agent is in a state, takes an action, and then transitions to a new state observing a real-valued reward signal. The agent's goal is to maximize a (possibly weighted)

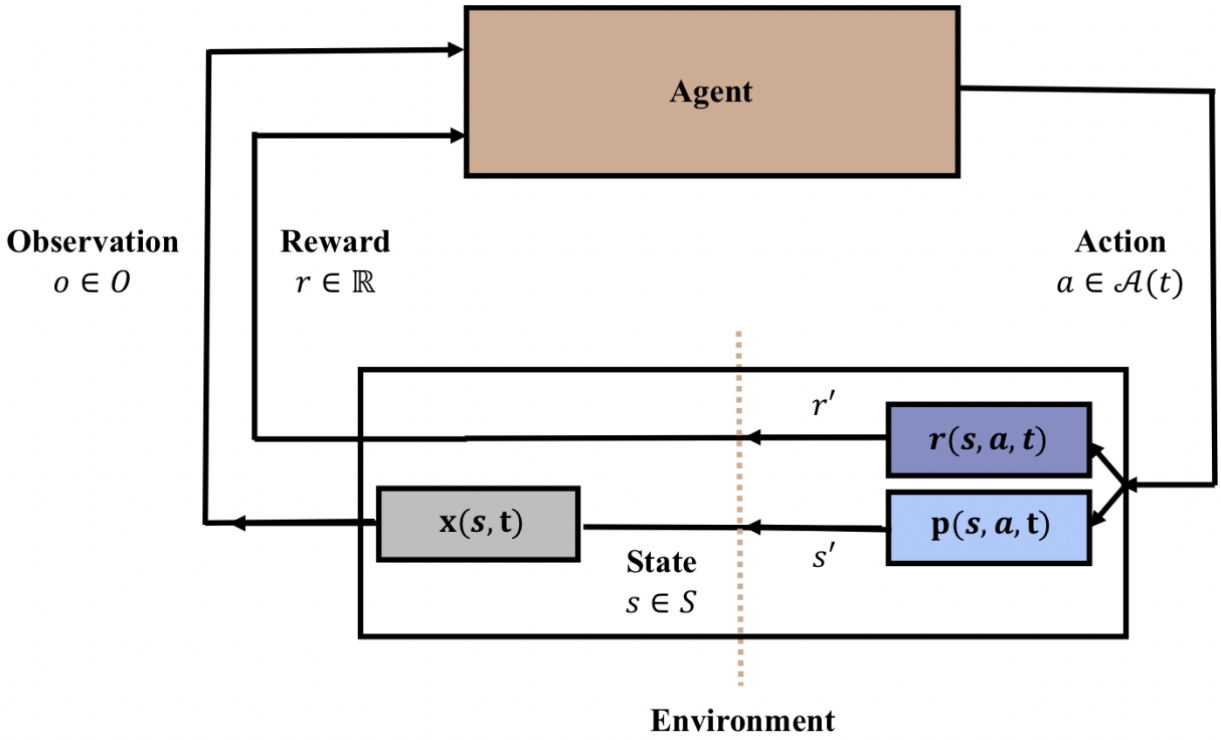


Figure 2.1: **Agent-Environment Interaction** with Potentially Time Dependent Environment Components. Highlighting the agent-environment interaction in reinforcement learning (from Khetarpal et al.[68]).

sum of future rewards. The actions taken by the agent influence the immediate reward it observes as well as future rewards. Consequently, this problem implicitly requires agents to deal with the trade-off between immediate and future rewards. In the reinforcement learning framework we formulate sequential decision making problems as tasks where an agent must learn how to act optimally through trial-and-error interactions with a complex, unknown, stochastic environment.

The most common RL formulation assumes the environment satisfies the Markov property and can be modelled as a discrete-time Markov decision process (MDP)[120;155]. An MDP is defined by $M = (S, A, r, p, p^0, \gamma)$. S and A denote state and action spaces respectively, $r : S \times A \to \mathbb{R}$ is the reward function, $p : S \times A \to Dist(S)$ is the environment dynamics model, $p^0 : Dist(S)$ is the environment initiation state probability distribution, and $\gamma \in [0, 1)$ is the discount factor. Starting at state $s \in p^0(S)$, for each timestep t, the agent takes action $a \in A$ and transitions to state $s' \in S$ according to environment one-step dynamics model $p(s'|s,a) \doteq P^a_{ss'} = Pr(S_{t+1} = s'|S_t = s, A_t = a)$, receiving a one-step expected reward $r(s,a) \doteq R^a_s = \mathbb{E}[R_{t+1}|S_t = s, A_t = a]$ from the environment. Collectively, $P^a_{ss'}$ and R^a_s represent a one-step model of the environment. In the partially observable POMDP setting[65], the agent does not have access to the Markovian environment state, and instead recieves an observation from $x(o|s)$:

$S \to Dist(O)$, with O representing the observation space.

The agent's goal is to learn a policy $\pi(a|s) : S \to Dist(A)$ that maps each state to a probability distribution over actions. The optimal policy maximizes, on expectation, the discounted cumulative sum of rewards, also known as return, defined as:

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{k+t+1} \tag{2.1}$$

with $\gamma \in [0,1)$ being the discount factor defining the relative value of future rewards. This thesis focuses on value-based methods for optimising the policy. Value-based methods estimate the *state-value function* $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$ or the *action-value function* $q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$. The expectations are taken with respect to the expected future rewards induced by agent π, environment dynamics model p and reward model r. These functions can be defined recursively:

$$v_\pi(s) = \mathbb{E}_{\pi(a|s)}[r(s,a) + \gamma \mathbb{E}_{p(s'|s,a)}[v_\pi(s')]] \tag{2.2}$$

$$q_\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{\substack{p(s'|s,a) \\ \pi(a'|s')}}[q_\pi(s',a')] \tag{2.3}$$

RL algorithms can be broadly classified as *model-free* or *model-based*. *Model-based* RL relies on environment dynamics and reward models to compute value functions and policies. In many practical situations, such models are unavailable and learning them can be problematic, especially in high-dimensional state-action domains. We focus instead on *model-free* RL where the agent does not have access to the environment dynamics or reward models. *Model-free* approaches estimate v_π and q_π directly from samples $(s_t, a_t, r_{t+1}, s_{t+1})$ without explicitly modelling environment dynamics or rewards (here r_{t+1} refers to the observed reward at time $t+1$, not the reward function itself). We focus on off-policy methods, due to their ability to learn value functions from agent-agnostic samples, in contrast to on-policy approaches that require samples from current policy π. As such, off-policy approaches are significantly more sample efficient than their on-policy counterparts. Specifically, we build on off-policy actor-critic approaches to policy optimisation, as these support continuous state-

action space domains. These methods build on Q-learning for estimating value functions. Q-learning is based on temporal difference learning[155], updating the critic, $\tilde{q}$, that estimates the action-value function, q_π, according to the temporal difference error such that:

$$\tilde{q}(s_t, a_t) \leftarrow \tilde{q}(s_t, a_t) + \alpha(r_t + \gamma \mathbb{E}_{\pi(a|s_{t+1})}[\tilde{q}(s_{t+1}, a)] - \tilde{q}(s_t, a_t)) \tag{2.4}$$

with α denoting step-size for the critic update. The critic is used to update the policy. In the greedy setting, the policy is chosen as follows:

$$\pi(s) \doteq \arg\max_{a \in A} \tilde{q}(s, a) \tag{2.5}$$

2.2 A Spectrum of Learning Settings

In many situations of interests, the learning paradigm may be non-stationary. In the RL context, environment non-stationarity can capture a wide range of learning scenarios and applications. In the field of robotics, some practical non-stationary scenarios include equipment wear and tear over time or sequential tasks (such as navigating novel routes on the fly for self-driving cars). By conditioning each environment component on time, seen in Figure 2.1, we capture all potential forms of non-stationary and present the most ambitious formalisation of continual RL, where the agent must handle non-stationary environment transitions, rewards, observations- and action- spaces. Handling this many non-stationarities presents a real challenge. Therefore, the community primarily focus on the practical sub-problems presented in Figure 2.2 tackling specific continual learning challenges. In this thesis, we focus on the fields of *domain adaptation* and *transfer learning*, as these enable safe and sample-efficient embodied learning (discussed in more detail later), crucial components for robotics. The following narrative focuses on these fields and their relation to continual learning. For a more comprehensive overview please refer to Khetarpal et al.[68].

Setting	Multiple Domains Of Deployment	Multiple Required Skills	Universal Master Policy	Non-stationary Evolution
Domain Adaptation	✓	X	X	X
Transfer Learning	✓	✓	X	✓
Meta-Training and Meta-Testing	✓	✓	X	X
Multi-task Learning	✓	✓	✓	X
Continual (Lifelong) Learning	✓	✓	✓	✓

Figure 2.2: **A Spectrum of Learning Settings Leveraging Prior Knowledge:** For each setting we consider whether they typically involve multiple domains, multiple skills, a universal master policy to solve all tasks, and a non-stationary evolution of the task distribution. We are not specifying how each setting should be tackled as that is the property of the approach (from Khetarpal et al.[68]).

2.2.1 Domain Adaptation

Domain adaptation tackles the challenge of adapting a policy exhibiting a single skill[1] (a policy that solves a single task such as block stacking) to a new domain(s) (such as stacking previously unseen blocks). The typical learning paradigm assumes environment stationarity during training, but not deployment. These approaches are particularly pertinent in robotics, where training on the target domain may be costly in terms of safety, time and money. *Sim2real* transfer in robotics is a sub-field of *domain adaptation* where the agent is trained in simulation with the expectation to generalise to the real-world. Domain randomisation is an approach to *sim2real* transfer tackling generalisation between source and target domains. Typically, *sim2real* transfer assumes domain shifts with respect to environment dynamics and observation functions. Nevertheless, the more general setting tackles domain shifts with respect to rewards and actions as well. Chapters 3 and 4 present *domain adaptation* methods that leverage state- and temporal- abstractions (more details in Section 2.3) to improve sample-efficiency and generalisation across observations and rewards, respectively. The reader is referred to recent surveys such as Tobin et al.[164] for a more comprehensive overview.

2.2.2 Transfer Learning

Transfer learning studies how to leverage *source* domains for improved learning on *target* domains. As shown in Figure 2.2, this setting usually assumes multiple domains and skills (multiple reward functions or tasks), that have to be learnt and leveraged in the presence of non-stationarity. This non-

[1]However, this restriction can be relaxed, as is the case in Chapter 4. See Section 2.3.2 for our definition of skill.

stationarity during training is the key distinction between *transfer learning* and *domain adaptation*. Typical *transfer learning* approaches assume abrupt non-stationarities that are given to the learner during training (such as task id each time the task changes). Therefore, *transfer learning* does not deal with task-inference[22], an additional challenge for fully autonomous continual learners. *Transfer learning* approaches commonly break up learning into distinct phases, such as pre-training (on the *source* domains) and fine-tuning (on the *target* domain). For embodied learners, where sample-efficiency is often the limiting factor for learning, transfer of knowledge across related domains/tasks may be crucial. Chapters 5 and 6 present *modularity and skill focused* (see Sections 2.3.1 and 2.3.2) *transfer learning* methods that improve exploration (and value estimation in the case of the latter) across domains with distinct rewards. Both approaches assume that the source and target tasks exhibit modularity (discussed further in Section 2.3) that can be leveraged for sample-efficient transfer. For a detailed literature review on *transfer learning* RL methods please refer to Taylor and Stone[158].

2.2.3 Continual Learning

Fully autonomous continual learners are closely related to *transfer learning* and *domain adaptation* agents, all leveraging prior experience from previous *source* domains to quickly adapt to the current *target* domain. One key distinction is that in *continual learning* a universal policy is learnt to coordinate behaviours across the agent's lifetime in the presence of environment non-stationarity. Fully autonomous learners are not given the boundaries of non-stationarity (which may not be abrupt) exacerbating the learning problem. Furthermore, in the lifelong setting, there exist multiple boundaries making it harder to retrieve relevant information from an increasingly large and diverse corpus of experience. Given computation and memory constraints, continual learners typically rely on information compression for knowledge retention and transfer. A few information compression approaches (discussed in Section 2.3) include the discovery of information bottleneck states (Chapter 6), sub-tasks (Chapter 4), action-/temporal-abstractions (Chapters 5 and 6) and state-abstractions (Chapter 3). Crucial to each approach is the extraction of knowledge that generalises favourably to the *target* domain, enabling sample-efficient adaptation of behaviours. The extent to which previous experience can benefit adaptation heavily depends on the similarity between *source* and *target* domains. As such, *curriculum learning* is a closely related field that tackles the automation of environment non-stationarities as to maximise transfer benefits. Finally, *catastrophic forgetting* and *task interference*

are common complications for continual learners where training data distributional shifts coupled with knowledge sharing lead to inferior final performance on the *source* and *target* domains. For a more detailed discussion please refer to Khetarpal et al.[68].

2.2.4 Meta and Multi-Task Learning

Meta and *multi-task learning* are closely linked to *continual learning*. We only provide a brief overview here as this thesis does not contribute to these fields. For a more comprehensive discussion see Khetarpal et al.[68]. *Meta learning* relates closely to *domain adaptation* with the goal to generalise behaviours from *source* to *target* domains. Both assume environment stationary during training. The main distinction is that *meta learning* focuses on discovering a learning framework that generalises as opposed to a policy. *Multi-task learning* is closely related to *transfer learning* leveraging shared structure between tasks to benefit exploration and learning between them. In contrast to *transfer learning*, tasks are learnt jointly, not sequentially, and thus the environment is stationary. We consider this setup less suited for robotics where tasks are inherently sequential.

2.3 A Taxonomy of Continual Learning Approaches

We proceed by discussing common approaches for tackling *continual learning* and related fields. Khetarpal et al.[68] categorises each approach into one of three high-level clusters (see Figure 2.3): *explicit knowledge retention*, *leveraging shared structure*, *learning to learn*. *Explicit knowledge retention* methods address catastrophic forgetting (forgetting previous behaviours as the data distribution shifts) in continual learning, focusing on the stability-plasticity dilemma[126]. *Learning to learn* deals with meta-learning and focuses on adaptability, exploration and context detection. This thesis builds on *leveraging shared structure* approaches to improve generalisation and adaptability in *domain adaptation* and *transfer learning*. As such, the following narrative focuses on *leveraging shared structure* and we refer the reader to Khetarpal et al.[68] for details on the remaining categories.

By discovering repeated and reoccurring structures deemed beneficial across a lifetime[160] and representing them as compressed abstractions[161;109] (which we refer to as knowledge), embodied continual learners with budgetary constraints are able to recompose previously acquired knowledge, representing aspects of previous solutions, to efficiently solve the task at hand[41]. *Leveraging shared*

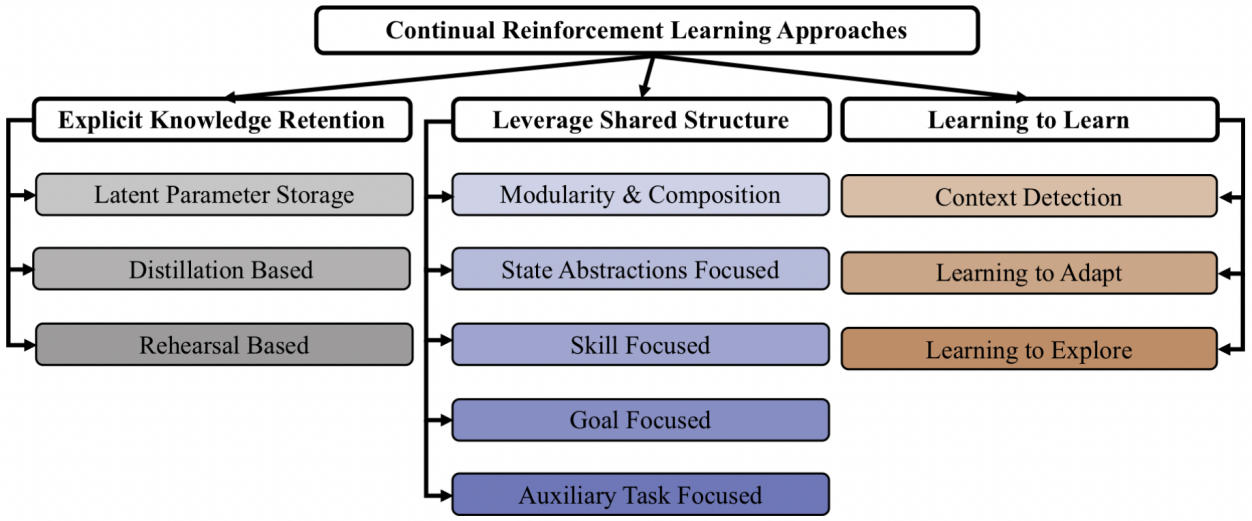


Figure 2.3: **Taxonomy of Continual RL Approaches:** Illustrating distinct continual RL clusters highlighting prominent threads of research within each family (from Khetarpal et al.[68]).

structure methods adhere to this philosophy, focusing on abstractions that to some degree generalise. The upcoming sections do not cover goal or auxiliary task focused methods as we do not build on them in this thesis. Refer to Khetarpal et al.[68] for details on their relation to the following sections.

2.3.1 Modularity and Composition

These approaches explicitly formulate tasks as compositionally structured with the goal of efficiently discovering this structure to achieve compositional generalisation and solve increasingly complex tasks. While task compositionality can occur at the state, action and temporal levels, we group the latter into Section 2.3.2 that concerns macro action discovery. A common thread for discovering task compositionality is to decompose monolithic neural networks into smaller modular components that specialise (usually at the task-level[24;178;32;128;17;8;176], but also embodiment-level[24] for embodied transfer). This decomposition can be provided by the user[24;17;8;176] or automated[178;32;128]. Furthermore, decomposition can recombination can occur as soft[178] and hard[24;32;8;176] boundaries.

Probabilistic hierarchical methods have recently found significant success in complex control domains[56;162;163;86;93;94] thanks to the powerful variational inference and policy-optimisation[3] frameworks they build on; both scaling favourably with number of tasks, experiences, and compositionality complexity. These methods support unsupervised, off-policy discovery of compositional behaviours and thus promote sample-efficient autonomous learning, crucial in robotics and *continual learning*. In this setting, the policy is augmented with latent actions able to capture high-level behaviours. For

a two-level hierarchy, the decomposition typically takes the form: $\pi(a,z|s) = \pi^L(a|z,s)\pi^H(z|s)$ (z representing the latent action). Apart from their empirical success, these methods provide high interpretability, as each latent directly corresponds to a low-level behavioural primitive, $\pi^L(a|z,s)$.

Nevertheless, task compositionality cannot always be strictly adhered to. When this is the case, *distillation methods* (see Figure 2.3) provide an alternative[159;162;163;86;36] imposing soft compositional constraints on shared behaviours across tasks at the expense of less directed transfer[134]. KL-regularised RL[159] proves a particularly competitive instantiation, also building on variational inference, and transfers knowledge across tasks through a learnt task-agnostic behavioural prior, $\pi_0(a|s)$, which the policy is regularised against: $\mathrm{D_{KL}}(\pi(a|s)||\pi_0(a|s))$. Nevertheless, discovering compositionality is an under-constrained problem, as there exist multiple ways to compose prior behaviour, varying in degrees of action, state and temporal abstractions. It is unclear which choice of abstraction should be learnt for maximal transfer benefits.

In Chapter 5, we provide theory behind the *transferability-expressivety* trade-off of discovered abstractions and introduce a *transfer learning* framework that combines hierarchical and KL-regularised RL for discovering these, leveraging their individual benefits. Our insights additionally contribute to the wider *continual learning* community. Refer to Rosenbaum et al.[128] for a more detailed survey on the additional challenges modular and compositional approached face.

2.3.2 Skill Focused (Temporal Abstractions)

We categorise *skill*[2] *focused* approaches as those that discover macro actions[55;161], also known as temporal abstractions, sidestepping the requirement to make decisions at every time-step within the MDP. These frameworks are appealing as they enable acting, reasoning (value estimation using TD methods[155] in model-free RL) and planning (for model-based RL) at multiple timescales akin to humans. As we will discuss later, when there are sample and computational constraints, such as in robotics, skills can improve acting, reasoning, and planning within and between tasks.

A semi-Markov decision process (SMDP)[120] provides a generalised framework supporting temporally abstract decision making in which the amount of time between two decision points is modelled as a random variable. Consider an agent in state s, following policy π, in the set of available policies

[2]The literature uses the term *skills* loosely and can also refer to regular action abstractions.

Π, with transit time k, before entering next state s'. In this setting, the state-transition probability from state s to s' could be expressed as $p(S^k = s'|S^0 = s, \pi)$. The accumulated discounted reward would be denoted by R_s^π. Considering discrete SMDPs, with discrete k, the SMDP Bellman equations for optimal state-value and action-value functions are given by:

$$v_*(s) = \max_{\pi \in \Pi} \left[R_s^\pi + \sum_{k=1}^{\infty} \gamma^{k-1} \mathbb{E}_{p(S^k=s'|S^0=s,\pi)}[v_*(s')] \right] \tag{2.6}$$

$$q_*(s,\pi) = R_s^\pi + \sum_{k=1}^{\infty} \gamma^{k-1} \mathbb{E}_{p(S^k=s'|S^0=s,\pi)}[\max_{\pi' \in \Pi} q_*(s',\pi')] \tag{2.7}$$

In theory, these abstractions enable the agent to ignore irrelevant details across time and space that are not necessary for decision making when reasoning over policies and their induced value functions for policy improvement. For example, when solving mazes, one strategises at the inter-, not intra-, corridor level. The states over which these strategies occur are often to referred to as *bottleneck states*. Intuitively, discovering these compressed temporal abstractions reduces the search space for sequential decision making, key for improving value estimation with respect to sample efficiency for model-free RL methods (as less strategies need evaluation in the environment) and computational efficiency for model-based RL approaches (due to search tree pruning).

These temporally extended policies, resulting in *temporally extended actions*, enable the discovery of abstractions that represent partial solutions to tasks that can be recomposed to efficiently solve a family of compositionally related problems. A key distinction between this framework and those presented in Section 2.3.1 is that temporal abstractions enables improved credit-assignment by supporting TD learning across multiple timescales. Credit-assignment is particularly problematic in long-horizon domains with sparse and delayed rewards. Robotic environments regularly exhibit these traits, obtaining rewards only upon task completion with learning and acting occurring at the micro-second temporal level. Continual (lifelong) learning, with an extremely long-horizon spanning the agent's entire lifetime, also suffers from this same issue.

The options framework present an expressive choice for temporal abstraction[156]. Markovian options $\omega \in \Omega$ can be expressed as tuple $\langle I_\omega, \pi_\omega, \beta_\omega \rangle$, with a markovian policy $\pi_\omega(a|s)$, termination condition

$\beta_\omega(s)$, and initiation set $I_\omega(s)$ commonly replaced with $I = 1 \forall s \in S$ following Bacon et al.[11]; Zhang and Whiteson[182]. In each state, a policy over options selects an option ω according to $\pi_C(\omega|s_t)$. The option ω then executes, producing actions according to π_ω until the termination condition is satisfied and terminates in s_{t+k}, upon which a successive option is sampled from $\pi_C(\omega|s_{t+k})$ (potentially constrained according to the initiation set).

While there exists a plethora of methods for discovering options, many focus on options that terminate at *bottleneck states* for the aforementioned reasons[91;145;150;53;124]. These have been shown beneficial for planning as the bottlenecks are the states most frequently visited when considering the shortest distance between any two states in an MDP[150]. While effective, the training regimes of these methods do not adhere to the requirements for *continual learning* in robotics. These approaches either do not support off-policy RL[124], crucial for sample efficiency, or are restricted to discrete state and/or action spaces[91;91;145;150;53], limiting their applicability in robotics. In contrast, in Chapter 6, we propose a model-free, off-policy framework for *transfer learning*, supporting continuous state-action spaces, that discovers *bottleneck states* and improves credit-assignment, value estimation, and policy performance when compared to the non-bottleneck counterpart[177].

The focus until now has been on discovering temporal abstractions unsupervised, as to support fully autonomous *continual learners*. Nevertheless, in many practical applications, additional supervision may be available to guide which abstractions are learnt and how to use them. In Chapter 4, we present an option-inspired approach that enables the discovery of temporal abstractions given weak supervision, enabling effective *domain adaptation* to new tasks through a human-robot command interface directing how to re-combine previously acquired skills to tackle novel challenges.

2.3.3 State Abstraction Focused

The central goal of abstractions is to capture common, task-relevant structures and suppress irrelevant ones (also considered as noise or distractors[133]) as to facilitate positive knowledge transfer. Li et al.[83] presents a unified theory for state abstractions in the MDP setting. Given MDP M, with abstracted version $\tilde{M}$, the abstraction function $\phi : S \rightarrow \tilde{S}$ maps states from the original MDP to those in the abstracted version. Useful abstractions are those that preserve information crucial for policy optimisation in the original or related MDPs if considering the *transfer learning* setting.

There exist multiple methods to discover such abstractions. PAC discovers state-abstractions that adhere to this property but only in the tabular RL setting[4]. Zhang et al.[179] discover task agnostic abstractions focusing on identifying causal states in POMDPs. Li et al.[83] discover abstractions that preserve underlying reward and transition models (across MDPs) leading to model-irrelevance abstractions. Zhang et al.[180] discover invariant abstractions for the block MDP[28] setting.

In contrast, in Chapter 3 we focus on abstractions suited for *sim2real domain adaptation*, a field of particular relevance to the robotics community, as it enables safe training and deployment of embodied reinforcement learners. We focus on state abstractions that generalise across image observations. Specifically, our representations are task-specific and invariant to, and generalise across, unseen distractors. In our setting, distractors are modular components of the observation-space that are task-independent (that an optimal policy should be invariant to). Unlike alternate approaches not designed for *sim2real domain adaptation*, we leverage privileged information available in simulation but not deployment, to guide abstraction learning. Compared with alternate *sim2real* approaches, our abstractions improve generalisation performance on domains with unseen distractors, key for safe and robust *sim2real* deployment.

Chapter 3

Self-Supervised State Abstractions For Domain Adaptation

We present a *state-abstraction* approach for *sim2real domain adaptation*, promoting the safe training and deployment of robotic machine learners. Our approach achieves compositional generalisation across modular image observations. Specifically, we generalise across distractors; object-level, task-independent, components of the observation space that the optimal policy should be invariant to. We leverage privileged information to sample-efficiently discover abstractions that are invariant to and generalise across distractors. Compared with alternate *sim2real* approaches, our abstractions improve generalisation to domains with unseen distractors, key for safe and robust *sim2real* deployment.

3.1 Introduction

While image-based deep reinforcement learning (RL) has recently provided significant successes in various high-data domains[96;144;84], its application to physical systems remains challenging due to expensive and slow data generation, challenges with respect to safety, and the need to be robust to unexpected changes in the environment.

When training visual models in simulation, we can obtain robustness either by adaptation to target domains[37;15;174], or by randomising system parameters with the aim of covering all possible environment parameter changes[164;123;105;116;131;168]. Unfortunately, training under a distribution of

randomised visuals [131;168], can be substantially more difficult due to the increased variability. This often leads to a compromise in final performance[105;164]. Furthermore, it is usually not possible to cover all potential environmental variations during training. Enabling agents to generalise to unseen visuals such as distractors (task-independent aspects of the observation-space that should not influence the optimal policy) is an important question in robotics where an agent's environment is often noisy (e.g. autonomous vehicles).

To increase robustness and reduce training time, we can make use of privileged information such as environment states, commonly accessible in simulators. By using lower-dimensional, more structured and informative representations directly as agent input, instead of noisy observations affected by visual randomisation, we can improve data efficiency and generalisation[157;112].

However, raw observations can be easier to obtain and dependence on privileged information during deployment can be restrictive. When exact states are available during training but not deployment, we can make use of information asymmetric actor-critic methods[116;139] to train the critic faster via access to the state while providing only images for the actor.

By introducing Attention-Privileged Reinforcement Learning (APRiL), we further leverage privileged information readily and commonly available during training, such as simulator states and object segmentations[166;27], for increased robustness, sample efficiency, and generalisation to distractors. As a general extension to asymmetric actor-critic methods, APRiL concurrently trains two actor-critic systems (one symmetric with a state-based agent, the other asymmetric with an image-dependent actor). Both actors utilise attention to filter their inputs, and we encourage alignment between both attention mechanisms. As state-space learning is unaffected by visual randomisation, the observation attention module efficiently attends to state- and task-dependent aspects of the image whilst explicitly becoming invariant to task-irrelevant and noisy aspects of the environment (distractors). We demonstrate that this leads to faster image-based policy learning and increased robustness to task-irrelevant factors (both within and outside the training distribution). See Figure 3.1 for a visualisation of APRiL, its attention, and generalisation capabilities on one of our domains.

In addition, APRiL shares a replay buffer between both agents, which further accelerates training for the image-based policy. At test-time, the image-based policy can be deployed without privileged

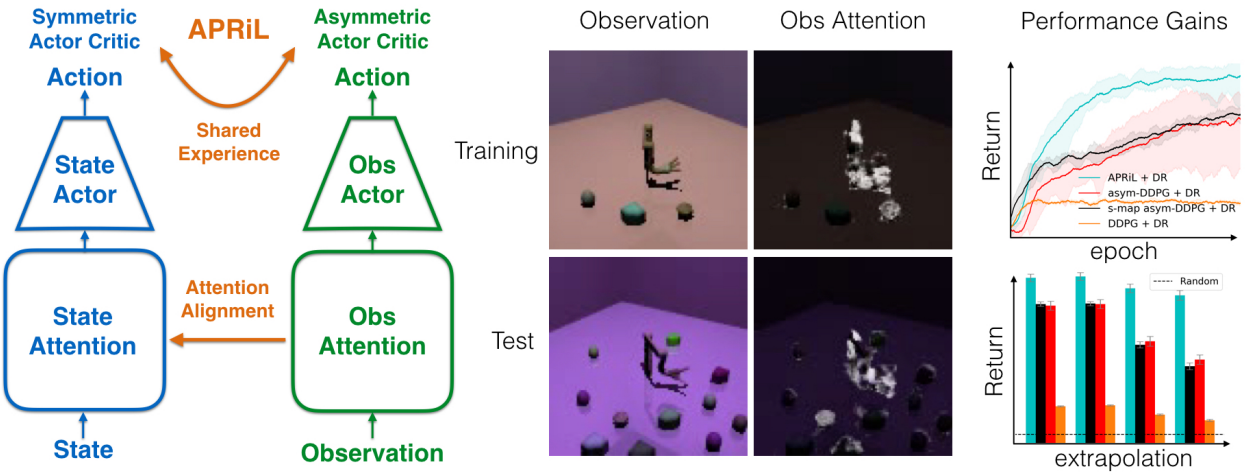


Figure 3.1: **Model diagram (left):** APRiL concurrently trains two attention augmented policies (one state-based, the other image-based). **Qualitative and quantitative results (middle & right):** By aligning the image attention to that of the state, image-based attention quickly suppresses highly varying, task-irrelevant, information **(middle second column)**. This leads to increased learning rate **(top right)** and robustness to extrapolated domains with increasing levels of unseen additional distractors **(bottom right)**. For *JacoReach*, attention (**middle second column**; white and black signify high and low values) is paid only to the target object and jaco arm in training and transfer domains.

information. We test our approach on a diverse set of simulated domains across robotic manipulation, locomotion, and navigation; and demonstrate considerable performance improvements compared to competitive baselines when evaluating on environments from the training distribution as well as in extrapolated and unseen settings with additional distractors.

3.2 Problem Formulation

Before introducing Attention-Privileged Reinforcement Learning (APRiL), this section provides a background for the RL algorithms used. For a more in-depth introduction please refer to Lillicrap et al.[84] and Pinto et al.[116].

3.2.1 Reinforcement Learning

We describe an agent's environment as a partially observable Markov decision process (POMDP) which is represented as the tuple $(S, O, A, P, r, \gamma, S_0)$, where S denotes a set of continuous states, A denotes a set of either discrete or continuous actions, $P : S \times A \times S \to \mathbb{R}_{\geq 0}$ is the transition probability function, $r : S \times A \to \mathbb{R}$ is the reward function, γ is the discount factor, and S_0 is the initial state distribution. O is a set of continuous observations corresponding to continuous states in S. At every time-step

t, the agent takes action $a_t = \pi(\cdot|s_t)$ according to its policy $\pi : S \to A$. The policy is optimised to maximize the expected return $\mathbb{E}[R_0|S_0]$, the discounted sum of future rewards $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$, and is dependent on π. The agent's Q-function is defined as $Q_\pi(s_t, a_t) = \mathbb{E}[R_t|s_t, a_t]$.

3.2.2 Asymmetric Deep Deterministic Policy Gradients

Asymmetric Deep Deterministic Policy Gradients (asymmetric DDPG)[116] represents a type of actor-critic algorithm designed specifically for efficient learning of a deterministic, observation-based policy in simulation. This is achieved by leveraging access to more compressed, informative environment states, available in simulation, to speed up and stabilise training of the critic.

The algorithm maintains two neural networks: an observation-based actor or policy $\pi_\theta : O \to A$ (with parameters θ) used during training and test time, and a state-based Q-function (also known as critic) $Q^\pi_\phi : S \times A \to \mathbb{R}$ (with parameters ϕ) which is only used during training.

To enable exploration, the method (like its symmetric version[142]) relies on a noisy version of the policy (called behavioural policy), e.g. $\pi_b(o) = \pi(o) + z$ where $z \sim \mathcal{N}(0, 1)$ (see Section 8.1.3 for our particular instantiation). The transition tuples $(s_t, o_t, a_t, r_t, s_{t+1}, o_{t+1})$ encountered during training are stored in a replay buffer[96]. Training examples sampled from the replay buffer are used to optimise the critic and actor. By minimizing the Bellman error loss $\mathcal{L}_{critic} = (Q(s_t, a_t) - y_t)^2$, where $y_t = r_t + \gamma Q(s_{t+1}, \pi(o_{t+1}))$, the critic is optimised to approximate the true Q values. The actor is optimised by minimizing $\mathcal{L}_{actor} = -\mathbb{E}_{s,o\sim\pi_b(o)}[Q(s, \pi(o))]$.

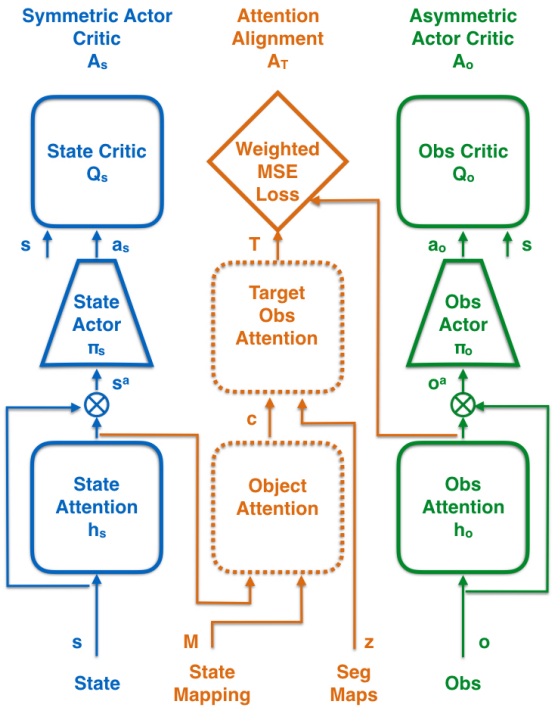


Figure 3.2: **APRiL's architecture**. Blue, green and orange represent symmetric and asymmetric actor-critic and attention alignment modules (A_s, A_o, A_T). The diamond represents the attention alignment loss. Dashed and solid blocks are non-trainable and trainable networks. The $\otimes$ operator signifies element-wise multiplication. Experiences are shared using a shared replay buffer.

3.3 Attention-Privileged Reinforcement Learning (APRiL)

APRiL improves the robustness and sample efficiency of an observation-based agent by using multiple ways to benefit from privileged information. First, we use an asymmetric actor-critic setup[116] to train the observation-based actor. Second, we additionally train a quicker learning state-based actor, while sharing replay buffers, and aligning attention mechanisms between both actors. Specifically, APRiL aligns attention mechanisms at the object-level defined by the simulator[166;27] aiding knowledge transfer, generalisation and robustness over this semantic space. In common simulators, such as MuJoCo[166], objects are often defined as *geoms*, the primitive geometric bodies that combined create the simulation environment (such as walls, individual links or digits in a Kinova robotic arm, or limbs for a humanoid). We emphasise here that our approach can be applied to any asymmetric, off-policy, actor-critic method[72] with the expectation of similar performance benefits to those demonstrated in this chapter. Specifically we choose to build off Asymmetric DDPG[116] due to its accessibility.

APRiL is comprised of three modules as displayed in Figure 3.2. The first two modules, A_s and A_o, each represent a separate actor-critic with an attention network incorporated over the input for each actor. For the *state-based* module A_s we use standard symmetric DDPG, while the *observation-based* module A_o builds on asymmetric DDPG, with the critic having access to states. Finally, the third component A_T represents the object-level alignment process between attention mechanisms of both actor-critic agents to more effectively transfer knowledge between both learners respectively.

A_s consists of three networks: Q_s^{π}, π_s, h_s (critic, actor, and attention) with parameters $\{\phi_s, \theta_s, \psi_s\}$. Given input state s_t, the attention network outputs a soft gating mask h_t of same dimensionality as the input, with values ranging between $[0,1]$. The input to the actor is an attention-filtered version of the state, $s_t^a = h_s(s_t) \odot s_t$. To encourage a sparse masking function, we found that training this attention module on both the traditional DDPG loss as well as an entropy loss helped:

$$\mathscr{L}_{h_s} = -\mathbb{E}_{s\sim\pi_b}[Q_s(s, \pi_s(s^a)) - \beta H(h_s(s))], \tag{3.1}$$

where β is a hyperparameter (set through grid-search, see Section 8.1.3) to weigh the additional entropy objective, and π_b is the behaviour policy that obtained experience (in this case from a shared replay buffer). The actor and critic networks π_s and Q_s are trained with the symmetric DDPG actor

and Bellman error losses. We found that APRiL was not sensitive to the absolute value of β, only the magnitude, and was set low enough to not suppress task-relevant parts of the state-space.

Within A_T, the state-attention obtained in A_s is converted to corresponding observation-attention T to act as a self-supervised target for the observation-attention module in A_o. This is achieved in a two-step process. First, state-attention $h_s(s)$ is converted into object-attention c, which specifies how task-relevant each object in the scene is. The procedure uses information about which dimension of the environment state relates to which object. Second, object-attention is converted to observation-space attention by performing a weighted sum over object-specific segmentation maps[1]:

$$c = M \cdot h_s(s), \qquad T = \sum_{i=0}^{N-1} c_i \cdot z_i \tag{3.2}$$

Here, $M \in \{0,1\}^{N \times n_s}$ (n_s is the dimensionality of s) is an environment-specific, predefined adjacency matrix that maps the dimensions of s to each corresponding object, and $c \in [0,1]^N$ is an attention vector over the N objects in the environment. c_i corresponds to the i^{th} object attention value. $z_i \in \{0,1\}^{W \times H}$ is the binary segmentation map of the i^{th} object segmenting the object with the rest of the scene, and has the same dimensions as the image. z_i assigns values of 1 for pixels in the image occupied by the i^{th} object, and 0 elsewhere. $T \in [0,1]^{W \times H}$ is the converted state- to observation-space attention to act as a target on which to train the observation-attention network h_o.

The observation module A_o also consists of three networks: Q_o^π, π_o, h_o (respectively critic, actor, and attention) with parameters $\{\phi_o, \theta_o, \psi_o\}$. The structure of this module is the same as A_s except the actor and critic now have asymmetric inputs. The actor's input is the attention-filtered version of the observation, $o_t^a = h_o(o_t) \odot o_t$ [2]. The actor and critic π_o and Q_o are trained with the asymmetric DDPG actor and Bellman error losses in Section 3.2.2. The main difference between A_o and A_s is that the observation attention network h_o is trained on both the actor loss and an object-weighted mean squared error loss:

$$\mathscr{L}_{h_o} = \mathbb{E}_{o,s \sim \pi_b}[\frac{1}{2} \sum_{ij} \frac{1}{w_{ij}} (h_o(o) - T)_{ij}^2 - \nu Q_o(s, \pi_o(o^a))] \tag{3.3}$$

[1] Simulators (e.g. [166;27]) commonly provide functionality to access these segmentations and semantic information for the environment state.

[2] In practice, the output of $h_o(o_t)$ is tiled to match the number of channels that the image contains

where weights w_{ij} denote the fraction of the image o that the object present in $o_{i,j,1:3}$ occupies, and ν represents a hyperparameter for the relative weighting of both loss components (see Section 8.1.3 for exact value). The weight terms, w, ensure that the attention network becomes invariant to the size of objects during training and does not simply fit to the most predominant object in the scene.

During training, experiences are collected evenly from both state- and observation-based agents and stored in a shared replay buffer (similar to Schwab et al.[139]). This is to ensure that: 1. Both critics Q_s, Q_o observe states that would be visited by either of their respective policies. 2. The attention modules h_s and h_o are trained on the same data distribution to better facilitate alignment. 3. Efficient discovery of highly performing states from π_s are used to speed up learning of π_o.

Algorithm 1 shows the pseudocode for a single actor implementation of APRiL. In practice, in order to speed up data collection and gradient computation, we parallelise the agents and environments and ensure equal data is generated by state- and image-based agents.

Algorithm 1 Attention-Privileged Reinforcement Learning

Initialize the actor-critic modules A_s, A_o, attention alignment module A_T, replay buffer R
for episode$= 1$ **to** M **do**
  Initial state s_0
  Set DONE $\leftarrow$ FALSE
  while $\neg$ DONE **do**
    Render image observation o_t and segmentation maps z_t:
      $o_t, z_t \leftarrow \text{renderer}(s_t)$
    if episode **mod** $2 = 0$ **then**
      Obtain action a_t using observation-behavioural policy and obs-attention network:
        $a_t \leftarrow \pi_o(h_o(o_t) \odot o_t)$
    else
      Obtain action a_t using state-behavioural policy and state-attention network:
        $a_t \leftarrow \pi_s(h_s(s_t) \odot s_t)$
    end if
    Execute action a_t, receive reward r_t, DONE flag, and transition to s_{t+1}
    Store $(s_t, o_t, z_t, a_t, r_t, s_{t+1}, o_{t+1})$ in R
  end while
  for $n = 1$ **to** N **do**
    Sample minibatch $\{s, o, z, a, r, s', o'\}_0^B$ from R
    Optimise state- critic, actor, and attention using $\{s, a, r, s'\}_0^B$ with A_s
    Convert state-attention to target observation-attention $\{T\}_0^B$ using $\{s, o, z\}_0^B$ with A_T
    Optimise observation- critic, actor, and attention using $\{s, o, T, a, r, s', o'\}_0^B$ with A_o
  end for
end for

3.4 Experiments

We investigate the following to evaluate how well APRiL facilitates transfer across visually distinct domains: Does APRiL: 1. Increase **sample-efficiency** during training? 2. Affect **interpolation** performance on unseen environments from the training distribution? 3. Affect **extrapolation** performance on environments outside the training distribution?

3.4.1 Environments

We evaluate APRiL over the following environments (see Section 8.1.1 for more details): 1. *NavWorld*: the circular agent is sparsely rewarded for reaching the triangular target in the presence of distractors. 2. *JacoReach*: the Kinova arm is rewarded for reaching the diamond-shaped object in the presence of distractors. 3. *Walker2D*: this slightly modified (see Section 8.1.1) Deepmind Control Suite environment[157] the agent is rewarded for walking forward, keeping its torso upright.

Training (Source) Domains

During training we perform Domain Randomisation (DR)[164;131], randomising the following environment parameters to enable generalisation with respect to them: camera position, orientation, textures, materials, colours, object locations, background (see Section 8.1.2). For *NavWorld* and *JacoReach*, distractor objects are randomly sampled from an object catalogue between episodes (from a subset of ShapeStacks[43] objects for *JacoReach* and 4 to 8 sided shapes for *NavWorld* - see Section 8.1.1). The distractors have distinct shapes to the target object.

Interpolated Transfer Domains

We evaluate performance on environments unseen during training but within the training distribution (e.g. interpolated camera position, orientation, object colours - see Section 8.1.2). For *NavWorld* and *JacoReach*, the interpolated environments have the same number of distractors, using the same sampling procedure as before. Please refer to Figure 3.5 for examples of the interpolated domains.

Extrapolated Transfer Domains

For *NavWorld* and *JacoReach*, we investigate how well each method generalises to extrapolated domains with **additional distractor** objects (specifically 4 or 8; referred as **ext-4** and **ext-8** in Sec-

tion 3.5) using the same distractor sampling procedure as the source and interpolated domains. We explore generalisation to environments with increased clutter as this forms a common scenario that many sim2real approaches wish to handle[44]. The textures and colours of these distractor objects are sampled from a held-old set not seen during training. The locations are sampled randomly for *NavWorld*, and from extrapolated arcs of two concentric circles of different radii for *JacoReach*. We do not extrapolate for *Walker2D*, as this domain does not contain distractors. However, if APRiL is beneficial during DR its application does not need to be restricted to environments with clutter. Please refer to Figure 3.5 for examples of the extrapolated domains. See Section 8.1.2 for more details.

3.4.2 Baselines

We start by comparing APRiL against two competitive baselines that also leverage privileged information during training. We compare against *Asymmetric DDPG* (asym-DDPG) baseline[116] to evaluate the importance of privileged attention and shared replay for learning and robustness to distractors. Our second baseline, *State-Mapping Asymmetric DDPG* (s-map asym-DDPG), introduces a bottleneck layer trained to predict the environment state using an L_2 loss. This is another intuitive approach that further exploits state information in simulation[181] to learn informative representations that are robust to visual randomisation. This approach does not incorporate object-centric attention or leverage privileged object segmentations. We note that since this baseline learns state estimation it is not expected to extrapolate well to domains with additional distractor objects and varying state-spaces (with respect to the training domain). We also compare APRiL with DDPG to emphasise the difficulty of these DR tasks if privilege information is not leveraged.

3.4.3 Ablations

We perform an ablation study to investigate which components of APRiL contribute to performance gains. The ablations consist of: 1. *APRiL no sup*: the full setup except without attention alignment. Here the observation-attention module must learn without guidance from the state-agent. 2. *APRiL no share*: APRiL without a shared replay. 3. *APRiL no back*: uniform object-attention values c are used to train the observation-attention module, thereby only suppressing the background. Here we investigate the importance of object suppression for generalisation.

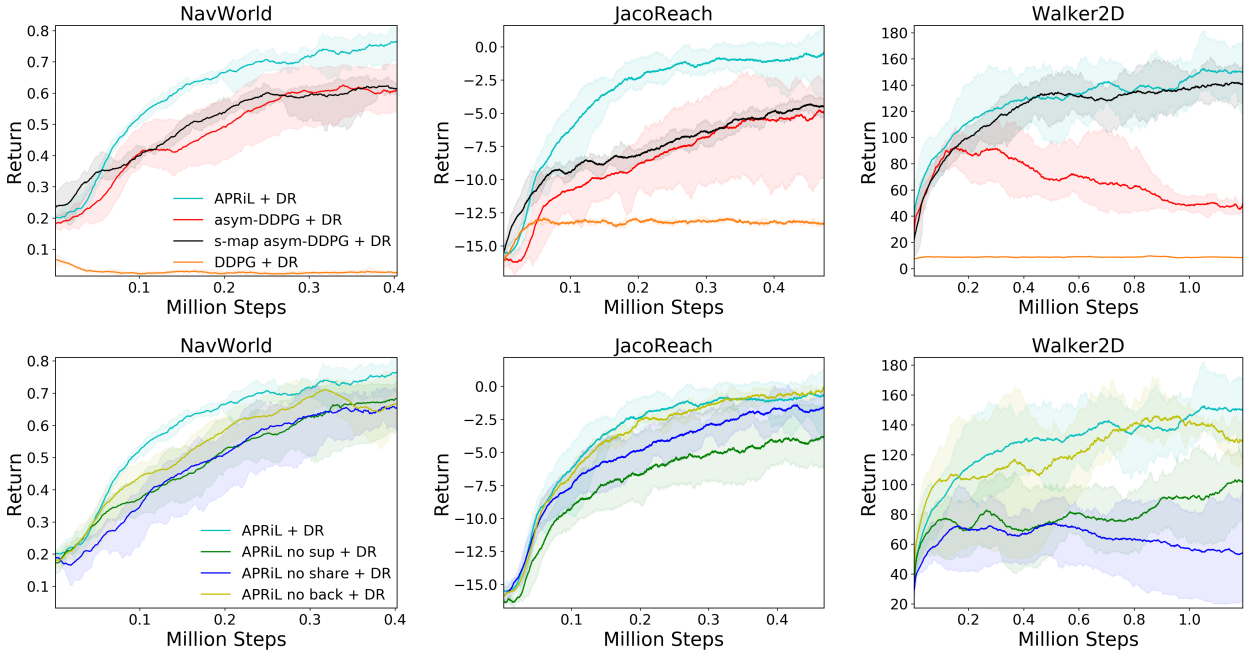


Figure 3.3: Learning curves for observation-based policies during Domain Randomisation (DR). **Top row**: comparison with baselines. **Bottom row**: comparison with ablations. **Solid line**: mean performance. **Shaded region**: covers minimum and maximum performances across 5 seeds. APRiL's attention and shared replay lead to stronger or commensurate performance.

3.5 Results

3.5.1 Performance On The Training Distribution

Figure 3.3 shows that APRiL outperforms the baselines for each environment (except *Walker2D* where it matches s-map asym-DDPG). The ablations in Figure 3.3 show that a shared replay buffer, background suppression, and attention alignment each individually provide benefits but are most effective when combined together. Interestingly, background suppression is extremely effective for sample-efficiency, as for these domains the majority of the irrelevant, highly varying, aspects of the observation-space are occupied by the background. It is also surprising that the s-map asym-DDPG baseline, which learns to map to environment states, does not outperform asym-DDPG and does not match APRiL's performance for *NavWorld* and *JacoReach*. For these domains, predicting states (including those of distractors) is difficult[3] and prediction errors limit policy performance. For *Walker2D*, in the absence of distractor objects, s-map asym-DDPG is a competitive baseline and APRiL provides marginal gains.

[3]For *JacoReach* prediction errors and policy performance are sensitive to state-space. In Figure 3.3 we plot the best performing state-space. Refer to Section 8.1.5 for further details.

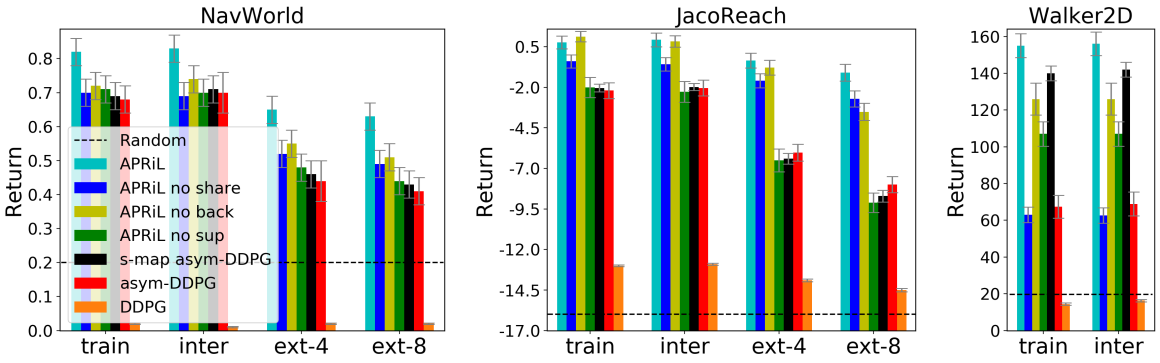


Figure 3.4: Comparing average return of the image-policy between training, interpolated and extrapolated domains (100 each). Plots reflect mean and 2 standard deviations for average return (5 seeds). APRiL generalises due to its attention and outperforms the baselines. We compare against a random agent to gauge the degree of degradation in policy performance between domains.

3.5.2 Interpolation: Transfer Domains From The Training Distribution

Fig. 3.4 plots the return on these held-out domains. For all algorithms, we observe minimal degradation in performance between training and interpolated domains. However, as APRiL outperforms on the training distribution (apart from s-map asym-DDPG for *Walker2D* and APRiL no back for *JacoReach*), its final performance on the interpolated domains is significantly better, emphasising the benefits of both privileged attention and a shared replay.

3.5.3 Extrapolation: Transfer Outside The Training Distribution

Fig. 3.4 shows that APRiL generalises and performs considerably better on the held-out domains than each baseline. Specifically, when comparing with the baselines that leverage privilege information, for *JacoReach* performance falls by 11%[4] for APRiL instead of 42% and 48% for asym-DDPG and s-map asym-DDPG respectively. The ablations demonstrate that effective distractor suppression is crucial for generalisation. This is particularly prominent for *JacoReach* where the performance drop for the methods that use attention alignment (APRiL and APRiL no share) is 11% and 15%, which is far less than 27% and 51% (APRiL no back and APRiL no sup) for those that do not learn to effectively suppress distractors.

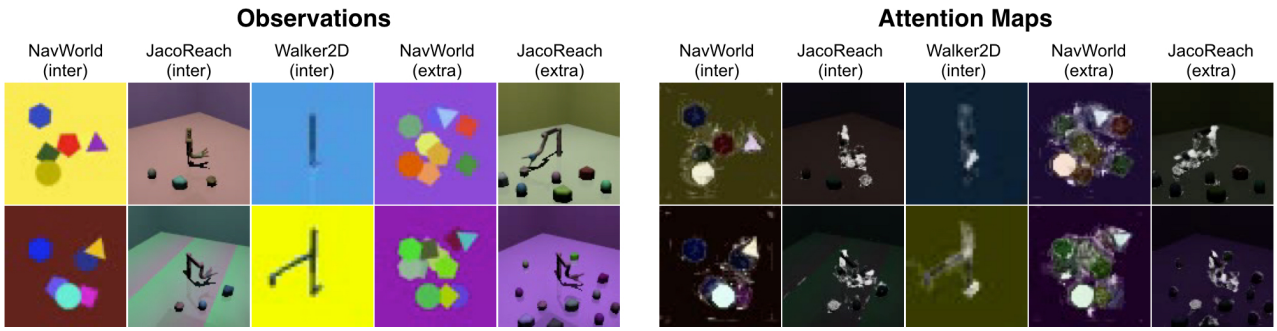


Figure 3.5: Held-out domains and APRiL's attention maps. For the extrapolated domain columns (extra), top and bottom represent **ext-4** and **ext-8**. White/black signify high/low attention values. Attention suppresses the background and distractors aiding at policy generalisation.

3.5.4 Attention Module Analysis

We visualise APRiL's attention maps (Figures 3.5, 8.2 and 8.3) on both interpolated and extrapolated domains. For *NavWorld*, attention is correctly paid to all relevant aspects (agent and target; circle and triangle respectively) and generalises well. For *JacoReach*, attention suppresses the distractors even on the extrapolated domains, achieving robustness with respect to them. Interestingly, as we encourage sparse attention, APRiL learns to only pay attention to every-other-link of the arm (as the state of an unobserved link can be inferred by observing those of the adjacent links). For *Walker2D*, dynamic object attention is learnt (different objects are attended based on the state of the system - see Figure 8.3). When upright, walking, and collapsing, APRiL pays attention to the lower limbs, every other link, and foot and upper body, respectively. We suspect that in these scenarios, the optimal action depends most on the state of the lower links (due to stability), every link (coordination), and foot and upper body (large torque required), respectively.

3.6 Related Work

A large body of work investigates the problem of learning robust policies that generalise well outside of the training distribution. Work on **transfer learning** leverages representations from one domain to efficiently solve a problem from a different domain[26;106;130]. In particular, many **domain adaptation** techniques aim to adapt a learned model to a specific target domain(s), often optimising models such that representations are invariant to the shift in the target domain [37;87;15;174]. These methods commonly require data from the target domain in order to transfer and adapt effectively.

In contrast, **domain randomisation** covers a distribution of environments by randomising visual[164]

[4]Percentage decrease is taken with respect to additional return over a random agent on the training domain.

or dynamical parameters[112] during training in order to generalise[131;123;168;57;105;78]. In doing so, such methods shift the focus from adaptation to specific environments to **generalisation and robustness** by covering a wide range of variations. Recent work automatically varies this distribution during training[7] or trains a canonical invariant image representation[62]. However, while randomisation can enable us to learn robust policies, it significantly increases training time due to the increased environment variability[105], and can reduce asymptotic performance. Our work partially addresses this fact by training two agents, one of which is not affected by visual randomisations.

Other works explicitly encourage representations **invariant** to observation-space variations[148;152;62]. Contrastive techniques[148;152] use a clear separation between positive (and negative) examples, predefined by the engineer, to encourage **invariance**. Unlike APRiL, these **invariances** are over abstract spaces and are not designed to exploit **privileged information**; shown to be beneficial by APRiL's ablations. Furthermore, APRiL's **invariance** is **task-driven** via attention. Approaches like James et al.[62]; Zhang et al.[181], learn invariances supervised, mapping from observation to a predefined space. Unlike APRiL, these methods are unable to discover task-independent aspects of the mapping-space, limiting robustness and generalisation. Finally, unlike APRiL as a model-free RL approach, some model-based works use forward or inverse models[111;50;136] to achieve invariance.

Existing comparisons in the literature demonstrate that, even without domain randomisation, the increased dimensionality and potential partial observability complicates learning for RL agents[157;139]. In this context, accelerated training has also been achieved by using **access to privileged information** such as environment states to asymmetrically train the critic in actor-critic RL[139;116;33]. In addition to using additional information to train the critic, Schwab et al.[139] use a **shared replay buffer** for data generated by image- and state-based actors to further accelerate training for the image-based agent. Our method extends these approaches by sharing information about relevant objects by aligning agent-integrated attention mechanisms between an image- and state-based actors.

Recent experiments have demonstrated the strong dependency and interaction between attention and learning in human subjects[80]. In the context of machine learning, **attention mechanisms** have been integrated into RL agents to increase robustness and enable interpretability of an agent's behaviour[151;97]. In comparison, we focus on utilising the attention mechanism as an interface to transfer information between two agents to enable faster training and improved generalisation.

3.7 Conclusion

We introduce Attention-Privileged Reinforcement Learning (APRiL), an extension to asymmetric actor-critic algorithms that leverages attention mechanisms and access to privileged information such as simulator environment states. The method benefits in two ways in addition to asymmetry between actor and critic: via aligning attention masks between image- and state-space agents, and by sharing a replay buffer. Since environment states are not affected by visual randomisation, we are able to learn efficiently in the image domain especially during domain randomisation where feature learning becomes increasingly difficult. Evaluation on a diverse set of environments demonstrates significant improvements over competitive baselines including asym-DDPG and s-map asym-DDPG; and show that APRiL learns to generalise favourably to environments not seen during training (both within and outside of the training distribution). Finally, we investigate the relative importance of the different components of APRiL in an extensive ablation.

Chapter 4

Weakly-Supervised Temporal Abstractions For Domain Adaptation

Chapter 3 introduces a *domain adaptation* approach for generalisation across *sim2real* observations, key for safe and robust robotic machine learners. We continue by exploring *domain adaptation* across modular tasks, crucial for multifaceted robots, such as self-driving cars navigating novel routes. Specifically, we introduce a method leveraging *temporal-abstractions* for compositional generalisation. Our approach uses weak-supervision to learn grounded, interpretable, *temporal-abstractions* corresponding to sub-tasks. Additionally, we provide a *human-robot* communication interface for generalisation across sub-task orderings. Our approach improves generalisation and sample-efficiency when compared to fully-supervised methods.

4.1 Introduction

Learning from demonstration (LfD) represents a popular paradigm to teaching complex behaviours to robots and virtual agents through demonstrations, without the need for explicit programming or other description of a task, such as a cost function[10]. The benefits of LfD over manual task definitions are numerous. First, some behaviours are difficult to program or manually encode, but can be easily demonstrated. Second, while programming behaviours requires expert knowledge of the application platform, LfD requires only the ability to control the agent.

However, complex real world tasks pose a severe challenge for simple LfD approaches such as *behavioural cloning* (BC). While a complex task can often be broken down into simpler sub-tasks, the algorithm itself lacks the means to discover the decomposition and must instead learn a monolithic policy for the whole task. The resulting policies are not only more complex but also less reusable. For example, a cube stacking task can be broken down into sub-tasks for approaching a cube, grasping it, moving to a location, and placing it onto an existing stack. In particular, sub-policies for approaching and grasping can be reused in other manipulation tasks such as cube rotation or throwing.

Modular LfD addresses this challenge by modelling the task as a composition of sub-tasks for which reusable sub-policies (modules) are learned. These sub-policies are often easier to learn and can be composed in different ways to execute new tasks, enabling zero-shot imitation. One approach to modular LfD is to provide the learner with additional information about the demonstrations. This comes in many forms, e.g., manual segmentation of demonstrations[59], interactive feedback during learning[104], or prior knowledge about the task. Such domain knowledge may come in the form of motion primitives, hard-coded parametric motion models[30;135] and problem-specific state modelling[2;73]. An additional benefit is that since these methods ground demonstrations on human-defined sub-tasks, the learned policies are interpretable. They are, however, labour intensive, perform sub-optimally if the underlying assumptions are incorrect, and require domain knowledge.

In this chapter, we consider a general, weakly supervised, modular LfD setting where demonstrations are augmented only with a *task sketch*. This sketch describes the sequence of *sub-tasks* that occur within the demonstration, without additional information on their alignment (see Figure 4.1). Drawing inspiration from speech recognition[39] as well as modular reinforcement learning[9], we introduce *temporal alignment for control* (TACO), an efficient, domain agnostic algorithm that learns modular and grounded policies from high level task descriptions, while relying purely on weak supervision. Instead of considering the alignment of a demonstration with the task sketch and the learning of associated sub-policies as two separate processes, in TACO the imitation learning stage affects the alignment and vice-versa by maximising the joint likelihood of the observed sketch and the observed action sequence given the states. TACO learns one sub-policy for each sub-task present in the data and extends each sub-policy's action space to enable self-termination. At test time, the agent is presented with new, potentially unseen, and often longer sketches, which it executes by composing the required

sub-policies. In addition to simplifying the imitation of complex tasks, the approach enables zero-shot imitation given only a sketch.

We evaluate the performance of TACO on four domains of varying complexity. First, we consider two toy domains, a 2D navigation task and the Craft domain proposed by Andreas et al.[9]. Finally, we consider the scenario of controlling a simulated robot arm to use a number pad and extend the task to use only image-based observations. We demonstrate that, in all domains, policies trained using TACO are capable of matching the performance of policies trained using a fully supervised method, where the segmentation of the demonstration is provided, at a small fraction of the labelling cost. At the same time, the approach significantly outperforms our baselines which separate the optimisation processes for segmentation and imitation.

4.2 Related Work

Learning from demonstration encompasses a wide range of techniques that focus on learning to solve tasks based on (human) expert demonstrations[10]. The fields of modular and hierarchical LfD aim to extract reusable policy primitives from complex demonstrations to increase data efficiency and transfer knowledge between tasks.

In robotics, sub-policies can be modelled as *motion primitives*[135] which build the foundation for various works on modular LfD[103;89;110]. In this context, most similar to the ideas underlying our approach is recent work based on *skill trees*[73] and semantically grounded finite representations[104]. However, these approaches consider separate segmentation of the trajectories and fitting of primitives and imposes stronger constraints on the type of the controllers. In contrast, our work addresses segmenting the demonstrations and learning the policies in one combined process.

Interleaving the two processes has been shown to provide better segmentations and policies in recent work[85]. The approach however considers learning via policy embeddings in task space building on probabilistic motion primitives[107], which restricts the approach with respect to the types of tasks and observations. The method presented in this chapter is less constrained and can handle arbitrary differentiable function approximator for the control policies.

Recent work on hierarchical LfD transfers concepts from the *options* framework[156], which mod-

els low-level policies as actions for a meta controller, to LfD[34;74;58]. Generally, options serve as tools for dividing a complex task into multiple sub-policies specialised for regions in the state space. TACO differs from option discovery frameworks[34;74] during both training and inference by replacing the functionality of the meta-controller with weak supervision in the form of a sequence of symbols. Weak supervision constrains the learned policies to follow the description in the task sketch. This prevents degenerate cases including high-frequency switching between policies common with options[74] as well as the potential collapse of the meta-controller to apply only a single option. Furthermore, by applying task sketches at test time, we enable the composition of sub-policies in unseen and longer sequences for zero-shot imitation.

Recent work on *modular reinforcement learning* (RL)[9], introduces the notion of sketches as additional information representing the decomposition of tasks. Similar to our work, Andreas et al.[9] assume that complex tasks can be broken down into sub-tasks. Our approach exploits a similar modular structure but utilises an imitation based objective that addresses the problem of aligning sequences of different lengths.

A common approach to sequence alignment in speech recognition is *connectionist temporal classification* (CTC)[39]. Previous extensions of CTC have been proposed to increase its flexibility by reducing the assumptions underlying the framework[38] and exploiting structure in the input space[60]. In this chapter, we extend CTC by combining sequence alignment and behavioural cloning.

4.3 Preliminaries

In this section, we introduce the required concepts and methods for the derivation of TACO.

4.3.1 Behavioural Cloning

Behavioural cloning (BC) models LfD as a supervised learning problem, by optimising a policy π to maximise the likelihood of the training dataset $\mathscr{D} = \{\rho_1, \rho_2, ..., \rho_M\}$. $\rho = ((s_1, a_1), (s_2, a_2), ..., (s_T, a_T))$ is a state-action demonstration trajectory of T pairs of states $s \in \mathscr{S}$ and actions $a \in \mathscr{A}$. Let $\pi_\theta(a|s)$ be the probability of taking action a in state s as modelled by a policy

π_θ parameterised by θ. BC performs the following optimisation:

$$\theta^* = \arg\max_\theta \mathbb{E}_\rho[\sum_{t=1}^{T} \log \pi_\theta(a_t|s_t)]. \tag{4.1}$$

One drawback of BC is its susceptibility to *covariate shift*, which occurs when small errors during testing cause the agent to drift away from states it encountered during training, yielding poor performance. One way to overcome this problem is using *disturbances for augmenting robot trajectories* (DART)[77], which introduces noise in the data collection process, allowing the agent to learn actions that can recover from errors. In this chapter, unless stated, we use a DART training approach.

The standard formulations of BC and DART, which learn only one policy per task, lack two important properties. The first is modularity: the demonstrated behaviour can have a hierarchical structure that decomposes into modules, or sub-policies. The second is reuse: the modules can be composed in various ways to perform different tasks.

4.3.2 Modular Learning from Demonstration

Modular LfD introduces modularity and reuse to the LfD problem. A schematic is shown in Figure 4.1. To render the policies reusable, it assumes that any task can be solved by multiple sub-policies, each of which operates in an augmented action space $\mathscr{A}^+$ that includes a STOP action (i.e., $\mathscr{A}^+ := \mathscr{A} \cup a_{STOP}$) that does not have to be observed in the demonstration. It also assumes that more than one sub-policy may be present per demonstration. In our formulation, extra information is provided via a *task sketch* $\tau = (b_1, b_2, \ldots, b_L)$, with $L \leq T$, and $b_l \in \mathscr{B}$, where $\mathscr{B} = \{1, 2, \ldots, K\}$, is a dictionary of sub-tasks. The sketch indicates which sub-tasks are active in a trajectory.

Although the a_{STOP} action is never observed, it can be inferred from the data. If the demonstration contains a simple task then a_{STOP} is called only at the end of the demonstration. If $L = T$, then we know which policy from $\mathscr{B}$ is active at each time-step. i.e., a_{STOP} for each policy is called as soon as the active policy changes within the demonstration. If all extra information is available, we can perform behavioural cloning, with two differences. First, maximising the likelihood takes place assuming an action-augmented policy $\pi(a^+|s)$. Second, we learn $|\mathscr{B}| = K$ modular policies π_{θ_k} from K datasets $\mathscr{D}_k$ containing trajectories ρ_k as segmented based on τ:

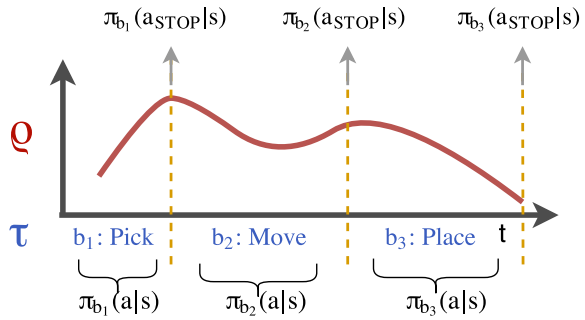


Figure 4.1: Problem setting: The trajectory ρ (red) is augmented by a task sketch τ (blue). The two sequences operate at different timescales. The whole trajectory is aligned (manually or automatically) and segmented into three parts. From this alignment three separate policies are learned. The unobserved a_{STOP} action for each policy is inferred to occur at the point where the policies switch from one to the other.

$$\theta^*_{k=1,\dots,K} = \arg\max_{\theta_k} \mathbb{E}_{\rho_k}[\sum_{t=1}^{T_\rho} \log \pi_{\theta_k}(a_t^+|s_t)]. \tag{4.2}$$

However, the fully supervised approach is labour intensive as each trajectory must be manually segmented into sub-policies. In this chapter, we consider cases where $L \ll T$ and τ contains only the sequence of active sub-tasks in the order they occur, without duplicates. Inferring when a_{STOP} occurs is therefore more challenging as τ and ρ operate at different timescales and must first be aligned.

4.3.3 Sequence Alignment

Since $L \ll T$, we cannot independently maximise the likelihood of active sub-tasks in τ for every time-step in ρ. The problem of aligning sequences of different lengths, a common challenge in speech recognition, is often addressed via *connectionist temporal classification* (CTC)[39]. Here, we review a variant of CTC adapted to the notation introduced so far, which does not include gaps between the predictions, based on the assumption that sub-tasks occur in sequence without pause.

A *path* $\zeta = (\zeta_1, \zeta_2, ..., \zeta_T)$ is a sequence of sub-tasks of the same length as the input sequence ρ, describing the active sub-task ζ_t from the dictionary $\mathscr{B}$ at every time-step. The set of all possible paths $\mathscr{Z}_{T,\tau}$ for a task sketch τ is the set of paths of length T that are equal to τ after removing all adjacent duplicates. For example, after removing adjacent duplicates, the path $\zeta = (b_1, b_1, b_2, b_3, b_3, b_3)$ equals the sketch $\tau = (b_1, b_2, b_3)$. The CTC objective maximises the probability of the sequence τ given the

input sequence ρ:

$$\psi^* = \arg\max_{\psi} \mathbb{E}_{(\rho,\tau)}[p_\psi(\tau|\rho)] \tag{4.3}$$

$$= \arg\max_{\psi} \mathbb{E}_{(\rho,\tau)}\Big[\sum_{\zeta \in \mathscr{Z}_{T,\tau}} p_\psi(\zeta|\rho)\Big] \tag{4.4}$$

$$= \arg\max_{\psi} \mathbb{E}_{(\rho,\tau)}\Big[\sum_{\zeta \in \mathscr{Z}_{T,\tau}} \prod_{t=1}^{T} p_\psi(\zeta_t|\rho)\Big] \tag{4.5}$$

where $p_\psi(\zeta_t|\rho)$ is commonly represented by a neural network parameterised by ψ that outputs the probability of each sub-task in $\mathscr{B}$. While naively computing (4.3) is infeasible for longer sequences, dynamic programming provides a tractable solution. Let $\mathscr{Z}_{t,\tau_{1:l}}$ be the set that includes paths $\zeta_{1:t}$ of length t corresponding to task sketches $\tau_{1:l}$ of length l, and $\alpha_t(l) = \sum_{\zeta_{1:t} \in \mathscr{Z}_{t,\tau_{1:l}}} p(\zeta|\rho)$ be the probability of being in task b_l at time-step t in the graph in Figure 4.2a. The probability of a task sketch given the input sequence $p(\tau|\rho)$ is equal to $\alpha_T(L)$.

We can recursively compute $\alpha_t(l)$ based on $\alpha_{t-1}(l)$, $\alpha_{t-1}(l-1)$, and the probability of the current sub-task $p(\zeta_t|\rho_t)$. As τ begins with a specific sub-task, the initial α's are deterministic and the probability of starting in the corresponding policy b_1 at time-step $t = 1$ is 1. Figure 4.2a depicts the recursive computation of the forward terms which is mathematically summarised as:

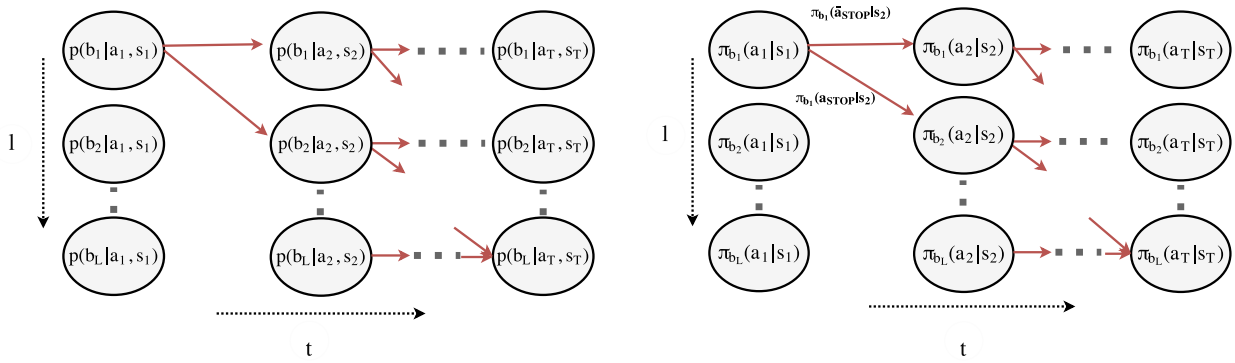
$$\alpha_t(l) = p(b_l|\rho_t)[\alpha_{t-1}(l-1) + \alpha_{t-1}(l))], \tag{4.6}$$

$$\alpha_1(l) = \begin{cases} 1, & \text{if } l = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{4.7}$$

Based on the recursive computation of the CTC objective in (4.6) and any automatic differentiation framework, we can optimise our model. For the manual derivation of the gradients and CTC backwards variables, please see the work of Graves et al.[39].

4.4 Methods

This section describes two ways to apply insights from CTC to address modular LfD. We first describe a naive adaptation which performs modular LfD with arbitrary differentiable architectures and discuss



(a) CTC - Computation of forward variables $\alpha_t(l)$ (b) TACO - Computation of forward variables $\alpha_t(l)$

Figure 4.2: Visualisation of the forward recursion for all paths ζ corresponding to the sketch τ. The horizontal axis denotes time, while the vertical denotes the task sequence. While a node at l,t in CTC is only weighted by the meta-controller probability $p(b_l|s_t,a_t)$, nodes and edges are weighted in TACO respectively via the sub-policies $\pi_l(a_t|s_t)$ and $\pi_l(a_{STOP}|s_t)$

its drawbacks. We then introduce TACO, which simultaneously optimises the alignment between trajectory ρ and task sequence τ and learns the underlying policies.

4.4.1 Naively Adapted CTC

A naive modular LfD algorithm can first align the state-action trajectories with the prescribed sketches via CTC to derive the required datasets $\mathscr{D}_k$ and then learn K modular policies with behavioural cloning as in (4.2), an approach we denote as CTC-BC.

However, this approach fundamentally differs from the regular application of CTC. At inference time, CTC is typically given a new trajectory ρ and computes the most likely sketch assignment τ. In contrast, we use CTC to align ρ with τ and subsequently train the sub-policies via BC while discarding the CTC based controller. This alignment for BC is derived via maximisation over $\alpha_t(l)$, as computed in (4.6), at each time-step, leading to the most likely path through the sequence. Following the determination of sketch-trajectory alignment, the sub-policies are optimised using (4.2).

While this approach only trains sub-policies via a single alignment based on the arg max of the forward variables α for every time-step in (4.6), we can account for the probabilistic assignment of active sub-policies by utilising a weighting based on the forward variables. However, as no aligned targets exist for the stop actions, they have to be derived based on the relations of the normalised α's between consecutive time-steps. A derivation of this α-weighted version of CTC-BC can be found in the Section 8.2. In our experiments, we consider the direct, single alignment resulting from taking the

arg max as well as the full optimisation of the joint probabilities via TACO.

A crucial drawback of CTC-BC is the independent computation of the optimisation for alignment and for imitation. The alignment affects the policy optimisation as it is performed in a later step but not vice-versa. The introduction of TACO in the next section addresses this shortcoming.

4.4.2 Temporal Alignment for Control (TACO)

Aligning the sequences ρ and τ via CTC treats the index for active sub-policies as pure symbols and fails to exploit the fact that we then need to learn the respective sub-policies via BC. Consequently, what CTC determines to be a good alignment might result in a badly conditioned optimisation problem for the sub-policies, converging to a local minimum and demonstrating degraded performance once deployed.

In this section, we propose *Temporal Alignment for Control* (TACO), in which the alignment is influenced by the performance of the sub-policies and addressed within a single optimisation procedure. Concretely, instead of maximising (4.3) followed by (4.2), we seek to maximise the joint log likelihood of the task sequence and the actions conditioned on the states:

$$p(\tau, \mathbf{a}_\rho | \mathbf{s}_\rho) = \sum_{\zeta \in \mathscr{Z}_{T,\tau}} p(\zeta | \mathbf{s}_\rho) \prod_{t=1}^{T} \pi_{\theta_{\zeta_t}}(a_t | s_t), \tag{4.8}$$

where $p(\zeta|\mathbf{s}_\rho)$ is the product of the stop, a_{STOP}, and non-stop, $\bar{a}_{STOP}$, probabilities associated with any given path. The first term in (4.8) is similar to the corresponding term in (4.5) but now depends only on states. Every possible alignment ζ dictates which data within the sequence ρ is associated with which sub-policy π_θ, which is the second term in (4.8) and corresponds to the BC objective. Maximising thus performs simultaneous alignment of τ and ρ and learns the associated policies for each sub-task.

As with CTC, the sketch length is expected to be shorter than the trajectory length, $L \ll T$, which for longer trajectories renders the computation of all paths ζ in $\mathscr{Z}_{T,\tau}$ intractable. However, the summation over paths can be performed via dynamic programming with a forward-backward procedure similar to that of CTC. Using consistent notation, the likelihood of a being at sub-task l at time t can be

formulated in terms of forward variables:

$$\alpha_t(l) \coloneqq \sum_{\zeta_{1:t} \in \mathscr{Z}_{t,\tau_{1:l}}} p(\zeta|\mathbf{s}_\rho) \prod_{t'=1}^{t} \pi_{\theta_{\zeta_{t'}}}(a_{t'}|s_{t'}). \tag{4.9}$$

Here $\tau_{1:l}$ denotes the part of the sub-task sequence until l. Even though not explicitly modelling the probability of a certain sub-task given the states and actions, $p(b|s,a)$, extending the policies with the stop action a_{STOP} enables switching from one sub-task in the sketch to the next. This allows computation of the probability of being in a certain sub-task b_l of the sketch, at time t. At $t = 1$ we know $\zeta_1 = \tau_1$, i.e., we always necessarily begin with the first sub-policy described in the sketch.

$$\alpha_1(l) = \begin{cases} \pi_{\theta b_1}(a_1|s_1), & \text{if } l = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{4.10}$$

Subsequently, a certain sub-policy can only be reached by staying in the same sub-policy or stopping the previous sub-policy in the sketch using the a_{STOP} action:

$$\begin{aligned} \alpha_t(l) = \pi_{\theta b_l}(a_t|s_t)[&\alpha_{t-1}(l-1)\pi_{\theta b_{l-1}}(a_{STOP}|s_t) \\ &+\alpha_{t-1}(l)(1-\pi_{\theta b_l}(a_{STOP}|s_t))]. \end{aligned} \tag{4.11}$$

Performing this recursion until T yields the forward variables. A visualisation of the recursion is shown in Figure 4.2b. The forward variables at the end of this recursion determine the likelihood in (4.8):

$$\alpha_T(L) = p(\tau, \mathbf{a}_\rho|\mathbf{s}_\rho). \tag{4.12}$$

Since the computation is fully differentiable, the backward variables and subsequently the gradient of the likelihood with respect to the parameters θ_i for each policy can be computed efficiently by any auto-diff framework. However, as the forward recursion can lead to underflow, we employ the forward variable normalisation technique from Graves et al.[39].

Intuitively, the probability for the stop actions of a policy at each time-step determines the weighting of data points (state-action pairs) for all sub-policies. If a sub-policy assigns low probability a specific data-point, e.g. if at similar states it has been optimised to fit different actions, the optimisation

increases the probability of the preceding and succeeding policy in the sketch for that data point, effectively influencing the probabilistic alignment.

A potential pitfall of simultaneously optimising for alignment and control is the early collapse of the alignment objective to a single path (Figure 4.2b). This stops further exposure of the sub-policies to potential state-action pairs they would be able to fit well. To achieve sufficient exploration of different possible alignments, we use dropout[153]. At every forward pass, different alignment paths are sampled, exposing the sub-policies to a wider range of data-points they could potentially fit, greatly improving performance.

4.5 Experiments

We evaluate TACO across four domains with different continuous and discrete states and actions including image-based control of a 3D robot arm. Our experiments aim to answer the following:

- How does TACO perform across a range of different tasks? Can it be successfully applied to a range of architectures and input-output representations?
- How does TACO perform with respect to zero-shot imitation on sequences not included in the training set and task sketches of different length?
- How does TACO perform in relation to baselines including CTC-BC (Section 4.4.1) and the fully supervised approach with all trajectories segmented into sub-tasks?
- How does the dataset size influence the relative performance of TACO in comparison with our baselines?

The latter questions are investigated by introducing a set of baselines based on Sections 4.3.2 and 4.4.1.

- GT-BC, which uses ground-truth, segmented demonstrations and performs direct maximum likelihood training to optimise the sub-policies (Eq. 4.2).
- CTC-BC, which uses both bidirectional *gated recurrent units*[19] CTC-BC(GRU) or *multi-layer perceptrons* CTC-BC(MLP) to perform CTC. In both cases, we apply MLPs for the sub-policies.

While the options framework presents a common approach to hierarchical LfD[34;74;58], it predomi-

nantly neglects the possibility of additional control information to switch between different high-level tasks, as is given by task sketches. Given the resulting limitation of independent modelling of different high level tasks, these framework cannot efficiently model multiple task sketches. This will result in highly degraded performance when evaluation is based on multiple different tasks, like the ones given in our evaluation (in both regular and in particular zero shot scenarios). For this reason, the evaluation focuses on approaches that utilise the task sketches.

The main evaluation metric is the task accuracy, i.e. the ratio of full tasks completed to the total attempted. In addition, the sub-task accuracy is ported to provide more insight. Finally, Table 8.3 provides results comparing the sequence alignment accuracy. That is, the percentage of time-steps that the demonstrated trajectory was given a correct sub-task label when compared to the ground truth alignment.

We focus on testing in a zero-shot setting: the task sequences prescribed at test time are not in the training data and are longer than the training sketches. Note that in the non-zero-shot setting, while the tasks to be completed have been seen, the world parameters such as feature positions are randomised. Finally, in all our evaluations we vary the size of the training set to investigate how performance varies with respect to available data.

4.5.1 Nav-World Domain

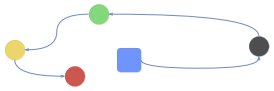


Figure 4.3: The Nav-World. The agent (Blue) receives a route as a task sketch. In this case. $\tau =$ (Black, Green, Yellow, Red)

We present the Nav-World domain, depicted in Figure 4.3, as a simple 2D navigation task. The agent (Blue) operates in a 8-dimensional continuous state space with four destination points (Green, Red, Yellow, Black). The state space represents the (x, y) distance from each of the destination points. The action space is 2-dimensional and represents a velocity (v_x, v_y). At training time, the agent is presented with state-action trajectories ρ from a controller that visits L destinations in a certain sequence given by the task sketch, e.g. $\tau =$ (Black, Green, Yellow, Red). At the end of learning, the agent outputs four sub-policies $\pi(a^+|s)$ for reaching each destination. At test time, it is given a sketch τ_{test} of length L_{test} containing a sequence of destinations. The task is considered successful if the agent visits

all destinations in the correct order. During demonstrations and testing, the agent's location and the destination points are sampled from a Gaussian distribution centred at predefined locations.

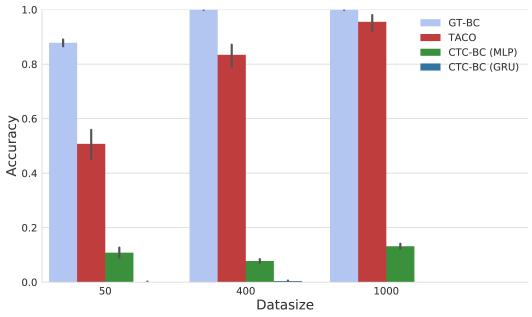


Figure 4.4: Nav-World results: Mean accuracy over 100 agents on 100 test tasks. Black error bars represent min/max achieved values. Task length at test time is $L_{test} = 4$ and at training time is $L = 3$. TACO (red) approaches the performance of a fully supervised sequence (grey) given enough data.

In this domain, the dataset sizes are 50, 400, and 1000 demonstrations. The task length at training time is $L = 3$. We report the task success rate for unseen, longer tasks of length $L_{test} = 4$ (zero-shot setting). 100 agents are trained for each task and each algorithm, with the evaluation based on 100 testing tasks.

As displayed in the results in Figure 4.4, CTC-BC performs poorly in both settings, with the MLP architecture performing slightly better. However, CTC (MLP) provides accurate alignment, often reaching 90% overlap between the predicted sub-task sequence and ground truth (Table 8.3). Fitting on smaller datasets however strongly reduces the quality of the policies. As the states are similar before and after a policy switch but the actions are different (different policies), small mistakes in alignment cause multi-modality in the data distribution. This causes relatively low performance for BC with a mean squared error (MSE) objective. In contrast, TACO avoids this problem by probabilistically weighting all policies when fitting to each time-step. From Table 8.3 we can see that TACO's inductive bias for control allows it to achieve much better alignment accuracy than CTC-BC. These two factors bring about performance that approaches that of GT-BC for larger datasets, a consistent trend in all our experiments.

4.5.2 Craft Domain

Andreas et al.[9] introduced the Craft Domain to demonstrate the value of weak supervision using policy sketches in the RL setting. In this domain, an agent is given hierarchical tasks and a sketch description of that task. The binary state space has 1076 dimensions and the action space enables

discrete motions as well as actions to pick up and use objects. A typical example task is $\tau_{\text{make planks}} =$ (get(wood), use(workbench)). The tasks vary from $L = 2$ to $L = 4$. The demonstrations are provided from a trained RL agent, which obtains near optimal performance. Performance is measured using the reward function defined in the original domain. We train agents for all baselines and deploy them on randomly sampled tasks from the same distribution seen during training.

Figure 4.5 shows the results, which are similar to those in Nav-World. CTC-BC fails to obtain substantial reward. However, unlike in Nav-World, CTC does not achieve good alignments with either architecture, as the abstract, binary state-action space makes it harder to detect distribution changes. Instead of concatenating state-action pairs, TACO learns a mapping from one to the other, making it easier to detect sub-policy changes. This allows TACO to match GT-BC with sufficient data.

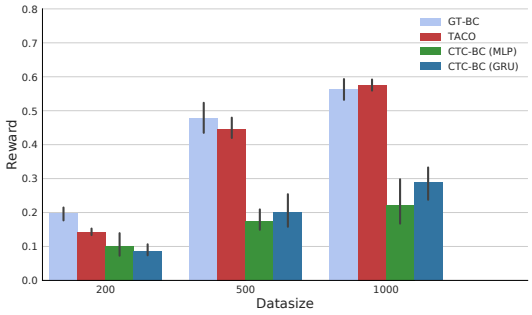


Figure 4.5: Craft results: Mean reward over 2000 tasks with 100 agents. Black error bars represent min/max achieved values between agents. The tasks are the same as during training but the environment instantiation is different. The RL agent used to derive the policies achieves a mean reward of 0.9. TACO (red) approaches the performance of a fully supervised method (grey) given enough data.

4.5.3 Dial Domain

Even though the Craft Domain is quite challenging, the lengths of the demonstrations and the resulting episodes are quite short, $T \leq 20$, making it easier to align ρ and τ. The final two experiments take place in a more realistic robotic manipulation domain. In the Dial Domain, a JACO 6 DoF manipulator simulated in MuJoCo[166] interacts with a large dial-pad, as shown in Figure 4.6. The demonstrations contain state-action trajectories that describe the process by which a PIN is pressed, e.g., $\tau = (0,5,1,6)$. A task is considered successful if all the digits in the PIN are pressed in order. Demonstrations in this domain come from a PID controller that can move to predefined joint angles. We sub-sample the data from the simulator by 20, resulting in trajectories $T \approx 200$ for $L = 4$. We report task and alignment accuracy as before. We consider two variants of the Dial Domain, one with joint-angle based states and the other with images. The action space represents the torques for each

joint of the JACO arm in both cases.

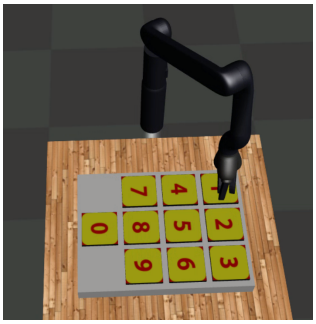


Figure 4.6: Dial Domain: A 6 DoF JACO arm must dial a PIN of arbitrary length.

Joint Space Dial Domain

In the first variant, the state is manually constructed and 39-dimensional, containing the joint angles and the distance from each digit on the dial pad in three dimensions. If the locations of the numbers in the dial-pad remains the same, however, the problem can be solved only using joint angles. For this reason, during each demonstration we randomly swap the location of the numbers. We test the policies on the standard number formation displayed in Figure 4.6, which is never observed during training.

Figure 4.7 shows the results, which follow the same trend as in the other domains. CTC-BC fails to complete any tasks and is not capable of aligning the sequences. TACO's performance significantly increases with data size, achieving superior min, mean, and max performances (on mean task accuracy) when compared to GT-BC at sizes of 1000 and 1200 demonstrations. We believe this is due to a regularising effect of TACO's optimisation procedure. To shed more light into this observation, we use a GT-BC policy and perform alignment with the TACO forward pass on 100 unseen trajectories (Table 8.3). The resulting alignment is lower than that of TACO, which suggests that GT-BC is more prone to overfitting.

To evaluate our methods in more complex zero-shot scenarios, we also measure the task accuracy over 100 tasks as L_{test} is increased, as shown in Figure 4.8. As expected, TACO's performance falls with increasing task length as the chance of failing at a single sub-task increases. The accuracy however decreases at a lower rate than that of the baseline.

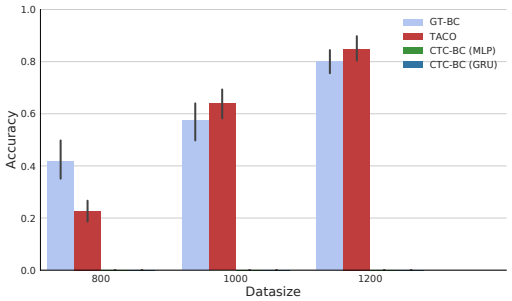


Figure 4.7: Joint Space Dial results: Mean task accuracy over 100 tasks of $L_{test} = 5$ with 100 agents. Black error bars represent min/max achieved values between agents. During training $L = 4$. Evaluation is performed in an unseen configuration of the dial pad. Bars for CTC based methods do not appear as they did not finish any tasks.

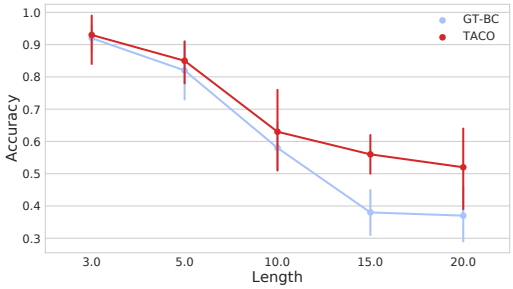


Figure 4.8: Mean task accuracy for increasing values of L_{test}. The accuracy is measured over 100 runs for TACO and GT-BC using 100 agents for each. Error bars represent min/max achieved values between agents.

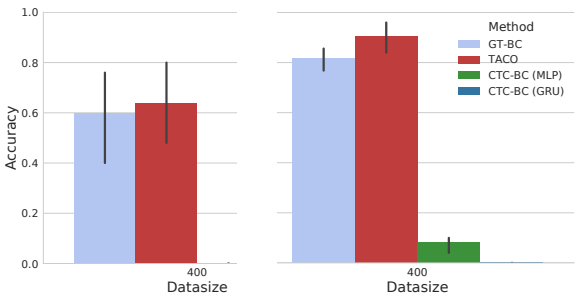
Image Space Dial Domain

The image-based variant of the Dial Domain considers the same task as above, but with an image-based state representation. We use RGB images of size 112×112, which are passed through a simple convolutional architecture before splitting into individual policies. In this case, we do not randomise the digit positions but discard joint angles from the agent's state space, as those angles would allow an agent to derive an optimal policy without utilising the image. Details of the architectures used can be found in Section 8.2.

Figure 4.9 details task and sub-task accuracy for this domain. We can see that TACO performs well in comparison with the baselines, despite the increased difficulty in the state representation.

4.6 Discussion

TACO requires only weak supervision for modular LfD while providing performance commensurate with fully supervised approaches on a range of tasks including zero-shot imitation scenarios.



(a) Task Accuracy (b) Sub-Task Accuracy

Figure 4.9: Image-Space Dial: Mean accuracy over 25 test tasks with 100 agents. Black error bars represent min/max achieved values between agents. CTC-BC is unable to perform full task sequences and only solves a minor percentage of the sub-tasks, while both GT-BC and TACO are able to complete most sub-tasks.

To successfully complete a task, a policy must not only complete the sub-tasks but also terminate the active policy at the right moment. Both GT-BC and CTC-BC rely on a single alignment between sub-tasks and demonstrations, based on ground truth and an arg max alignment respectively. TACO's strength lies in optimising the sub-policies over a distribution instead of a point estimate of the alignment. Training policies over a distribution of alignments exposes the sub-policies to more data points, which induces a regularising effect. This view is supported by the alignment accuracy results of Table 8.3 in which we evaluate trained policies for alignment on unseen sequences (based on (4.10) and (4.11)). For larger datasets GT-BC achieves less accurate alignment than TACO (Table 8.3), which in turn suggests that it may be more prone to overfitting. Better alignment on test demonstrations is correlated with task accuracy. This suggests that the idea of integrating the optimisation objectives in TACO for sequence alignment as well as imitation learning could have applications in cases where the end objective is good alignment rather than control policies.

TACO has been effectively applied to the presented tasks and significantly reduces supervision efforts. However, the given sketches are highly structured and dissimilar to natural human communication. An interesting avenue for future work is the combination of the increased modularity of TACO with more flexible architectures that can handle natural language[92;18]. In addition, future work aims at applications in more complex hierarchical tasks and on real robots.

While TACO reduces the annotation effort compared to temporally segmented trajectories, it relies on weak supervision via task sketches. Further work into relaxing the assumptions underlying the use of these sketches can aim at omitting the constraint on the order of the sub-tasks.

4.7 Conclusion

We presented TACO, a novel method to address modular learning from demonstration by incorporating weakly supervised information in the form of a task sketch, that provides a high-level description of sub-tasks in a demonstration trajectory. We evaluated TACO in four different domains consisting of continuous and discrete action and state spaces, including a domain with purely visual observations. With limited supervision, TACO performs commensurate to a fully supervised approach while significantly outperforming the straightforward adaptation of CTC for modular LfD in both control and alignment.

Chapter 5

Unsupervised Action Abstractions For Transfer Learning

Chapter 4 presents a weakly-supervised *domain adaptation* approach leveraging abstractions and a human-robot communication interface for generalisation across modular tasks. We continue with a *transfer learning* method, leveraging unsupervised abstractions, for modular task generalisation. Specifically, we leverage *action-abstractions* for compositional generalisation. We present a hierarchical KL-regularised[159] framework able to capture complex and generalisable mutli-modal behaviours. We introduce theory behind the transferability-expressivity of our abstractions, and the crucial role information asymmetry plays. Our unsupervised framework takes a step towards automated abstraction reuse in continual learning.

5.1 Introduction

While reinforcement learning (RL) algorithms recently achieved impressive feats across a range of domains[144;96;84], they remain sample inefficient[3;46] and are therefore of limited use for real-world robotics applications. Intelligent agents during their lifetime discover and reuse behaviours at multiple levels of abstraction to efficiently tackle new situations. For example, in manipulation domains, beneficial abstractions could include low-level motor primitives as well as higher-level object manipulation strategies. Endowing lifelong learning RL agents[108] with a similar ability could be vital

towards attaining comparable sample-efficiency.

To this end, two paradigms have recently been introduced. *KL-regularised* RL[159;36] presents an intuitive approach for automating behaviour reuse in multi-task learning. By regularising policy behaviour against a learnt task-agnostic prior, common behaviours across tasks are distilled into the prior, which encourages their reuse. Concurrently, *hierarchical* RL also enables behaviour discovery[175;94;56;45;177] by considering a two-level hierarchy in which the high-level policy is task-conditioned, whilst the low-level remains task-agnostic. The lower level of the hierarchy therefore also discovers behaviours that are transferable across tasks. Both hierarchy and priors offer their own behavioural (action) abstraction. However, when combined, one can in theory discover and leverage multiple action abstractions. Whilst prior works[162] have attempted this, they were unable to yield transfer benefits from a learnt prior.

In fact, successful transfer for both approaches critically depends on the correct choice of information asymmetry (IA). IA more generally refers to an asymmetric masking of information across architectural modules. This masking forces independence to, and ideally generalisation across, the masked dimensions[36]. Therefore, IA crucially biases learnt behaviour and how it transfers across environments. Previous works have motivated their chosen IAs primarily on intuition and have explored a narrow range of asymmetries[162;36;175], which, if sub-optimal, limits transfer benefits. We demonstrate that this indeed is the case for the hierarchical KL-regularised method in Tirumala et al.[162]. A more systematic, theoretically driven, approach to identifying IA schemes is therefore required in order to benefit from action-abstractions for transfer learning.

In this chapter, we employ hierarchical KL-regularised RL to effectively transfer behaviours across multiple abstraction levels. We begin by theoretically showing the crucial trade-off, controlled by choice of IA, that exists between the *expressivity* and *transferability* of action-abstractions across sequential tasks. With this insight, we consider a broader range of asymmetries than previously explored, across hierarchical levels, policy and prior, to successfully benefit from both priors and hierarchy. Our analysis provides concrete insights into which design choices are crucial for effective transfer. Further, we demonstrate that, while priors are significantly more effective than hierarchy, hierarchy is still important for enabling expressive priors. We apply our approach to a robot block-stacking, complex, sparse-reward domain unsolvable by the state-of-the-art baselines, and show that

with the right choice of asymmetries, sample-efficiency can be greatly improved.

5.2 Behaviour Transfer in Reinforcement Learning

This chapter considers multi-task reinforcement learning in partially observable Markov decision processes (POMDPs), defined by $M_k = (\mathscr{S}, \mathscr{X}, \mathscr{A}, r_k, p, p_k^0, \gamma)$, where tasks k are sampled from $p(\mathscr{K})$. $\mathscr{S}$, $\mathscr{A}$, and $\mathscr{X}$ denote observation, action, and observation-action history spaces. $p(\mathbf{x}'|\mathbf{x},\mathbf{a}) : \mathscr{X} \times \mathscr{X} \times \mathscr{A} \to \mathbb{R}_{\geq 0}$ represents the dynamics model. We denote the history of observations $\mathbf{s} \in \mathscr{S}$ and actions $\mathbf{a} \in \mathscr{A}$ up to time-step t as $\mathbf{x}_t = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_t)$. The reward functions $r_k : \mathscr{X} \times \mathscr{A} \times \mathscr{K} \to \mathbb{R}$ are history-, action- and task-dependent.

5.2.1 KL-Regularised Reinforcement Learning

The typical multi-task KL-regularised RL objective[165;67;125;137] takes the form:

$$\mathscr{O}(\pi, \pi_0) = \mathbb{E}_{\substack{\tau \sim \rho_\pi, \\ k \sim p(\mathscr{K})}} \left[\sum_{t=0}^{\infty} \gamma^t \left(r_k(\mathbf{x}_t, \mathbf{a}_t) - \alpha_0 \mathrm{D}_{\mathrm{KL}} \left(\pi(\mathbf{a}|\mathbf{x}_t, k) \parallel \pi_0(\mathbf{a}|\mathbf{x}_t) \right) \right) \right] \tag{5.1}$$

where γ is the discount factor and α_0 weighs the individual objective terms. π and π_0 denote the task-conditioned policy and task-agnostic prior respectively. The expectation is taken over tasks and trajectories τ from policy π ($\tau \sim \rho_\pi$) and initial state distribution $p_k^0(\mathbf{s}_0)$. When optimised with respect to π, this objective can be viewed as a trade-off between maximising rewards whilst remaining close to trajectories produced by π_0. When π_0 is learnt, it can be viewed as a method for learning shared behaviours present across tasks, and can thus bias multi-task exploration[159]. We consider the sequential learning paradigm, where action-abstractions are learnt from past (source) tasks, $p_{past}(\mathscr{K})$, and leveraged while attempting the current (target) set of tasks, $p_{current}(\mathscr{K})$.

5.2.2 Hierarchical KL-Regularised Reinforcement Learning

While KL-regularised RL has achieved success across various settings[3;159;115;45], recently Tirumala et al.[162] proposed a hierarchical extension where policy π and prior π_0 are augmented with latent variables, $\pi(\mathbf{a},\mathbf{z}|\mathbf{x},k) = \pi^H(\mathbf{z}|\mathbf{x},k)\pi^L(\mathbf{a}|\mathbf{z},\mathbf{x})$ and $\pi_0(\mathbf{a},\mathbf{z}|\mathbf{x}) = \pi_0^H(\mathbf{z}|\mathbf{x})\pi_0^L(\mathbf{a}|\mathbf{z},\mathbf{x})$, where subscripts H and L denote the higher and lower hierarchical levels. This structure encourages the shared low-level policy ($\pi^L = \pi_0^L$) to discover task-agnostic behavioural primitives, whilst the high-level discovers

higher-level behaviours relevant to each task. By not conditioning the high-level prior on task-id, Tirumala et al.[162] encourage the reuse of common high-level abstractions across tasks. Tirumala et al.[162] propose the following upper bound for approximating the KL-divergence between hierarchical policy and prior:

$$\mathrm{D_{KL}}\left(\pi(\mathbf{a}|\mathbf{x}) \parallel \pi_0(\mathbf{a}|\mathbf{x})\right) \leq \mathrm{D_{KL}}\left(\pi^H(\mathbf{z}|\mathbf{x}) \,\middle\|\, \pi_0^H(\mathbf{z}|\mathbf{x})\right) + \mathbb{E}_{\pi^H}\left[\mathrm{D_{KL}}\left(\pi^L(\mathbf{a}|\mathbf{x},\mathbf{z}) \,\middle\|\, \pi_0^L(\mathbf{a}|\mathbf{x},\mathbf{z})\right)\right] \quad (5.2)$$

where we omit task conditioning and explicitly declared shared modules to emphasise that this bound is agnostic to these choices. Whilst in principle this approach can be used to learn a high-level behavioural prior, π_0^H, in practice Tirumala et al.[162] do not observe benefits from learning it.

5.2.3 Information Asymmetry

Information Asymmetry (IA) is a key component in both of the aforementioned approaches, promoting the discovery of behaviours that generalise. IA can be understood as the masking of information accessible by certain modules. Not conditioning on specific environment aspects forces independence and generalisation across them[36]. In the context of hierarchical KL-regularised RL, the explored asymmetries between the high-level policy, π^H, and prior, π_0^H, have been narrow[162;115]. Tirumala et al.[162]; Pertsch et al.[115] explore latent priors of the form: $\pi_0^H(\mathbf{z}_t|\mathbf{x}_t) = N(\mathbf{z}_t|\mu(\mathbf{z}_{t-1}), \sigma^2(\mathbf{z}_{t-1}))$ and $\pi_0^H(\mathbf{z}_t|\mathbf{x}_t) = N(\mathbf{z}_t|\mu(\mathbf{s}_t), \sigma^2(\mathbf{s}_t))$ respectively. Both choices condition on minimal information, limiting their ability to distil and transfer knowledge across tasks. However, as discussed next, with a more principled approach to choice of IA, richer behaviours can be discovered and transferred.

5.3 Model Architecture and the Expressivity-Transferability Trade-Off

To rigorously investigate the contribution of priors, hierarchy, and information asymmetry for transfer learning, it is important to isolate each individual mechanism while enabling the recovery of previous models of interest. To this end, we present the unified architecture in Figure 5.1, which introduces *information gating functions* (IGFs) as a means of decoupling IA from architecture. Each component has its own IGF, depicted by coloured rectangles. Every module is fed all environment

information $\mathbf{x}_k = (\mathbf{x}, k)$ and distinctly chosen IGFs mask which part of the input each network has access to, thereby influencing which behaviours they learn. By presenting multiple priors, we enable a comparison with existing literature[162;115;61;47]. With the right masking, one can recover previously investigated asymmetries[162;115], explore additional ones, and also express purely hierarchical[175] and KL-regularised equivalents[36].

While prior works focused on the role of IA on policy (π) regularisation for multi-task learning[36], we focus on its influence over the prior's (π_0) ability to handle covariate shift across sequential tasks (transfer learning). In the sequential setting, there exists abrupt non-stationarities for task and agent trajectory distributions. As such, in this setting, it is particularly important that priors handle the non-stationarity and associated covariate shift. IA plays a crucial role here. Specifically, the choice of asymmetry between policy and prior affects the level of covariate shift encountered by the prior:

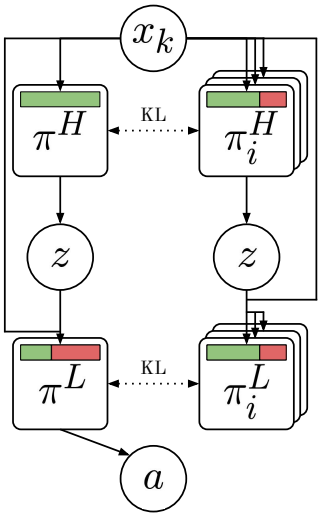


Figure 5.1: Hierarchical KL-regularised architecture. The hierarchical policy modules π^H and π^L are regularised against their corresponding prior modules π_i^H and π_i^L. The inputs to each module are filtered by an information gating function, depicted by coloured rectangles.

Theorem 5.3.1. *The more random variables a network depends on, the larger the covariate shift (input distributional shift, here represented by KL-divergence) encountered across sequential tasks. That is, for distributions p, q and inputs* $\mathbf{b}$, $\mathbf{c}$ *such that* $\mathbf{b} = (b_0, b_1, ..., b_n)$ *and* $\mathbf{c} \subset \mathbf{b}$*:*

$$\mathrm{D_{KL}}\left(p(\mathbf{b}) \parallel q(\mathbf{b})\right) \geq \mathrm{D_{KL}}\left(p(\mathbf{c}) \parallel q(\mathbf{c})\right).$$

Proof. See Section 8.3.2. □

In our case, p and q can be interpreted as training (source) and transfer (target) distributions over network inputs (such as π_0^H). Intuitively, Theorem 5.3.1 states that *the more variables you condition your network on, the less likely it will transfer* due to increased covariate shift encountered between training and transfer domains, thus promoting minimal information conditioning. However, the less information a prior is conditioned on, the less knowledge can be distilled and transferred:

Theorem 5.3.2. *The more random variables a network depends on, the greater its ability to distil knowledge in the expectation (output distributional shift between network and target distribution, here represented by the expected KL-divergence). That is, for target distribution p and network q with outputs a and possible inputs* $\mathbf{b}$*,* $\mathbf{c}$*,* $\mathbf{d}$*, such that* $\mathbf{b} = (b_0, b_1, ..., b_n)$ *,* $\mathbf{d} \subset \mathbf{c} \subset \mathbf{b}$ *,* $\mathbf{e} \in \mathbf{d} \oplus \mathbf{c}$*:*

$$\mathbb{E}_{q(\mathbf{e}|\mathbf{d})} \left[\mathrm{D}_{\mathrm{KL}} \left(p(a|\mathbf{b}) \parallel q(a|\mathbf{c}) \right) \right] \leq \mathrm{D}_{\mathrm{KL}} \left(p(a|\mathbf{b}) \parallel q(a|\mathbf{d}) \right).$$

Proof. See Section 8.3.2. □

In this particular instance, p and q could be interpreted as policy and prior distributions, π and π_0, respectively, a as action, $\mathbf{b}$ as history $\mathbf{x}$, and $\mathbf{c}$, $\mathbf{d}$, $\mathbf{e}$ as subsets of the history. Intuitively, Theorem 5.3.2 states *in the expectation, conditioning on more information improves knowledge distillation between policy and prior*. Therefore, IA leads to an impactful trade-off between the *transferability* and *expressivity* of discovered behaviours. Interestingly, covariate shift is upper-bounded by policy shift: $\mathrm{D}_{\mathrm{KL}} \left(p_{\pi_I}(\tau) \parallel p_{\pi_T}(\tau) \right) \geq \mathrm{D}_{\mathrm{KL}} \left(p_{\pi_I}(\tau_f) \parallel p_{\pi_T}(\tau_f) \right)$ (using Theorem 5.3.1), with $\tau_f = IGF(\tau)$ denoting IGF-filtered trajectories (i.e. network inputs), and π_I and π_T the training and transfer policies, respectively. It is therefore crucial, if possible, to minimise *both* behavioural and covariate shift across domains, if we wish to benefit from previously discovered behaviours.

While IAs influence *which* action-abstractions are discovered by hierarchy and priors, an important distinction must be made between *how* each approach *transfers* them. Hierarchy transfers knowledge as *hard constraints* on transfer behaviour, by enforcing the reuse of low-level action-abstractions (π^L). In contrast, priors transfer behaviours through *soft constraints* (via KL-regularisation), allowing the policy to deviate from them when necessary. As such, it is more crucial for IA between hierarchical levels to handle covariate shift (and hence generalise), than between policy and prior. This promotes reduced information conditioning for π^L than π_0. Thus, in-line with previous works[162;175], we share the low-level policy π^L between policy and prior, and condition only on state $\mathbf{s}_t$, ensuring minimal covariate shift and the discovery of instantaneous behaviours that generalise favourably. Unlike prior works, we consider conditioning the high-level prior, π_0^H, on additional information enabling richer behaviour discovery and transfer. Specifically, we explore temporal conditioning on varying levels of histories $\mathbf{x}_{t-i:t}$, where i denoting depth, thus enabling priors to capture reusable, sequential, high-level behaviours across tasks.

We additionally explore the importance of hierarchy for increased prior expressivity. Here, hierarchy enables a richer prior distribution, capturing multi-modal behaviours arguably present in many real-world examples. Importantly, while hierarchy increases expressivity, it does not influence covariate shift and harm transferability. See Section 8.3.4 for further architectural details.

5.4 Method

As described in Section 5.3, transfer learning poses distinct challenges, primarily abrupt covariate shifts. As such, we designed experiments to isolate each component's importance in this setting.

5.4.1 Training Regime

In Tirumala et al.[162,163], for each choice of IA, they first jointly train separate hierarchical policies and priors in a multi-task setting (over source domains), and then freeze the shared low-level policy and high-level prior when training on the transfer (target) task. Accordingly, they cannot isolate the effects of distinct IAs on transfer learning, as distinct IAs lead to distinct low-level policies when learning over the source tasks. To decouple the effects of IA and hierarchy for transfer learning, we propose the following regime. **Stage 1** trains a single hierarchical policy π in the multi-task setup whilst regularising against *multiple* high-level priors with distinct IGFs. Importantly, unlike Tirumala et al.[162], we stop gradient flow to policy π from the KL terms with learnt priors, enabling the multiple learnt priors to imitate the policy whilst not influencing it. In **stage 2**, we freeze the shared low-level policy and high-level prior and train on the target task. Refer to Algorithm 2 for an overview. Unlike Tirumala et al.[162,163], during stage 1 we train using variational behavioural cloning (BC) to bypass initial exploration issues. Given we focus solely on transfer learning, this choice is unimportant. Nevertheless, our method for learning action-abstractions from an expert can be considered of interest to the transfer learning community. During BC, we apply DAGGER[129], as per Algorithm 3, as this improves learning rates[79]. For further details refer to Section 8.3.4.

5.4.2 Variational Behavioural Cloning as KL-regularised Reinforcement Learning

Variational BC can be considered as KL-regularised RL in the absence of rewards, with the expert playing the role of the prior:

$$\mathscr{O}_{bc}(\pi,\pi_e) = -\mathbb{E}_{\substack{\tau\sim\rho_\pi,\\ k\sim p(\mathscr{K})}}\left[\sum_{t=0}^{\infty}\gamma^t\alpha_e \mathrm{D}_{\mathrm{KL}}\left(\pi(\mathbf{a}|\mathbf{x}_t,k)\parallel\pi_e(\mathbf{a}|\mathbf{x}_t)\right)\right] \tag{5.3}$$

More generally, Equations (5.1) and (5.3) can be extended to the setting of multiple priors:

$$\mathscr{O}_{bc}(\pi,\{\pi_i\}^{i\in I}) = \sum_{i\in I}\mathscr{O}_{bc}(\pi,\pi_i) \tag{5.4}$$

$$\mathscr{O}_{rl}(\pi,\{\pi_i\}^{i\subset I}) = E_\pi[R(\tau)] + O_{bc}(\pi,\{\pi_i\}^{i\subset I}) \tag{5.5}$$

For BC, $i\in\{0,1,...,N,u,e\}$, such that $\{\pi_i\}_{i\in\{0,...,N\}}$ denote the learnt priors, π_u the uniform prior, and π_e the expert prior. For RL, $i\in\{0,...,N,u\}$, as on the transfer tasks we do not have access to the expert policy. $E_\pi[R(\tau)]$ corresponds to the expected discounted return for policy π. Combining these objectives with Equation (5.2), we enable stage 1 and 2 training and transferring of hierarchical policies and priors. During transfer, task-conditioned policies, π^H, are not shared as they are task-specific. Uniform priors enable high-entropic policies, encouraging exploration and stabilising hierarchical learning[61]. We use Retrace[98] and double Q-learning[54] to train our critic. Refer to Sections 8.3.1 and 8.3.4 for full training details and a deeper discussion on variational BC and RL.

5.5 Experiments

Our experiments are designed to answer the following questions related to transfer learning over modular tasks: **(1)** Can we benefit from hierarchy and priors to effectively transfer knowledge over multiple levels of action-abstraction? **(2)** How important is the choice of IA between policy and prior, and does it lead to an expressivity-transferability trade-off of behaviours? **(3)** How important is hierarchy for enabling an expressive framework able to capture complex behaviours across tasks? **(4)** What are the relative contributions of priors, hierarchy, and IA for effective knowledge transfer?

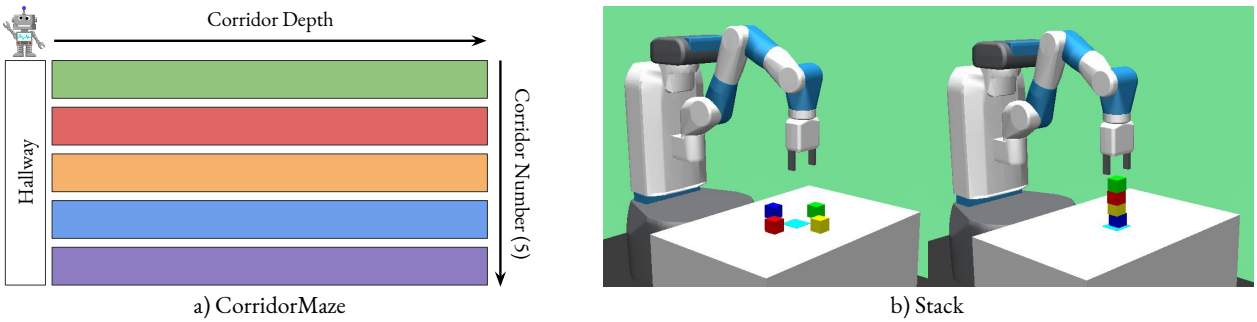


Figure 5.2: *Environments*. *a) CorridorMaze:* A hard-exploration toy domain. The agent starts in the hallway and must traverse a given number of corridors in a given sequence. The agent completes a given corridor by traversing to its depth and back. *b) Stack:* A hard exploration, robotic, domain. The agent must stack the cubes in a given ordering over the light blue target pad.

5.5.1 Environments

We explore these questions on two domains: a toy domain designed for controlled investigation of core agent capabilities and another, more challenging, robotics domain demonstrating the applicability of our approach in a practical setting (see Figure 5.2). Both of these domains exhibit modular behaviours at multiple levels of abstraction whose discovery would yield transfer benefits across modular tasks. Refer to Section 8.3.3 for full environmental setup details.

- **CorridorMaze.** The agent must traverse five corridors in a given ordering. We collect $4k$ trajectories from a scripted policy traversing any random ordering of two corridors, representing our source domains. During transfer, an inter- or extrapolated ordering must be traversed (number of sequential corridors $= \{2, 4\}$) allowing us to inspect the *generalisation ability* of distinct priors and how well they handle increasing levels of covariate shift. To see how *priors affect exploration*, we consider two sparsity levels: *semi-sparse*, rewarding per correct half-corridor traversal, and *sparse*, upon task completion. Our transfer tasks are *sparse 2 corridor* and *semi-sparse 4 corridor*.
- **Stack.** The agent must stack a subset of four blocks over a target pad in a given ordering. The blocks have distinct masses and only lighter blocks should be placed on heavier ones. Therefore, for this domain, the ability to discover long-term, sequential behaviours, namely priors corresponding to sequentially stacking blocks with respect to their masses, is beneficial. We collect $17.5k$ trajectories from a scripted policy, stacking any two blocks given this requirement, representing our source domains. To demonstrate the ability to achieve *generalisable transfer* and discover sequential coordination behavioural priors, the transfer task requires all blocks be stacked according their masses, heaviest first. Rewards are given per correct individual block stacked.

5.5.2 Hierarchy and Priors For Transfer Learning

Table 5.1: Comparing average return (a) and additional return (b), across 100 episodes. Reporting mean and standard deviation over 4 random seeds. Experiments that leverage prior experience were ran for 150k environment steps. The remainder, (Hier-)RecSAC, were ran for 1M steps.

(a) *Transfer Results*. There exists a transferability-expressivity trade-off for chosen IAs, with APES-H1 performing the best. Sparse reward domains are more influenced by IA.

Environment	CorridorMaze		Stack
Transfer type	interpolate	extrapolate	extrapolate
Reward type	sparse	semi-sparse	sparse
Transfer task	2 corridor	4 corridor	4 blocks
APES-H20	0.16 ± 0.09	3.84 ± 0.19	1.03 ± 0.45
APES-H10	0.22 ± 0.09	4.03 ± 0.55	1.22 ± 0.26
APES-H1	$\mathbf{0.80 \pm 0.03}$	$\mathbf{6.37 \pm 0.17}$	$\mathbf{3.11 \pm 0.12}$
APES-S	0.00 ± 0.00	3.03 ± 0.38	1.98 ± 0.16
APES-no_prior	0.00 ± 0.00	2.34 ± 0.19	0.00 ± 0.00
Hier-RecSAC	0.00 ± 0.00	0.07 ± 0.03	0.01 ± 0.01
RecSAC	0.00 ± 0.00	0.08 ± 0.02	0.01 ± 0.01
Expert	1	8	4

(b) *Covariate Shift Analysis*. In general, reduced IA benefits more from reduced covariate shift. Sparse domains suffer more from shift, seen by clearer IA covariate trends.

CorridorMaze		Stack
interpolate	extrapolate	extrapolate
sparse	semi-sparse	sparse
2 corridor	4 corridor	4 blocks
$\mathbf{0.28 \pm 0.03}$	0.12 ± 0.22	0.84 ± 0.06
0.24 ± 0.07	$\mathbf{0.62 \pm 0.30}$	$\mathbf{1.20 \pm 0.31}$
0.13 ± 0.02	0.34 ± 0.32	0.22 ± 0.24
0.00 ± 0.00	0.48 ± 0.21	0.00 ± 0.17
0.00 ± 0.00	0.09 ± 0.08	0.49 ± 0.26
0.00 ± 0.00	0.02 ± 0.03	0.00 ± 0.01
0.00 ± 0.00	0.27 ± 0.13	0.19 ± 0.05
0	0	0

We begin by exploring the relative importance of priors and hierarchy for transfer. The full setup, which leverages learnt high-level priors and hierarchy, is called *APES*. *APES-no_prior* represents a hierarchical model without a learnt prior, as in Tirumala et al.[162]. *RecSAC* represents a history-dependent *SAC*[48], trained directly on the transfer task and without prior experience. *Hier-RecSAC*, represents a hierarchical-equivalent of *RecSAC* using same hierarchical decomposition as *APES*, and is also trained directly on the task at hand, and is akin to the method proposed in Wulfmeier et al.[175]. Table 5.1a compares performance of all approaches on the transfer tasks. As expected, methods that leverage prior experience (*APES*, *APES-no_prior*) outperform those that do not (*Hier-RecSAC*, *RecSAC*), highlighting the importance of knowledge transfer. *APES-no_prior*, which only leverages hierarchy for knowledge transfer yields marginal benefits on these domains. *APES* strongly outperforms the rest of the methods, suggesting that the combination of hierarchy and priors is important for learning. Unsurprisingly, for the semi-sparse domains, the smaller the observed benefits, given that rewards are enough to guide learning. We provide further qualitative analysis of the performance of these methods in Section 5.5.6.

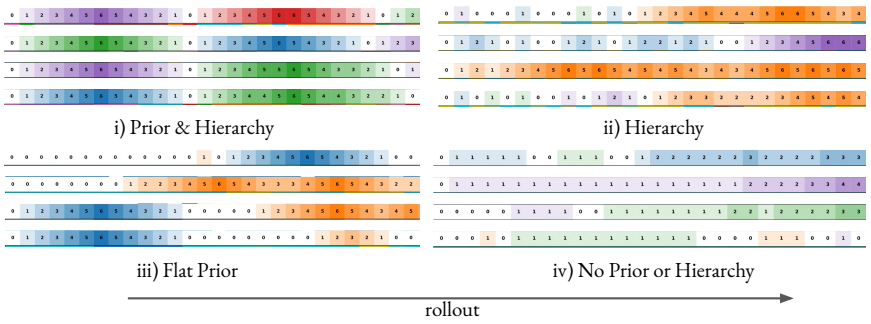


Figure 5.3: *CorridorMaze 2 corridor.* 4 rollouts shown for each method, with each episode rolled out horizontally. Corridors are colour coded and depth within them is denoted by shade; the darker the deeper. Hallway is white. See text for detailed analysis.

5.5.3 Information Asymmetry For Transfer Learning

To investigate the importance of IA for transfer, we compare high-level priors with access to increasing levels of asymmetry: *APES-{H20, H10, H1, S}* where *S* denotes a state-dependent prior without access to historical content and *Hi* denotes a history-dependent prior with history dependency $\mathbf{x}_{t-i:t}$. None of the priors are given access to task-dependent information, namely task id or exteroceptive information (non-egocentric data, e.g. cube position), ensuring their discovered behaviours transfer across these instances. The results in Table 5.1a demonstrate the role that IA has in achieving effective transfer across sequential tasks, with distinct choices leading to drastically different transfer performance. There exists a general trend where conditioning on too little (APES-S) or too much (APES-H20) information limits effective knowledge transfer. We also observe that the influence of IA is dependent on the reward type: for domains with semi-sparse rewards, transfer performance varies less as the rewards guide exploration to such an extent that effective knowledge transfer is unnecessary. We note, however, that in many practical tasks of interest, the rewards are sparse. Regardless of whether the transfer domain is interpolated or extrapolated, IA plays a important role in effective transfer, suggesting that *IA is important over a wide range of transfer tasks*.

5.5.4 Transferability-Expressivity Trade-Off

To understand the effects of IA on transfer, we refer back to Section 5.3, stating that reduced IA increases covariate shift and is upper-bounded by *behavioural shift* (shift in agent trajectories across tasks). Therefore, the smaller the behavioural shift, the smaller the covariate shift. In sequential learning, shift can occur due to three primary reasons:

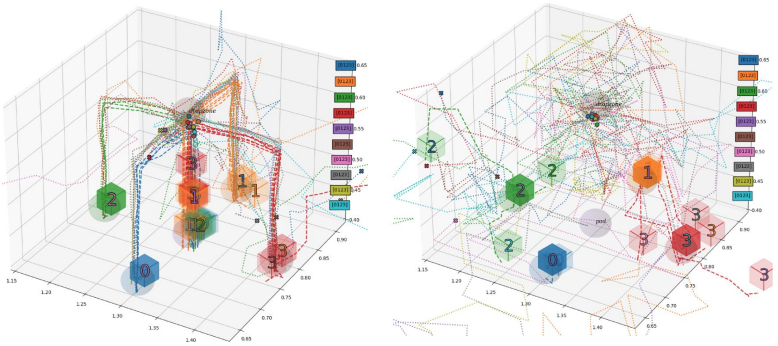
Table 5.2: Distillation Losses

Environment	CorridorMaze	Stack
APES-H1	0.22 ± 0.00	0.65 ± 0.00
APES-H10	0.12 ± 0.00	0.49 ± 0.00
APES-H20	$\mathbf{0.11 \pm 0.00}$	$\mathbf{0.47 \pm 0.00}$
APES-S	0.81 ± 0.00	0.75 ± 0.00
Max (min)	0.84 (0.00)	1.71 (0.00)

1) the solution to the task necessitates a shift, 2) any network components (e.g. the task-dependent high-level policy, π^H) are reinitialised, 3) during online training, an approximate critic incorrectly encourages sub-optimal, out-of-distribution, behaviour. Our setting, like many, is influenced by a combination of these.

We investigate whether the increased covariate shift negatively impacts transfer in practice with additional experiments focused on reducing behavioural shift, and hence the covariate shift, across sequential tasks. Table 5.1b presents the improvement in transfer performance obtained for experiments with an additionally pre-trained task-agnostic high-level policy $\pi^H(\mathbf{x})$ shared across the tasks (or flat policy for non-hierarchical equivalent). As such, no networks are reinitialised and initial behavioural shift across domains is minimised. These new experiments will reduce the covariate shift more for priors with less IA, as they inherently experience more shift (Theorem 5.3.1). We thus expect larger improvement in transfer performance for these priors. Table 5.1b confirms this trend demonstrating the *importance of prior covariate shift in transferability of behaviours*. Again, the trend is less apparent for the semi-sparse domain. Additionally, for the interpolated transfer task (*2 corridor*), the solution is entirely in the support of the training set of tasks. Naïvely, one would expect pre-training to fully recover lost performance and match the most performant method. However, as aforementioned, this is not the case as the critic, trained solely on the transfer task, quickly encourages sub-optimal out-of-distribution behaviours.

The previous experiments analysed the benefits that reduced IA has on positive transfer. However, as discussed in Section 5.3, conditioning on too little information limits the expressivity of the behaviours captured, as per Theorem 5.3.2. To observe whether this is the case, we compare distillation losses between prior and policy for various IAs. The results are shown in Table 5.2 (showing mean/standard deviation for 4 seeds). As per Theorem 5.3.2, *reduced IA reduces distillation losses* showing improved ability to distil behaviours present during training on the source domains. Tables 5.1a, 5.1b and 5.2 demonstrate the *transferability-expressivity trade-off* for distinct IAs, which is independent of transfer task and more important for sparse domains.



i) Prior & Hierarchy ii) Hierarchy

Figure 5.4: *Stack 4 blocks*. Rollouts with end-effector and cube movement depicted by dotted and dashed lines, colour-coded per rollout. Each rollout's end position represented by cross or cube respectively. Cube numbers correspond to their inherent ordering. See text for detailed analysis

5.5.5 Hierarchy for Expressivity

To investigate the role of hierarchy for effective task transfer, we compare the performance of the most performant hierarchical method, *APES-H1*, with its non-hierarchical equivalent, *APES-H1-flat*, which is otherwise equivalent to the full *APES* setup except with a flat rather than hierarchical prior. As discussed in Section 5.3, hierarchy increases the prior distribution expressivity, enabling the representation of multi-modal behaviours, whilst not increasing prior covariate shift and suffering from the associated transfer problems. When regularising against a flat prior, KL-regularisation must occur over the raw rather than latent action space. Therefore, to adequately compare hierarchical and non-hierarchical equivalents, we additionally compare against a hierarchical setup where regularisation occurs only over the action-space, *APES-H1-KL-a*.

Table 5.3: Hierarchy Ablation

Environment	CorridorMaze	
Reward type	sparse	semi-sparse
Transfer task	2 corridor	4 corridor
APES-H1	0.80 ± 0.03	6.37 ± 0.17
APES-H1-KL-a	0.76 ± 0.09	5.59 ± 0.17
APES-H1-flat	0.05 ± 0.035	4.52 ± 0.53
Expert	1	8

Comparing transfer results for ablations *APES-H1-KL-a* and *APES-H1-KL-flat*, in Table 5.3 (reporting mean and standard deviation across 4 random seeds), we see the *benefits that a hierarchical prior brings* to transferability in CorridorMaze domain. The flat prior is unable to solve the task, which on further inspection (see Section 5.5.6) is caused by its inability to capture multi-modal behaviours present in the source tasks. For bottleneck states[156], where multi-modality is needed the most, this leads to regularisation against a highly suboptimal unimodal action distribution. Interestingly, by comparing *APES-H1* and *APES-H1-KL-a*, we observe minimal benefits regularising against latent rather than raw actions, suggesting that with alternate methods for achieving multi-modal pri-

ors, hierarchy may not be necessary. Finally, comparing *APES-no_prior* with *RecSAC* , in Table 5.1a, we see hierarchy alone yields significantly less benefits for sparse-domain tasks than priors.

5.5.6 Exploration Analysis

To gain further understanding on the effects of hierarchy and priors, we visualise the policy rollouts early on ($5k$ steps) during transfer for APES, its ablations, and baselines. For *CorridorMaze* (Figure 5.3), the full setup using hierarchy and priors explores at the corridor level, traversing corridors in a random order across episodes. Hierarchy alone, unable to express preference over high-level abstractions, leads to temporally uncorrelated behaviours, and thus unable to explore at the corridor level. The flat prior, unable to represent multi-modal behaviours, leads to suboptimal exploration at the bottleneck state, the intersection of corridors, often leading the agent's position to remain static. Without priors nor hierarchy, exploration is further hindered, rarely traversing corridor depths. For *Stack* task (Figure 5.4), the full setup again explores at the individual block stacking level, alternating the ordering of blocks between episodes, but the prior leads to primarily stacking lighter upon heavier blocks. Hierarchy alone, explores undirectedly without the knowledge of what blocks to operate on, leading to temporally uncorrelated, inconsistent, high-frequency switching of low-level individual block manipulation behaviours.

5.6 Related Work

Hierarchical frameworks have a long history[156]. The options RL literature tackle the semi-MDP setup and explore the benefits that hierarchy and temporal-abstraction bring[100;177;61]. Wulfmeier et al.[175,177] use hierarchy to enforce knowledge transfer through shared hierarchical modules. However, for lifelong learning, where number of beneficial behaviours increase over time, it is unclear how well these approaches will fare, without priors to narrow exploration over them.

Priors have been used in various fields. In the context of offline-RL, Siegel et al.[141]; Wu et al.[172] primarily use priors to tackle value overestimation[81]. In the variational literature, priors have been used to guide latent-space learning[56;61;115;93]. Hausman et al.[56] learn episodic skills, limiting their ability to transfer. Igl et al.[61] learn options and priors, but is on-policy and therefore suffers from sample inefficiency, making the application to robotic domains challenging. In the multi-task litera-

ture, priors have been used to guide exploration[115;36;141;114;159], yet without hierarchy expressivity in learnt behaviours is limited. In the transfer learning literature, priors have also been used to bias exploration[115;6;146], yet either do not leverage hierarchy[115] or condition on minimal information[6;146], limiting expressivity. Recently, concurrent with our research, Tirumala et al.[163] were able to exploit hierarchical priors to transfer entire history-dependent high-level behaviours. However, their setup was in a high-data regime ($1e9$ data samples for pre-training), where covariate shift is less predominant. Unlike the aforementioned works, we consider the POMDP setting (on the source and transfer domains), arguably more suited for real-world robotics.

Whilst most previous works rely on information asymmetry, choice is often motivated by intuition. For example, Igl et al.[61]; Wulfmeier et al.[175,177] only employ task or goal asymmetry and Tirumala et al.[162]; Merel et al.[94]; Galashov et al.[36] use exteroceptive asymmetry. Salter et al.[133]; Galashov et al.[36] use asymmetry as a form of achieving robust and generalisable behaviours. Salter et al.[133] also investigate a way of learning asymmetry. We provide principled investigation on the role of asymmetry and point out its crucial role in transfer learning.

5.7 Conclusion

In this chapter, we employ hierarchical KL-regularised RL to efficiently transfer behaviours across multiple abstraction levels, showing the effectiveness of combining hierarchy and priors. We ablate over a broader range of temporal information asymmetries than previously explored, to effectively discover and transfer beneficial action-abstractions across sequential tasks, using variational BC as a sample efficient method for discovering these. We theoretically and empirically show the crucial trade-off, controlled by choice of IA, between the *expressivity* and *transferability* of any given abstraction across sequential tasks. Our experiments validate the importance of this trade-off for not only extrapolated domains, but also interpolated ones. We demonstrate that, while priors are significantly more effective than hierarchy, hierarchy is still important for enabling expressive priors. Finally, we apply our approach to a complex, sparse-reward robot block-stacking domain, unsolvable by the state-of-the-art baselines, showing that with the right choice of asymmetries, sample-efficient transfer can be achieved. We hope these insights will motivate future researchers to explore a wider range of asymmetries due to the potential performance benefits they may bring.

Chapter 6

Unsupervised Temporal Abstractions For Transfer Learning

Chapter 5 introduces a method that leverages *action-abstractions* for exploration over modular tasks during *transfer learning*. In this chapter, we develop a method leveraging abstractions for exploration **and** learning, crucial components for sample-efficient continual learning. We present an unsupervised *temporal-abstraction transfer learning* approach building on the options framework[155]. Specifically, our options terminate at bottleneck states, improving value estimation and exploration. To our knowledge, we introduce the first bottleneck-options framework meeting embodied continual learning requirements, namely off-policy learning over continuous state-action spaces.

6.1 Introduction

While reinforcement learning (RL) algorithms have recently achieved impressive feats across a range of domains[144;96;84], they remain sample-inefficient[3;46] which makes it challenging applying them to robotics applications. Natural embodied intelligence across their lifetime discover and reuse skills at varying levels of behavioural and temporal abstraction to efficiently tackle new situations. For example, in manipulation domains, beneficial abstractions could include low-level instantaneous motor primitives as well as higher-level object manipulation strategies. Endowing RL agents with a similar ability could be vital for attaining comparable sample efficiency[108].

The options framework[156] presents an appealing approach for discovering and reusing such abstractions, with options representing temporally abstract skills. We are interested in methods supporting sample-efficient embodied transfer learning[68], where skills learnt across prior tasks are used to improve performance on downstream ones. Of particular interest is the recent *Hindsight Off-Policy Options (HO2)*[177] framework supporting off-policy learning over continuous state-action spaces (crucial for sample reuse in embodied applications), training all options across every experience in hindsight (further boosting sample-efficiency). The latter is achieved by computationally-efficiently marginalising across all possible option-segmented trajectories during policy improvement, similar to TACO[140], leading to state-of-the-art results on transfer learning benchmarks. Whilst Wulfmeier et al.[177] discovered options online, we are interested in offline discovery, further encouraging sample-efficiency (discovering abstractions without any additional environment interactions).

Even though the options framework supports skill discovery, it does not constrain their behaviour. As such, vanilla approaches often lead to degenerate solutions[52], with options switching every time-step, maximising policy flexibility and return, at the expense of skill reuse across tasks. Therefore, constraining skill properties may be necessary. Many traditional methods[91;145;150;53;124] sought out *bottleneck options* that terminate at bottleneck states. Bottleneck states are defined as the most frequently visited states when considering the shortest distance between any two states in an MDP[150], and are considered beneficial for planning by inducing plans of minimum description length across a set of tasks[53;150]. Intuitively, bottleneck states connect different parts of an environment, and are hence visited more often by successful trajectories[53;91;154]. Unfortunately, these methods do not support off-policy hindsight learning, necessary for sample-efficiency, nor continuous state-action spaces, crucial in robotics.

To address this, we present *Model-Based Offline Options* (MO2), an offline hindsight *bottleneck options* framework supporting continuous state-action spaces, that discovers skills suited for planning (in the form of value estimation) and acting across modular tasks. We refer to modular tasks as a family of MDPs whose optimal policies are obtained by recomposing (a subset of) shared temporal behaviours. For example, for self-driving cars, traversing between junctions are temporal-abstractions shared between distinct navigation tasks. While this modular constraint appears restrictive, options can collapse to the original action-space, transitioning per time-step, if necessary. MO2 discovers

bottleneck options by extending HO2 to the offline context, training options to maximise: 1) the log-likelihood of the offline behaviours; 2) a *predictability objective* for option termination states across the episode, computed as the *cumulative 1-step option-transition log-likelihood*. Maximising the 1-step option-transition log-likelihood encourages low-entropic, predictable terminations. The cumulative equivalent (across all transitions), encourages minimal switching only at bottleneck states where behaviours diverge. Once options are learnt offline, we freeze them and perform online transfer learning over the option-induced semi-MDP, learning and acting over the temporally abstract skill space. We compare MO2 against state-of-the-art baselines on complex continuous control domains[35] and perform an extensive ablation study. We demonstrate that MO2's options are bottleneck aligned (see Figure 6.2) and improve acting (and exploring), value estimation, and learning of a jumpy option-transition model (defined in Section 6.3). On the challenging, sparse, long-horizon, AntMaze domain, MO2 drastically outperforms all baselines.

6.2 Preliminaries

Our work considers reinforcement learning in Markov decision processes (MDPs), defined by $M = (S, A, R, p, p^0, \gamma)$. S, A, R denote state, action and reward spaces. $p(s'|s,a) : S \times S \times A \to \mathbb{R}_{\geq 0}$ represents the dynamics model. $p^0(s) : S \to \mathbb{R}_{\geq 0}$ represents the initial state distribution. We denote the history of states $s \in S$ and actions $a \in A$ up to timestep t as $h_t = (s_0, a_0, s_1, a_1, \ldots, s_t)$. In this work, we consider option-augmented policies, taking the *call-and-return* formulation[156], defining options as triple $(I(s_t, o_t), \pi^L(a_t|s_t, o_t), \beta(s_t, o_t))$. I, β represent an option's initial and termination conditions and π^L denotes its behaviour. Following Bacon et al.[11]; Zhang and Whiteson[182], we define $I = 1 \forall s_t \in S$ and sample a new option o_t from $\pi^C(o_t|s_t)$ (the option-level controller) only if the previous one has terminated. $b(a|h)$ denotes the behavioural distribution that collects experiences.

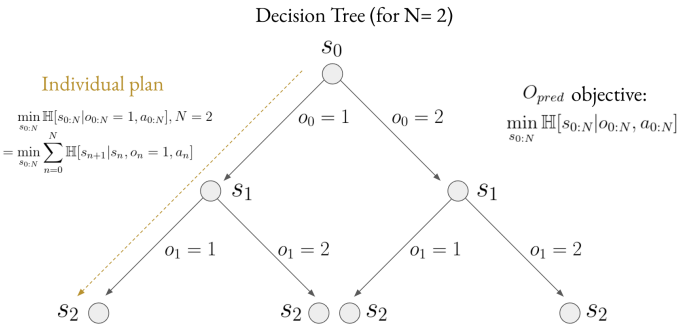


Figure 6.1: *Option Decision Tree (depth = 2)*: MO2's O_{pred} objective encourages low-entropic, compressed, option-level plans (over s_n) that reconstruct offline behaviours with high confidence. See Section 6.3.1 for more details.

6.2.1 HO2: Hindsight Off-Policy Options

We build on Hindsight Off-Policy Options (HO2)[177] as a highly sample-efficient off-policy actor-critic algorithm that builds on Smith et al.[149]. HO2 achieves higher sample efficiency than alternate options frameworks for two reasons: 1) HO2 trains all options in hindsight across all experiences within a trajectory, not just the current time-step. This is achieved by back-propagating gradients through HO2's dynamic programming inference procedure (discussed below), training all options end-to-end. 2) It uses Maximum A-Posteori Policy Optimisation (MPO)[3], a highly performant, variational, off-policy framework with monotonic improvement guarantees providing robustness.

To train all options across all experiences, HO2 unrolls the graphical model for option-based policies and calculates the joint $\pi(a_t, o_t|h_t)$, marginalising across all combinations of option sequences $o_{0:t-1}$. To avoid a combinatoral explosion, HO2 uses the following recursive relation, similar to the one introduced in Chapter 4:

$$p(o_t|s_t, o_{t-1}) = \begin{cases} 1-\beta(s_t, o_{t-1})(1-\pi^C(o_t|s_t)) & \text{if } o_t = o_{t-1} \\ \beta(s_t, o_{t-1})\pi^C(o_t|s_t) & \text{otherwise} \end{cases} \tag{6.1}$$

$$\tilde{\pi}^H(o_t|h_t) = \sum_{o_{t-1}=1}^{M} [p(o_t|s_t, o_{t-1})\pi^H(o_{t-1}|h_{t-1})\pi^L(a_{t-1}|s_{t-1}, o_{t-1})] \tag{6.2}$$

The distribution is normalized per timestep with $\pi^H(o_t|h_t) = \tilde{\pi}^H(o_t|h_t)/\sum_{o'_t=1}^{M} \tilde{\pi}^H(o'_t|h_t)$. π^H denotes option probabilities given histories and is not the same as π^C, the option controller for any given state. π^H is used to train all options across all experiences in hindsight, as aforementioned. Building on option probabilities we attain the joint, $\pi(a_t, o_t|h_t) = \pi^L(a_t|s_t, o_t)\pi^H(o_t|h_t)$. Refer to the original paper for more details.

6.3 MO2: Model-Based Offline Options

While HO2 provides a sample-efficient online framework for discovering options, no constraints are placed on their behaviour. As such, it is unclear whether HO2's abstractions are suited for acting, planning, and transfer across tasks. In this work, we focus instead on offline learning of options (over *source* domains) that are beneficial for online acting (and exploring) as well as learning (specifically

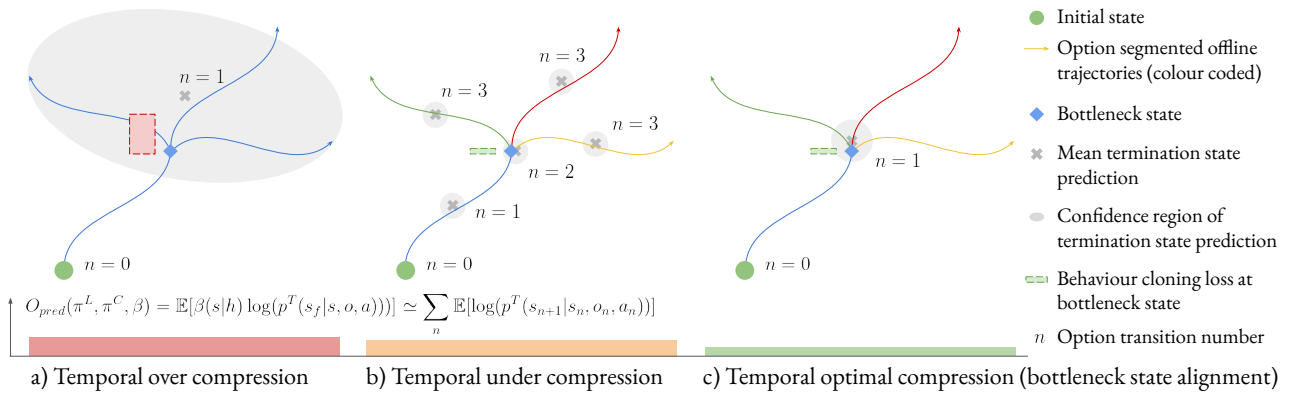


Figure 6.2: *Intuition behind MO2*. MO2 trains an option-based policy and option-transition model, $\tilde{p}^T(s_f|s,o,a)$, predicting terminal state s_f of option o given state s and action a. MO2's *cumulative 1-step option-transition log-likelihood*, O_{pred}, represents the predictability of option-transitions across the episode, $p^T(s_{n+1}|s_n,o_n,a_n)$, with n the option-transition number, s_{n+1}/s_n the n^{th} options termination/initiation states respectively. Individual likelihoods (any given n) encourage predictable, low-entropic transitions, minimising any termination confidence region (individual ellipse area), ruling out *a) temporal over compression*. Given positive transition entropies, the cumulative log-likelihoods (over all ns) lead to switching only when necessary, to reduce cumulative entropies (intuitively similar to the total area across ellipses), ruling out *b) temporal under compression*. MO2's *behaviour cloning loss* encourages terminations where multimodal behaviours exist. Combined, MO2 leads to *c) temporal optimal compression (bottleneck options)*, transitioning only at bottlenecks, low entropic, state distributions where behaviours diverge.

value estimation) on *transfer* domains. In line with prior works[91;145;150;53;124], we argue that options that terminate in bottleneck states are particularly beneficial.

Bottleneck states are generally considered as concentrated regions of the state-space that are highly visited when traversing distinct and diverse parts of the environment (such as doors in multi-room domains)[150;53]. Bottlenecks give rise to a compressed high-level decision problem, easy to solve due to the predictability of high-level option-transitions (given their concentrated nature) compressing the trajectory-level decision space (reasoning only over bottleneck transitions connecting distinct aspects of the environment)[150]. Prior literature has shown such states encode patterns of state visitations[91;154], centrality measures on state-transition graphs[150], and are beneficial for planning and acting across long horizons[53]. Unfortunately, existing methods do not support off-policy learning, necessary for sample-efficiency, nor continuous state-action spaces, crucial in robotics.

6.3.1 Discovering Bottleneck Options Offline

To this end, we introduce Model-Based Offline Options (MO2), an offline hindsight *bottleneck options* approach supporting continuous state-action spaces. MO2 extends HO2 with an additional learned

option-level transition model, and a *predictability* term encouraging option-level transitions across the episode to be predictable (in turn encouraging *bottleneck options* beneficial for planning[150]):

$$O_{pred} = \sum_{n=0}^{N} \mathbb{E}_{s_n,o_n,a_n,s_{n+1}\sim\pi}[\log(p^T(s_{n+1}|s_n,o_n,a_n))] = -\sum_{n=0}^{N} \mathbb{H}_\pi[s_{n+1}|s_n,o_n,a_n] = -\mathbb{H}_\pi[s_{0:N}|o_{0:N},a_{0:N}] \tag{6.3}$$

Here, n is the option-transition number within the episode, N the maximum, s_{n+1} and s_n are the n^{th} option's termination and initiation states respectively, and $p^T(s_f|s,o,a)$ is the option-transition distribution over the terminal state s_f of an option o, given state s and action a. Note that $n \neq t$ as options act at a different timescale. Shown in Equation (6.3) (for proof see Section 8.4.5), encouraging predictable option-transitions is equivalent to encouraging individual conditional entropies to be low $\mathbb{H}_\pi[s_{n+1}|s_n,o_n,a_n]$, ruling out the *temporal over compression* of offline behaviours (see Figure 6.2 for an explanation). Maximising O_{pred} is equivalent to minimising the joint conditional entropy of option-transition states $s_{0:N}$ given options $o_{0:N}$ and first option-executed actions $a_{0:N}$ from policy π, $\mathbb{H}_\pi[s_{0:N}|o_{0:N},a_{0:N}]$ (see Section 8.4.5 for proof and Figure 6.1 for a visual depiction). We subscript entropies by π to emphasise they are dependent on the policy. If individual transition entropies are positive (see Equation (8.25) how to achieve this), then O_{pred} encourages minimal option-transitions able to reconstruct offline behaviours using options, as to minimise N and the accumulation of positive entropies. As such, O_{pred} rules out the *temporal under compression* and leads to the *temporal optimal compression (bottleneck state alignment)* scenario in Figure 6.2. For a more detailed explanation why this objective discovers bottleneck options we refer the reader to Section 8.4.5.

Equation (6.3) requires evaluating the agent π in the environment to obtain s_{n+1}, s_n. As such, this objective does not support offline learning, as desired. Additionally, inline with HO2, we wish to train all options across all experiences in hindsight, as to maximise sample-efficiency. Therefore, we present an alternate form for O_{pred} supporting offline hindsight learning, unrolling the graphical model for option-based policies and training across a distribution of option-transitions given histories (by efficiently marginalising over option-transitions $o_{0:t-1}$ using Equations (6.1) and (6.2)):

$$O_{pred} = \mathbb{E}_{\substack{s_t,h_t\sim D \\ o\sim\pi^H(\cdot|h_t),a\sim\pi^L(\cdot|s_t,o) \\ s_f\sim p^T(\cdot|s_t,o,a)}} [\beta(s_t|h_t)\log(p^T(s_f|s_t,o,a))] \tag{6.4}$$

With $\beta(s_t|h_t) = \sum_{o_{t-1}=1}^{M} \beta(s_t,o_{t-1})\pi^H(o_{t-1}|h_t)$, representing the probability that state s_t is terminal,

for the previous option o_{t-1}, and initial for the subsequent option o_t (denoted as o in Equation (6.4)), given history h_t. $\pi^H(o_{t-1}|h_t) = \pi^H(o_{t-1}|h_{t-1})\pi^L(a_{t-1}|o_{t-1},s_{t-1})/\pi(a_{t-1}|h_{t-1})$ represents the distribution over o_{t-1} given history. D represents the offline data created by behavioural policy $b(a|h)$. Options are sampled from $\pi^H(o_t|h_t)$ as the offline data does not contain option labels. This form of O_{pred} uses β as a weighting for how probable s_t is a starting state (s_n) of an option. The expectation is now taken over offline trajectories, with β representing (proportionally) the summation over option-transitions, akin to the original O_{pred}. We use p^T to sample terminal states s_f (or s_{n+1}) in Equation (6.4). In practice, we do not have access to p^T and learn it offline instead (see Section 6.3.2). We refer the reader to Section 8.4.5 for a detailed description under what conditions both Equations (6.3) and (6.4) are equivalent from an optimisation perspective. Our overall objective combines O_{pred} with a behavioural cloning, maximum likelihood, objective ensuring we distil offline behaviours, $O_{bc} = \mathbb{E}_\pi[\log(\pi(a,o|h))]$:

$$\max_{\pi^L,\pi^C,\beta} O_{mo2}(\pi^L,\pi^C,\beta) = \mathbb{E}_{\substack{s_t,a_t,h_t\sim D \\ o\sim\pi^H(\cdot|h_t),a\sim\pi^L(\cdot|s_t,o) \\ s_f\sim p^T(\cdot|s_t,o,a)}} [\beta(s_t|h_t)\log(p^T(s_f|s_t,o,a)) + \log(\pi(a_t,o|h_t))] \quad (6.5)$$

While $p^T(s_f|s_t,o,a)$ encourages terminations predictable across all encountered states s_t, the additional weighting $\beta(s_t|h_t)$ focuses predictability only over the distribution of (rather than sampled) option initiation states given histories. In line with the intuition from Figure 6.2, O_{pred} and O_{bc} together encourage offline hindsight *bottleneck options* discovery over continuous state-action spaces.

6.3.2 Learning the Option-Transition Model Offline

MO2 assumes access to the option-level transition distribution $p^T(s_f|s_t,o,a)$ in Equation (6.5). Given that we do not have access to such a model, we instead learn to approximate it, $\tilde{p}^T(s_f|s_t,o,a)$, using a two-step self-supervision process. We first sample options according to $\pi(o_t|h_t)$ and then sample terminations according to $p^T(s_f|s_{t:T},a_{t:T},o_t)$, the termination distribution of the option given future states and actions. Specifically, we sample s_f by sampling termination condition $\beta(s_k|o_t)$ over all future timesteps, $k \in [t,T]$, with T representing episodic length. Each time the termination sample is

true, the corresponding state is labelled terminal, $s_f = s_k$.

$$\max_{\tilde{p}^T} O_{tran}(\tilde{p}^T) = \mathbb{E}_{\substack{s_{0:T}, a_{0:T}, h_t \sim D \\ o \sim \pi(\cdot|h_t) \\ s_f \sim p^T(\cdot|s_{t:T}, a_{t:T}, o_t)}} [\log(\tilde{p}^T(s_f|s_t, o, a_t))] \tag{6.6}$$

As such, we learn an option-transition model that predicts the termination state distribution of an option for any state during its execution (not just the initial state). We note that this objective is biased, as the future states $s_{t:T}$ and actions $a_{t:T}$ used to sample s_f are not from the option-policy π over which we predict option-transitions, but the behavioural policy b that created the offline behaviours. However, over time, as $\pi(s_{t:T}, a_{t:T}|h_t)$ aligns with $b(s_{t:T}, a_{t:T}|h_t)$, as encouraged by Equation (6.5), this bias tends to zero. We train $\tilde{p}^T(s_f|s_t, o, a)$ using Equation (6.6), simultaneously using it to train $\pi(a, o|h)$ (Equation (6.5)). During transfer, we use these options to act, explore, and aid value estimation.

6.3.3 Online Transfer Learning

Once options are learnt offline (over *source* domains), there are several ways one can use them to improve online learning on a *transfer* domain. We make the assumption that the *source* and *transfer* MDPs are *modular* and support compositional generalisation (there exists shared temporal abstractions between training and transfer domains that can be recomposed to obtain optimal performance on both). As such, during transfer, MO2's options are frozen and we train a new high-level categorical controller, $\pi^C(o_t|s_t)$, to recompose them, using MPO (see Section 8.4.3). If options do not suffice and require fine-tuning, one would need to tackle catastrophic forgetting of skills (see Kirkpatrick et al.[69]). We leave this as future work. To discover shared abstractions across *modular* MDPs, we need access to optimal policies for *multiple source* MDPs. This is the main assumption about the offline data, that it is multi-task and representative enough of the family of *modular* MDPs of interest, such that we can discover all the relevant *modular* abstractions. While the *modular* constraint may appear restrictive, the increasingly diverse the *source* MDPs are (as expected during lifelong learning[68]), the increasingly probable that the optimal *transfer* policy can be obtained by recomposing a subset of the *source* abstractions. MO2's temporal abstractions afford two benefits: 1) temporally-correlated actions and exploration (acting at the bottleneck state level); 2) improved credit assignment and value estimation over the option induced semi-MDP (with a lower temporal granularity than the original MDP and occurring over concentrated regions of the state-space beneficial for planning[150]).

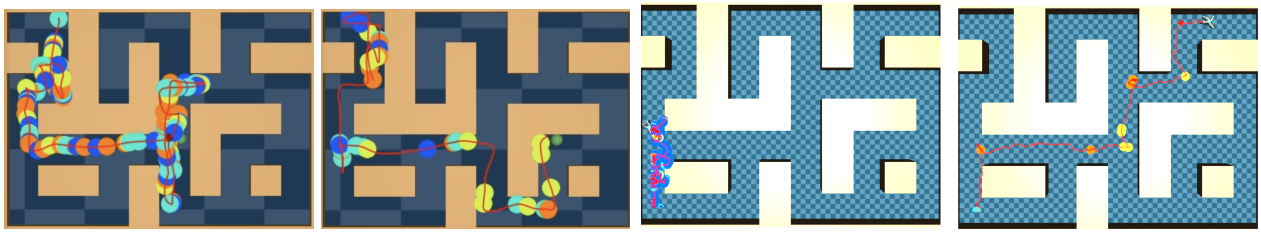


Figure 6.3: *Policy Rollouts*. We plot policy rollouts across 1 episode for Maze2D (left two plots) and AntMaze (right two plots), for HO2-lim (offline) (inner-left for each domain) and MO2 (inner-right for each domain). These rollouts are obtained early during training on the transfer domain. The centre of mass trajectory is plotted as a red line. The termination locations of options are denoted as coloured circles along the rollout, with the colour denoting the subsequent chosen option. MO2 leads to more temporally compressed options, switching less frequently, and primarily at environment-aligned, bottleneck states, corresponding with corridor intersections. As such, MO2 leads to temporally correlated behaviour, traversing the depths of the maze. In contrast, HO2-lim (offline) exhibits high frequency switching with shallow maze traversal for the long horizon, high dimensional, AntMaze domain.

6.4 Experimental Setup

We explore how beneficial MO2's behavioural abstractions are for: 1) acting and exploration; 2) value estimation; 3) learning an option-transition model; 4) discovering compressed abstractions terminating at bottleneck states. We compare our results against two offline versions of HO2 trained solely to maximise O_{bc} and not on O_{pred}. We call these baselines HO2 (offline) and HO2-lim (offline), a variant introduced in Wulfmeier et al.[177] that encourages minimal option switching by penalising switches (yet does not encourage predictable terminations). We compare against these baselines to inspect how important option predictability is for all aforementioned questions. We compare with HO2 as we build on it as a state-of-the-art, sample-efficient, options framework supporting continuous state-action spaces. Finally, we compare against HO2 trained from scratch on the transfer task (HO2 scratch) to quantify skill transfer importance. See Section 8.4 for full experimental details.

6.4.1 Semi-MDP vs MDP Ablations

To investigate the relative importance of MO2's options for both exploration and value estimation, we run two variants of each method during transfer: 1) value estimation occurs over the semi-MDP; 2) value estimation occurs over the MDP (akin to the original HO2 that we build on[177]). By contrasting these experiments we can infer the importance of MO2's temporal-abstractions, in relation to others, for credit-assignment. Comparing against the HO2 baselines, we can evaluate the importance of

MO2's options for exploration. For all methods with pre-trained options, we act at the option-level during transfer. See Section 8.4.3 for further ablation information.

6.4.2 Domains

We evaluate on two D4RL suite domains[35]: Maze2D and AntMaze (see Figure 6.3 and Section 8.4.4 for details). We choose these domains as the *source* and *target* domains are modular in structure, meaning there exists shared temporal-abstractions (individual corridor traversals) able to maximise performance across both (distinct corridor orderings). The *source* domains are MDPs whose expert policies produce the offline data over which we discover skills. The *target* domain refers to the MDP that we perform *transfer learning* over, bootstrapping off the skills from the *source* domains. These domains have easy-to-interpret bottlenecks (corridor intersections, see Figure 6.3) helping with analysis. The source domains correspond to random ordering of corridor traversals (see Fu et al.[35] for how the expert data was collected). During transfer, the agent much reach an unknown maze location, traversing a distinct ordering of corridors (reordering the modular abstractions). Both *target* domains have sparse rewards (rewarded upon goal reaching) and long-horizons, thus benefiting from abstractions for exploration and credit-assignment.

We compare AntMaze and Maze2D as both have distinct state and action dimensionalities, and episodic-lengths, allowing us to see how well MO2 scales to increasingly hard exploration domains. Maze2D acts as an easier domain with lower dimensional state-action spaces, where the agent controls a pointmass and during transfer must reach a goal on the other end of the maze. AntMaze is a more ambitious, 3D example, with a higher dimensional state-action space necessitating more directed behaviours and exploration to discover the sparse rewards (as the agent needs to learn to coordinate the ant before being able to navigate it). Furthermore, for AntMaze, credit-assignment is increasingly difficult taking 900 steps, as opposed to 200 for Maze2D, to reach the goal.

6.5 Results

In this section, we perform a qualitative and quantitative analysis of the options belonging to MO2 and the baselines, and in the following sections answer the questions from Section 6.4.

6.5.1 Temporal Compression Of Offline Behaviours (Bottleneck Options)

Table 6.1: Temporal Compression Metrics

(a) Average Switch Rate

Environment	Maze2D	AntMaze
HO2 (offline)	0.48 ± 0.04	0.56 ± 0.04
HO2-lim (offline)	0.15 ± 0.01	0.36 ± 0.01
MO2	$\mathbf{0.03 \pm 0.00}$	$\mathbf{0.01 \pm 0.00}$
Max (min)	1.00 (0.00)	1.00 (0.00)

(b) Expected Behaviour Cloning Performance

Environment	Maze2D	AntMaze
HO2 (offline)	$\mathbf{0.102 \pm 0.004}$	$\mathbf{1.30 \pm 0.05}$
HO2-lim (offline)	0.082 ± 0.005	0.98 ± 0.02
MO2	0.079 ± 0.001	$\mathbf{1.27 \pm 0.04}$
Max (min)	0.102 (-5.000)	1.30 (-50.00)

We analyse the properties of the options for each method. In Figure 6.3, we visualise representative rollouts of the transfer agent, early during training, using the pre-trained, frozen, options. We plot rollouts for MO2 and HO2-lim (offline). We plot centre-of-mass rollouts for each domain, depicted as a red line, together with option termination locations, depicted as circles colour coded by the subsequent chosen option.

For both domains, we observe that MO2's options primarily align with individual corridor traversal with terminations occurring at the intersection of corridors. MO2's options have therefore discovered the environment bottleneck states (corridor intersections) leading to switching of options at a low-entropic distribution of states that are highly visited and predictable, where behaviours naturally diverge, connecting distinct regions of the environment (individual corridors). This leads to high temporal compression of offline behaviours, with option switches occurring infrequently.

In contrast, HO2-lim (offline) exhibits far lower temporal compression, with option switching occurring more frequently, and at seemingly random locations, not aligning with bottleneck states. This suggests that penalising option switches, as achieved through HO2-lim (offline), is not enough to achieve effective temporal compression and bottleneck discovery. Comparing average switch rates (the frequency of option switching over the offline trajectories $= 1/$ expected option duration), seen in Table 6.1a, additionally shows the compression disparity between each approach. Importantly, MO2's increased temporal compression barely influences controllability, as seen by comparing behaviour cloning O_{bc} values in Table 6.1b (taken over offline trajectories, the higher the score the better). In Table 6.1b, max (min) scores refer to the range of values achieved across all methods during training. We report mean $\pm$ standard deviations, obtained across four random seeds.

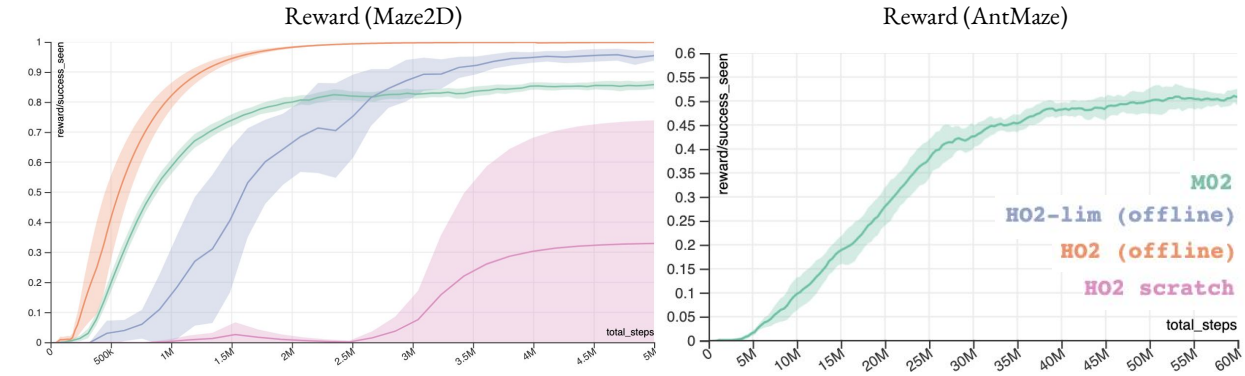


Figure 6.4: *Return Curves*. We plot return learning curves for both transfer environments and all methods. Experiments were ran with 4 seeds, and we plot mean (solid line) and 1 standard deviation (shaded region). Curves are periodically obtained by evaluating the current policy on the environment. Temporally compressed, bottleneck aligned, abstractions are essential for the sparse, long horizon, AntMaze, with MO2 the only method with non zero reward. For the less sparse, Maze2D, domain, MO2's abstractions are less necessary and hinder asymptotic performance.

6.5.2 Exploration

In Figure 6.4, we plot transfer performance for each approach on both domains. For the sparser, longer horizon, higher-dimensional action-space AntMaze domain, we observe that MO2 is the only method able to attain any rewards and solve the task, demonstrating the benefits of MO2's temporally compressed bottleneck options for exploration. In contrast, for Maze2D, temporally compressed behaviours, achieved either by MO2 or HO2-lim (offline), do not yield benefits, demonstrating that for simpler exploration domains, temporal compression is not necessary for effective transfer, and can slightly hinder policy flexibility and asymptotic performance.

Comparing initial policy rollouts in Figure 6.3, we see that MO2's reduced switch rate leads to more directed, temporally correlated, exploration, with wider coverage of the maze for AntMaze. This is not the case for Maze2D, suggesting that for low-dimensional environments with short horizons, compression is not necessary for exploration.

6.5.3 Value Estimation

To evaluate the importance of temporal-abstractions for learning, specifically value estimation, we plot semi-MDP vs MDP policy evaluation learning curves, where value estimation occurs either over the semi-MDP or MDP, both acting at the option-level[4] (see Figure 6.5). We additionally compare

[4]The Reward and Value Estimates horizontal axes are aligned despite different metrics (total environment steps vs number of network updates). The metrics differ due to the asynchronous learner.

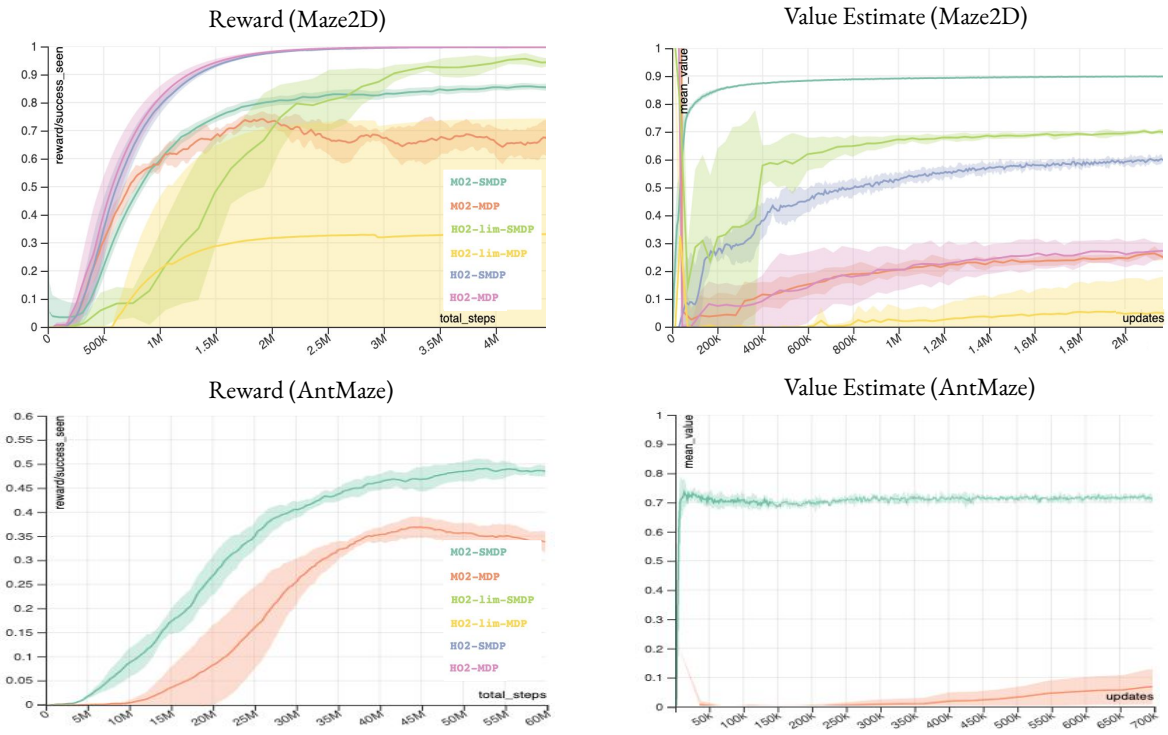


Figure 6.5: *Semi-MDP vs MDP Learning Curves*. To evaluate the importance of temporal-abstraction for policy evaluation and improvement, we compare expected return and value estimates between methods that perform value estimation over the semi-MDP vs MDP, but both acting at the option-level. We plot mean (solid line) and standard deviation (shaded region) across 4 seeds. Curves are periodically obtained by evaluating the current policy on the environment. As seen by comparing semi-MDP and MDP experiments, contrasting reward and value estimate curves, abstractions reduce value bias, improve value convergence, and policy performance. The degree of improvement directly correlates to the level of temporal compression. The more compressed (MO2 $>$ HO2-lim $>$ HO2) the greater the improvement between semi-MDP and MDP experiments. Nevertheless, for the simpler Maze2D domain, temporal-abstraction is unnecessary with HO2-MDP performing best.

policy performance for both setups, to inspect how important an accurate critic is for policy improvement (see Figure 6.5). Comparing policy performance and evaluation for each method, it is apparent that value estimation over MO2's more temporally compressed options yields a faster learning, less biased critic (seen when comparing return and value estimate curves). For both domains, improved policy evaluation yields improved policy performance, although the improvement is more significant for the longer horizon AntMaze. Additionally, the more temporally compressed, the larger the improvement for value bias and convergence as well as policy improvement, when value estimation occurs over the semi-MDP. This highlights the importance of temporal compression for policy evaluation and improvement. Interestingly, for Maze2D, an increasingly biased critic, HO2-MDP, does not hinder performance, suggesting that here, reasonable advantage estimation is occurring.

6.5.4 Option-Transition Predictability

In Section 6.3.1, we proposed discovering *bottleneck options* by optimising $O_{pred} = \sum_n \mathbb{E}_\pi[\log(p^T(s_{n+1}|s_n, o_n, a_n))]$ (Equation (6.5)), predictable option-transitions across offline episodes (or predictable state-transitions over the option-induced semi-MDP). This objective is similar to the motivation behind bottleneck abstractions in the neuroscience literature[150], as abstractions beneficial for planning across tasks. In our case, the tasks are the *source* MDPs whose expert policies created the offline dataset. In order to plan effectively over the temporal-abstractions, the state-transitions over the semi-MDP should be predictable (as to minimise planning errors). O_{pred} optimises for such behaviour. As outlined in Section 6.3.2, we use a learnt option-transition model $\tilde{p}^T$ to optimise O_{pred}, as we do not have access to the true p^T. We train such a model using Equation (6.6). To validate that Equations (6.5) and (6.6) lead to predictable semi-MDP transitions on the *source* domains, despite using a learnt transition model to optimise behaviours, we evaluate the following:

$$CE(\tilde{p}^T) = -\sum_{n=0}^{N} \mathbb{E}_{s_n, a_n, o_n, s_{n+1} \sim \pi}[\log(\tilde{p}^T(s_{n+1}|s_n, o_n, a_n)))] \tag{6.7}$$

The expectation is taken over agent π rollouts in the environment after skills are frozen (as oppposed to over the offline dataset as in Equation (6.5)), but before training on the *transfer* domain. As such, we evaluate predictable semi-MDP transitions according to an agent optimised over the *source* domains, as desired. n corresponds to option transition number with N the final transition in the episode, o_n is the n^{th} sampled option, s_n its initiation state, s_{n+1} its terminal state, a_n the first action taken by the option during its execution. $CE(\tilde{p}^T)$ represents how well our model $\tilde{p}^T$ can predict transitions across the semi-MDPs. This metric is influenced by both the individual transition predictability and the temporal resolution of the transitions as they influence the total episodic number of transitions.

We report CE results in Table 6.2. The lower the score the more predictable the transitions. In Table 6.2, the max value refers to the initial value achieved before offline training of π and $\tilde{p}^T$ and is shown to contextualise the results. Min refers to minimum value achieved across all methods. We report mean $\pm$ standard deviations, obtained across

Table 6.2: $CE(\tilde{p}^T)$ (see text)

Environment	Maze2D	AntMaze
HO2 (offline)	$\mathbf{0.45 \pm 0.00}$	7.20 ± 0.01
HO2-lim (offline)	1.00 ± 0.02	13.00 ± 1.00
MO2	0.70 ± 0.05	$\mathbf{4.00 \pm 0.05}$
Max (min)	5.00 (0.45)	50.00 (4.00)

four random seeds. Interestingly, for AntMaze, with high dimensional state-action spaces and a long horizon (episodic length), MO2 leads to the best predictability scores, thanks to its low switch rate transitioning at predictable bottleneck states (as seen in Figure 6.3). For the simpler, low dimensional domain with short horizon, Maze2D, MO2's temporal-abstractions are not necessary for high predictability (and slightly hinder performance). This may partly explain why we do not observe *transfer learning* benefits here (Figure 6.4). For both domains, even though HO2-lim (offline), like MO2, reduces option switch rate when compared with HO2 (offline), this reduction does not lead to a lower *CE*. This suggests that penalising option-switching, achieved with HO2-lim (offline), does not by itself lead to predictable semi-MDP transitions. Altogether, these results suggest that Equations (6.5) and (6.6) lead to compressed abstractions with predictable transitions, favouring longer horizon, higher dimensional domains (arguably where abstractions are most necessary).

6.6 Related Work

Sutton et al.[156] introduce the options framework for discovering temporal-abstractions. Multiple works build on this framework to improve option switching degeneracy[66;52], sample efficiency[177;126], off-policy corrections[82;99] and optimisation[177;149;182]. Wulfmeier et al.[177] present HO2, a hindsight off-policy options framework combining many of the previous advancements, enabling sample-efficient intra-option learning from off-policy data. During MO2's development, Klissarov and Precup[71] published a similar alternate method to HO2 that also trains all options off-policy. While outperforming non-hierarchical RL methods, optimising a purely control objective on the *source* domains and not explicitly encouraging compressed and generalisable options, often leads to high-frequency behaviours that maximise return during training, but that are not beneficial for exploration, generalisation, and credit-assignment during *transfer*.

MO2 is a bottleneck option method similar to McGovern and Barto[91]; Şimşek and Barto[145]; Solway et al.[150]; Harutyunyan et al.[53]; Ramesh et al.[124] discovering temporal-abstractions that induce plans of minimum description length over a set of tasks[150]. Nevertheless, the following existing approaches do not support continuous state-spaces: Harutyunyan et al.[53], as the predictability objective only supports tabular TD-learning; Solway et al.[150], as they require constructing a graph over state-space transitions, thus not easily scalable; Şimşek and Barto[145], building a partial transition

graph with time complexity of $O(T^3)$ (T = horizon length); McGovern and Barto[91], discovering regions of maximum diverse density using exhaustive search, thus scaling poorly. Whilst Ramesh et al.[124] supports continuous spaces, they are on-policy, discovering successor options using PPO, and are thus sample-inefficient. In contrast, MO2 is the first bottleneck options approach that supports offline hindsight learning over continuous state-action spaces.

Hierarchical and variational approaches present alternate methods for discovering and reusing behavioural-abstractions. Hausman et al.[56]; Haarnoja et al.[45]; Wulfmeier et al.[175] discover (multi-level) action-abstractions suited for re-purposing, often by encouraging abstraction distinguishability, but do not support temporal-abstractions. Singh et al.[146]; Merel et al.[93,94]; Salter et al.[134] encourage temporally consistent behaviours by regularising against auto-regressive or learnt priors, yet do not inherently leverage temporal-abstractions for value estimation. Pertsch et al.[115,114]; Ajay et al.[6]; Co-Reyes et al.[21] discover temporally abstract skills, shown essential for exploration, but predefine their temporal resolution. Additionally, unlike bottleneck approaches, all aforementioned works do not explicitly optimise for skills beneficial for planning and acting across tasks, potentially limiting their transfer learning benefits.

Similar to MO2 are methods that impose information-theoretic constraints on the discovered skills. Harutyunyan et al.[53] introduce the termination critic, encouraging compressed options with a low entropic termination state distribution, $\min\{H(s_f|o)\}$. Unlike MO2, the termination critic learns options in a tabular setting, and does not support continuous state-action spaces, limiting their applicability to robotic domains. Furthermore, by building on HO2, MO2 enables intra-option learning, shown to be beneficial. Wang et al.[170] present TAIC, that maximises mutual information between options and initial and termination states, $\max\{I(o;s_i,s_f)\}$, while simultaneously minimising individual mutual informations, $\min\{I(o;s_i)+I(o;s_f)\}$, thereby encouraging options independent of their initial and termination states. Wang et al.[170] learn their abstractions via a variational auto-encoder approach. Unlike MO2, TAIC is on-policy and builds on PPO[138] which reduces sample efficiency. Additionally, unlike MO2, TAIC's options have limited ability to specialise, due to their independence to initial and termination states. Shiarlis et al.[140] introduce TACO, a weakly supervised approach for discovering temporally compressed abstractions aligning with sub-tasks, given a sub-task sketch by the user. MO2 discovers skills unsupervised presenting a more viable approach in many scenarios

where supervision is expensive.

Finally, in contrast to MO2, Machado et al.[88] propose eigen-, instead of bottleneck-, options, demonstrating superior performance in settings when bottlenecks are task-agnostic and shared across non-modular MDPs. MO2 is suited for sharing bottlenecks across modular tasks. Approaches like Jinnai et al.[64] optimise for options (or latent actions) that lead to wide state coverage[31;40;5;66], focusing entirely on exploration. These works take the intrinsic-curiosity RL paradigm, not focusing on compression and predictability as tools for reasoning and acting across long horizons. Explicitly encouraging diversity is important for exploration and skill development when learning from scratch over sparse rewards. Nevertheless, it is unclear whether these skills are temporally compressed and suited for value estimation. We consider extending MO2 to the online skill learning context, and learning from sub-optimal demonstrations[113], as future work.

6.7 Conclusions

We introduce Model-Based Offline Options, MO2, a novel offline hindsight bottleneck options framework supporting continuous state-action spaces, that discovers skills suited for planning (in the form of value estimation) and acting across modular tasks (MDPs whose optimal policies can be obtained by recomposing shared temporal-abstractions). MO2 achieves this with a *cumulative 1-step option-transition log-likelihood* objective encouraging predictable option-terminations across an episode. Once options are learnt offline, we freeze them and perform online transfer learning over the option-induced semi-MDP, learning and acting over the temporally abstract skill space. We compare MO2 against state-of-the-art baselines on complex continuous control domains and perform an extensive ablation study. We demonstrate that MO2's options are bottleneck aligned and improve acting (and exploring), value estimation, and learning of a jumpy option-transition model. On the challenging, sparse, long-horizon, AntMaze domain, MO2 drastically outperforms all baselines.

Chapter 7

Discussions and Future Work

This thesis set out to validate its central claim that **state-, action-, and temporal-abstractions can be used to design domain-independent sample-efficient algorithms that improve compositional generalisation across modular tasks and observations.**

In Section 7.1, we highlight the practicality of this thesis' central aim with respect to embodied continual learners. We continue, in Section 7.2, by summarising the contributions of this thesis and how they support this claim. Finally, we discuss limitations of the presented methods and related future research directions.

7.1 Thesis Practicality

This thesis focuses on **compositional generalisation across modular tasks and observations**. Recalling Section 1.1, we assume there exists beneficial repeated structures (compressed into knowledge abstractions) over the observation, state, action and decision-making states, shared across source M_{source} and target M_{targ} domains, that suffice for obtaining optimal policies π^* when learning and acting instead over abstracted MDPs, $\tilde{M}_{source}$ and $\tilde{M}_{targ}$. This assumption holds if the beneficial structures (those used for inferring optimal policies π^*) present across M_{targ} are on the support of M_{source}. There are two extremes for which this is the case. In the first extreme, the distribution over source and target abstractions are perfectly aligned and observed (e.g. $D_{KL}(p_{source}(X)||p_{targ}(X)) = 0$, with X representing the abstraction space), such as when both source and target MDPs are sampled from

the same MDP distribution (e.g. the same source and transfer tasks). For the other extreme, the source abstraction distribution is so broad that it covers all possible necessary abstractions for solving M_{targ} using $\tilde{M}_{targ}$. This is the case if the source domains observe and cover all possible structures (i.e. when $M_{targ} \in M_{source}$ and $M_{source} = M_{all}$, with M_{all} covering all possible MDPs).

Neither extreme is particularly interesting. The latter requires covering potentially infinite number of source MDPs and will not benefit from abstracting repeated structures as they are inherently so broad that they will not bias how one learns and acts on the target domain. The primer assumes perfect abstraction alignment and is thus a very restrictive assumption, supporting a narrow range of target domains. Nevertheless, there exists many practical scenarios in between these extremes that adhere to the *modularity* requirement and benefit from *compositional generalisation* afforded by the discovered *abstractions*. These scenarios represent when the source domains contain superfluous abstractions for optimal transfer using $\tilde{M}_{targ}$ (e.g. $D_{KL}(p_{source}(X)||p_{targ}(X)) > 0$), but do not cover all possible MDPs ($M_{source} \subset M_{all}$), and thus still bias learning and acting during transfer.

Both fields of *Offline Reinforcement Learning*[76;122;147;75] as well as *Natural Language Programming for Few/Zero Shot Learning*[16;121;25] lie between these two extremes, adhering to the philosophy that in a world of increasingly rich and diverse data accessibility, it is increasingly probable to discover beneficial abstractions from source data that support effective transfer to target domains when reasoning over them. Both fields demonstrate many practical applications from general question answering[16], translation[16;121] and sequence classification tasks[25] to real-world robotic tasks such as pick-and-place[76], open drawer[147;76;122], and stacking[76]. Additionally, for *embodied continual learners*, especially those provided with a curriculum, it becomes increasingly probable across a lifetime that the target domain becomes on the support of the source MDPs[68]. For example, for self-driving vehicles or manufacturing agents, it becomes increasingly probable across a lifetime of navigating distinct routes or constructing distinct assemblies that the current route or assembly can be obtained from prior sub-routes or assemblies. This thesis proposes methods beneficial for these, arguably broad, situations. In Section 7.3, we discuss non-modular examples, and how to extend our methods to tackle them. We continue by summarising the contributions made in Chapters 3 to 6.

7.2 Summary of Contributions

This thesis claims that **abstractions can be leveraged by domain-independent sample-efficient algorithms for improved compositional generalisation across modular domains**. The central focus is to develop methods that ultimately contribute towards embodied continual learning, whilst additionally demonstrating practicality in robotics. As such, the proposed methods all support sample-efficient off-policy learning, leverage minimal supervision for abstraction discovery, and support continuous state-action spaces.

Chapter 3 introduces APRiL, a *domain adaptation*, self-supervised approach for state-abstraction discovery that supports compositional generalisation across modular observations. For our domains, the modular observations all share object-level compositionality which if discovered can support compositional generalisation. APRiL discovers this abstraction in a self-supervised manner, suppresses task-independent distractor objects by using an attention mechanism, improving generalisation to modular observations with additional unseen distractors. APRiL achieves this while improving sample-efficiency in simulation by leveraging privileged information. We apply APRiL to several diverse domains demonstrating consistent performance. We highlight APRiL's applicability to *sim2real* settings, providing robustness to unseen domains, crucial in robotics.

Chapter 4 introduces TACO, a *domain adaptation*, weakly-supervised approach for discovering temporal-abstractions that support compositional generalisation across modular tasks. For our domains, the modular tasks all share sub-task compositionality which if discovered support compositional generalisation. TACO discovers this abstraction in a weakly-supervised manner, being provided with task-sketches. Given a new task-sketch, TACO compositionally generalises to unseen modular tasks with a new ordering of sub-tasks. TACO improves sample-efficiency and segmentation, control, and generalisation performances, whilst requiring less supervision than alternate methods. Additionally, TACO provides a *human-robot control interface*, which is of particular interest to the robotics community. TACO is applied across diverse domains with consistent performance.

Chapter 5 introduces APES, a *transfer learning*, unsupervised approach for discovering action-abstractions supporting compositional generalisation across modular tasks. For our experiments, the modular tasks share common behaviours, namely corridor traversal and block manipulation. APES

discovers these abstractions unsupervised improving generalisation to unseen modular tasks with unseen ordering of block stacks and corridor traversals. Unlike existing approaches, we present a hierarchical KL-regularised framework able to capture complex and generalisable mutli-modal behaviours. We additionally introduce theory behind the transferability-expressivity of our abstractions, and the crucial role information asymmetry plays. Our unsupervised framework takes a step towards automated abstraction reuse in continual learning.

Chapter 6 introduces MO2, a *transfer learning*, unsupervised approach for temporal-abstraction discovery that supports compositional generalisation across modular tasks. For our domains, the modular tasks share common transitions, between bottleneck states, whose discovery would support compositional generalisation. MO2 discovers these abstractions, learning options that transition between bottlenecks. MO2's options improve generalisation to unseen modular tasks that require a unique ordering of bottleneck traversals. MO2 improves both exploration and value estimation compared with its non-bottleneck counterparts. MO2 is evaluated across several domains. Finally, to our knowledge, MO2 provides the first bottleneck options framework meeting embodied continual learning requirements, namely off-policy learning over continuous state-action spaces.

7.3 Limitations and Future Work

We categorise limitations and potential research directions according to problem formulation.

7.3.1 Domain Adaptation

APRiL achieves effective learning and generalisation by leveraging states and object-level segmentations. For domains where states are not clearly definable, such as deformable object manipulation, APRiL is not suited. Additionally, for this domain and others, attention constrained to the simulator object-level semantic space may be sub-optimal for transfer. Relaxing this requirement could be beneficial. To this end, we have investigated an extension where the 'state-space' learner is replaced with one whose input in an unrandomised image, with pixel-level attention. We do not report these experiments in the thesis and leave results for future publications.

TACO assumes optimal task-sketch supervision. Sub-optimal labelling could heavily influence seg-

mentation and control. Extending TACO to handle sub-optimal labelling, taking inspiration from maximum entropy inverse reinforcement learning[70], could be of value. Another extension worth considering is relaxing the requirement for order-dependent task-sketches, learning instead from tags (e.g. from youtube).

7.3.2 Transfer Learning

APES' requirement to sweep over IAs is limiting. Investigating methods that leverage multiple priors for learning with distinct IAs, similar to ensembles[171], could negate this necessity.

APES and MO2 share shortcomings. During transfer, both are vastly sample-inefficient at policy-optimisation despite (experimentally observed) effective exploration. We believe this boils down to the sample-inefficiency of stochastic gradient descent across sparse reward domains. As highlighted in Pritzel et al.[119], the small learning rates required for stable optimisation[90], coupled with the class imbalance problem introduced by sparse rewards, limit the rate at which experiences can be incorporated into the critic and policy. An interesting extension concerns improving policy optimisation, potentially leveraging the abstractions. We propose three directions: 1) Non-parametric RL methods such as Neural Episodic Control[119]. 2) Model-based RL methods[51] using a learnt dynamics model as a generator of free experiences. MO2's transition model can be beneficial here, reducing planning complexity and errors. For planning, a reward model is required. We propose non-parametric methods, such as k-means, for sample-efficiency. 3) General Policy Updates using successor features[12]. Non-parametric methods can be used for reward model learning.

In general, the transfer domain may not support compositional generalisation (Section 1.1) which is when the source abstractions do not suffice for obtaining the optimal transfer policy. For example, consider stacking tasks, where an agent has learnt to manipulate cubes on the source domains, but during transfer additionally needs to manipulate different geometries. In this setting, we wish to leverage abstractions whenever possible (e.g. individual cube stacking), but not be restricted by them (i.e. additionally learn how to stack distinct geometries). Naïvly, instead of just training the abstraction on the source domains, continuing to train them during transfer overcomes this problem, enabling the discovery of novel abstractions. However, from experiments ran during this thesis, for sparse domains, knowledge retention during transfer is problematic. This phenomenon is known as

catastrophic forgetting[68], such as forgetting how to stack cubes during transfer. We propose two primary extensions: 1) Implement network plasticity to promote knowledge retention[69]. 2) Use the *source* domain policy as an *exploratory* policy. Combine experiences from the *exploratory* policy and a newly instantiated *transfer* policy, akin to e-greedy methods[167], for learning. As such, the *transfer* policy is not bounded to behaviours on the support of the *exploratory* policy. For MO2, we additionally suggest augmenting the option-space with additional options during transfer. The original options remain frozen. As such, knowledge retention is achieved whilst discovering new skills.

7.3.3 Embodied Continual Learning

APES and MO2 discover compositionally generalisable abstractions from structured experiences collected by an expert. For many continual learning applications, prior experience may be significantly less structured. It remains unclear how well either approach favours under this setting. For MO2, bottleneck discovery may be impacted. Additionally, for continual learning from scratch, discovering diverse experiences to learn reusable abstractions over will likely be problematic. Incorporating intrinsic curiosity rewards[111] for exploration is one suggestion. Interestingly, MO2's temporal-abstractions may be helpful, aiding long-horizon planning over intrinsic return[21].

In general, many additional hurdles need overcoming for continual embodied machine learners to become reality. These include, but are not limited to: catastrophic forgetting, context detection, forwards and backwards transfer interference, handling active and passive non-stationarities, exploration, cause and effect reasoning, lifelong planning, out-of-distribution generalisation, stability vs plasticity, safety, memory, and finally, appropriate benchmarks and broader evaluation criteria[68]. This thesis takes one small step in the direction of continual embodied machine learners.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] N. Abdo, H. Kretzschmar, L. Spinello, and C. Stachniss. Learning manipulation actions from a few demonstrations. In *2013 IEEE International Conference on Robotics and Automation*, pages 1268–1275. IEEE, 2013.

[3] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

[4] D. Abel, D. Arumugam, L. Lehnert, and M. Littman. State abstractions for lifelong reinforcement learning. In *International Conference on Machine Learning*, pages 10–19. PMLR, 2018.

[5] J. Achiam, H. Edwards, D. Amodei, and P. Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

[6] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

[7] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[8] F. Alet, T. Lozano-Pérez, and L. P. Kaelbling. Modular meta-learning. In *Conference on Robot Learning*, pages 856–868. PMLR, 2018.

[9] J. Andreas, D. Klein, and S. Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175, 2017.

[10] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[11] P. L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pages 1726–1734, 2017.

[12] A. Barreto, S. Hou, D. Borsa, D. Silver, and D. Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48):30079–30087, 2020.

[13] R. Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[14] E. Bengio, J. Pineau, and D. Precup. Interference and generalization in temporal difference learning. In *International Conference on Machine Learning*, pages 767–777. PMLR, 2020.

[15] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351, 2016.

[16] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[17] M. B. Chang, A. Gupta, S. Levine, and T. L. Griffiths. Automatically composing representation transformations as a means for generalization. *arXiv preprint arXiv:1807.04640*, 2018.

[18] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov. Gated-attention architectures for task-oriented language grounding. *arXiv preprint arXiv:1706.07230*, 2017.

[19] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[20] M. Cholodovskis Machado. Efficient exploration in reinforcement learning through time-based representations. 2019.

[21] J. Co-Reyes, Y. Liu, A. Gupta, B. Eysenbach, P. Abbeel, and S. Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International conference on machine learning*, pages 1009–1018. PMLR, 2018.

[22] B. C. Da Silva, E. W. Basso, A. L. Bazzan, and P. M. Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd international conference on Machine learning*, pages 217–224, 2006.

[23] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2(6), 2019.

[24] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2169–2176. IEEE, 2017.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[26] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[27] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

[28] S. Du, A. Krishnamurthy, N. Jiang, A. Agarwal, M. Dudik, and J. Langford. Provably efficient rl with rich observations via latent state decoding. In *International Conference on Machine Learning*, pages 1665–1674. PMLR, 2019.

[29] G. Dulac-Arnold, D. Mankowitz, and T. Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[30] S. Ekvall and D. Kragic. Learning task models from multiple human demonstrations. In *RO-MAN 2006-The 15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 358–363. IEEE, 2006.

[31] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[32] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[33] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[34] R. Fox, S. Krishnan, I. Stoica, and K. Goldberg. Multi-level discovery of deep options. *arXiv preprint arXiv:1703.08294*, 2017.

[35] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[36] A. Galashov, S. M. Jayakumar, L. Hasenclever, D. Tirumala, J. Schwarz, G. Desjardins, W. M. Czarnecki, Y. W. Teh, R. Pascanu, and N. Heess. Information asymmetry in kl-regularized rl. *arXiv preprint arXiv:1905.01240*, 2019.

[37] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[38] A. Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[39] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

[40] K. Gregor, D. J. Rezende, and D. Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[41] T. L. Griffiths, F. Callaway, M. B. Chang, E. Grant, P. M. Krueger, and F. Lieder. Doing more with less: meta-reasoning and meta-learning in humans and machines. *Current Opinion in Behavioral Sciences*, 29:24–30, 2019.

[42] O. Groth, F. B. Fuchs, I. Posner, and A. Vedaldi. Shapestacks: Learning vision-based physical intuition for generalised object stacking. In *ECCV (1)*, volume 11205 of *Lecture Notes in Computer Science*, pages 724–739. Springer, 2018.

[43] O. Groth, F. B. Fuchs, I. Posner, and A. Vedaldi. Shapestacks: Learning vision-based physical intuition for generalised object stacking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 702–717, 2018.

[44] O. Groth, C.-M. Hung, A. Vedaldi, and I. Posner. Goal-conditioned end-to-end visuomotor control for versatile skill primitives. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1319–1325. IEEE, 2021.

[45] T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

[46] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[47] T. Haarnoja, A. Zhou, S. Ha, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.

[48] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[49] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.

[50] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[51] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[52] J. Harb, P.-L. Bacon, M. Klissarov, and D. Precup. When waiting is not an option: Learning options with a deliberation cost. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[53] A. Harutyunyan, W. Dabney, D. Borsa, N. Heess, R. Munos, and D. Precup. The termination critic. *arXiv preprint arXiv:1902.09996*, 2019.

[54] H. V. Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2613–2621, 2010.

[55] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. L. Dean, and C. Boutilier. Hierarchical solution of markov decision processes using macro-actions. *arXiv preprint arXiv:1301.7381*, 2013.

[56] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. 2018.

[57] D. Held, Z. McCarthy, M. Zhang, F. Shentu, and P. Abbeel. Probabilistically safe policy transfer. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 5798–5805. IEEE, 2017.

[58] P. Henderson, W.-D. Chang, P.-L. Bacon, D. Meger, J. Pineau, and D. Precup. OptionGAN: Learning Joint Reward-Policy Options using Generative Adversarial Inverse Reinforcement Learning. *ArXiv e-prints*, Sept. 2017.

[59] G. E. Hovland, P. Sikka, and B. J. McCarragher. Skill acquisition from human demonstration using a hidden markov model. In *Proceedings of IEEE international conference on robotics and automation*, volume 3, pages 2706–2711. Ieee, 1996.

[60] D.-A. Huang, L. Fei-Fei, and J. C. Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision*, pages 137–153. Springer, 2016.

[61] M. Igl, A. Gambardella, J. He, N. Nardelli, N. Siddharth, W. Böhmer, and S. Whiteson. Multitask soft option learning. *arXiv preprint arXiv:1904.01033*, 2019.

[62] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019.

[63] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[64] Y. Jinnai, J. W. Park, M. C. Machado, and G. Konidaris. Exploration in reinforcement learning with deep covering options. In *International Conference on Learning Representations*, 2019.

[65] L. P. Kaelbling. Learning to achieve goals. In *IJCAI*, volume 2, pages 1094–8. Citeseer, 1993.

[66] A. Kamat and D. Precup. Diversity-enriched option-critic. *arXiv preprint arXiv:2011.02565*, 2020.

[67] H. J. Kappen, V. Gómez, and M. Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

[68] K. Khetarpal, M. Riemer, I. Rish, and D. Precup. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020.

[69] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[70] E. Klein, M. Geist, B. Piot, and O. Pietquin. Inverse reinforcement learning through structured classification. In *Advances in Neural Information Processing Systems*, pages 1007–1015, 2012.

[71] M. Klissarov and D. Precup. Flexible option learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[72] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[73] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012. doi: 10.1177/0278364911428653. URL `https://doi.org/10.1177/0278364911428653`.

[74] S. Krishnan, R. Fox, I. Stoica, and K. Goldberg. Ddco: Discovery of deep continuous options for robot learning from demonstrations. In *Conference on Robot Learning*, pages 418–437, 2017.

[75] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[76] A. Kumar, A. Singh, S. Tian, C. Finn, and S. Levine. A workflow for offline model-free robotic reinforcement learning. *arXiv preprint arXiv:2109.10813*, 2021.

[77] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on Robot Learning*, pages 143–156, 2017.

[78] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

[79] H. Le, N. Jiang, A. Agarwal, M. Dudík, Y. Yue, and H. Daumé. Hierarchical imitation and reinforcement learning. In *International Conference on Machine Learning*, pages 2917–2926. PMLR, 2018.

[80] Y. C. Leong, A. Radulescu, R. Daniel, V. DeWoskin, and Y. Niv. Dynamic interaction between reinforcement learning and attention in multidimensional environments. *Neuron*, 93(2):451 – 463, 2017. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2016.12.040. URL `http://www.sciencedirect.com/science/article/pii/S089662731631039X`.

[81] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[82] A. Levy, G. Konidaris, R. Platt, and K. Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

[83] L. Li, T. J. Walsh, and M. L. Littman. Towards a unified theory of state abstraction for mdps. *ISAIM*, 4(5):9, 2006.

[84] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning, 2015.

[85] R. Lioutikov, G. Neumann, G. Maeda, and J. Peters. Learning movement primitive libraries through probabilistic segmentation. *The International Journal of Robotics Research*, 36(8): 879–894, 2017.

[86] S. Liu, G. Lever, Z. Wang, J. Merel, S. Eslami, D. Hennes, W. M. Czarnecki, Y. Tassa, S. Omidshafiei, A. Abdolmaleki, et al. From motor control to team play in simulated humanoid football. *arXiv preprint arXiv:2105.12196*, 2021.

[87] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[88] M. C. Machado, M. G. Bellemare, and M. Bowling. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*, pages 2295–2304. PMLR, 2017.

[89] S. Manschitz, J. Kober, M. Gienger, and J. Peters. Learning to sequence movement primitives from demonstrations. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4414–4421. IEEE, 2014.

[90] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[91] A. McGovern and A. G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. 2001.

[92] H. Mei, M. Bansal, and M. R. Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*, volume 1, page 2, 2016.

[93] J. Merel, L. Hasenclever, A. Galashov, A. Ahuja, V. Pham, G. Wayne, Y. W. Teh, and N. Heess. Neural probabilistic motor primitives for humanoid control. *arXiv preprint arXiv:1811.11711*, 2018.

[94] J. Merel, S. Tunyasuvunakool, A. Ahuja, Y. Tassa, L. Hasenclever, V. Pham, T. Erez, G. Wayne, and N. Heess. Catch & carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG)*, 39(4):39–1, 2020.

[95] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning, 2013.

[96] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[97] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. J. Rezende. Towards interpretable reinforcement learning using attention augmented agents. *ArXiv*, abs/1906.02500, 2019.

[98] R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare. Safe and efficient off-policy reinforcement learning. *arXiv preprint arXiv:1606.02647*, 2016.

[99] O. Nachum, S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *arXiv preprint arXiv:1805.08296*, 2018.

[100] O. Nachum, H. Lee, S. Gu, and S. Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, pages 3303–3313, 2018. URL `https://sites.google.com/view/efficient-hrl`.

[101] Y. Naddaf. Game-independent ai agents for playing atari 2600 console games. 2010.

[102] M. Neunert, A. Abdolmaleki, M. Wulfmeier, T. Lampe, T. Springenberg, R. Hafner, F. Romano, J. Buchli, N. Heess, and M. Riedmiller. Continuous-discrete reinforcement learning for hybrid control in robotics. In *Conference on Robot Learning*, pages 735–751. PMLR, 2020.

[103] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5239–5246, Oct 2012. doi: 10.1109/IROS.2012.6386006.

[104] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.

[105] OpenAI, :, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation, 2018.

[106] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[107] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann. Probabilistic movement primitives. In *Advances in neural information processing systems*, pages 2616–2624, 2013.

[108] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[109] R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, 10, 1997.

[110] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768, May 2009. doi: 10.1109/ROBOT.2009.5152385.

[111] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017.

[112] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1–8. IEEE, 2018.

[113] X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[114] K. Pertsch, Y. Lee, Y. Wu, and J. Lim. Guided reinforcement learning with learned skills.

[115] K. Pertsch, Y. Lee, and J. J. Lim. Accelerating reinforcement learning with learned skill priors. *arXiv preprint arXiv:2010.11944*, 2020.

[116] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. *Robotics: Science and Systems*, 2018.

[117] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

[118] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[119] A. Pritzel, B. Uria, S. Srinivasan, A. P. Badia, O. Vinyals, D. Hassabis, D. Wierstra, and C. Blundell. Neural episodic control. In *International Conference on Machine Learning*, pages 2827–2836. PMLR, 2017.

[120] M. L. Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

[121] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[122] R. Rafailov, T. Yu, A. Rajeswaran, and C. Finn. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*, pages 1154–1168. PMLR, 2021.

[123] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

[124] R. Ramesh, M. Tomar, and B. Ravindran. Successor options: An option discovery framework for reinforcement learning. *arXiv preprint arXiv:1905.05731*, 2019.

[125] K. Rawlik, M. Toussaint, and S. Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Robotics: Science and Systems (RSS)*, 2012.

[126] M. Riemer, M. Liu, and G. Tesauro. Learning abstract options. *arXiv preprint arXiv:1810.11583*, 2018.

[127] M. B. Ring. Child: A first step towards continual learning. In *Learning to learn*, pages 261–292. Springer, 1998.

[128] C. Rosenbaum, I. Cases, M. Riemer, and T. Klinger. Routing networks and the challenges of modular and compositional computation. *arXiv preprint arXiv:1904.12774*, 2019.

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

[130] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016.

[131] F. Sadeghi and S. Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[132] S. Salter, D. Rao, M. Wulfmeier, R. Hadsell, and I. Posner. Attention-privileged reinforcement learning. In *Robotics: Science and Systems Workshop on Structured Approaches to Robot Learning for Improved Generalization*, 2020.

[133] S. Salter, D. Rao, M. Wulfmeier, R. Hadsell, and I. Posner. Attention-privileged reinforcement learning. In J. Kober, F. Ramos, and C. Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 394–408. PMLR, 16–18 Nov 2021. URL `https://proceedings.mlr.press/v155/salter21a.html`.

[134] S. Salter, K. Hartikainen, W. Goodwin, and I. Posner. Priors, hierarchy, and information asymmetry for skill transfer in reinforcement learning. *arXiv preprint arXiv:2201.08115*, 2022.

[135] S. Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer, 2006.

[136] K. Schmeckpeper, A. Xie, O. Rybkin, S. Tian, K. Daniilidis, S. Levine, and C. Finn. Learning predictive models from observation and interaction. In *European Conference on Computer Vision*. Springer, 2020.

[137] J. Schulman, P. Abbeel, and X. Chen. Equivalence between policy gradients and soft Q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

[138] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[139] D. Schwab, T. Springenberg, M. F. Martins, T. Lampe, M. Neunert, A. Abdolmaleki, T. Herkweck, R. Hafner, F. Nori, and M. Riedmiller. Simultaneously learning vision and feature-based control policies for real-world ball-in-a-cup. *arXiv preprint arXiv:1902.04706*, 2019.

[140] K. Shiarlis, M. Wulfmeier, S. Salter, S. Whiteson, and I. Posner. Taco: Learning task decomposition via temporal alignment for control. *arXiv preprint arXiv:1803.01840*, 2018.

[141] N. Y. Siegel, J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess, and M. Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[142] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

[143] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587): 484–489, Jan 2016. ISSN 0028-0836. Article.

[144] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[145] Ö. Şimşek and A. G. Barto. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 95, 2004.

[146] A. Singh, H. Liu, G. Zhou, A. Yu, N. Rhinehart, and S. Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020.

[147] A. Singh, A. Yu, J. Yang, J. Zhang, A. Kumar, and S. Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500*, 2020.

[148] R. B. Slaoui, W. R. Clements, J. N. Foerster, and S. Toth. Robust domain randomization for reinforcement learning. *arXiv preprint arXiv:1910.10537*, 2019.

[149] M. J. Smith, H. Van Hoof, and J. Pineau. An inference-based policy gradient method for learning options. In *35th International Conference on Machine Learning, ICML 2018*, volume 11, pages 7481–7490, 2018. ISBN 9781510867963. URL `https://pdfs.semanticscholar.org/809f/951c77b5a39e2a9d556e9cf9938de87f2393.pdf?{_}ga=2.168186657.1832697831.1553020853-1357972575.1551696370`.

[150] A. Solway, C. Diuk, N. Córdova, D. Yee, A. G. Barto, Y. Niv, and M. M. Botvinick. Optimal behavioral hierarchy. *PLoS computational biology*, 10(8):e1003779, 2014.

[151] I. Sorokin, A. Seleznev, M. Pavlov, A. Fedorov, and A. Ignateva. Deep attention recurrent q-network. *arXiv preprint arXiv:1512.01693*, 2015.

[152] A. Srinivas, M. Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

[153] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[154] M. Stolle and D. Precup. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223. Springer, 2002.

[155] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3 (1):9–44, 1988.

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

[157] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[158] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

[159] Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4496–4506, 2017.

[160] S. Thrun and J. O'Sullivan. Discovering structure in multiple learning tasks: The tc algorithm. In *ICML*, volume 96, pages 489–497, 1996.

[161] S. Thrun and A. Schwartz. Finding structure in reinforcement learning. *Advances in neural information processing systems*, 7, 1994.

[162] D. Tirumala, H. Noh, A. Galashov, L. Hasenclever, A. Ahuja, G. Wayne, R. Pascanu, Y. W. Teh, and N. Heess. Exploiting hierarchy for learning and transfer in kl-regularized rl. *arXiv preprint arXiv:1903.07438*, 2019.

[163] D. Tirumala, A. Galashov, H. Noh, L. Hasenclever, R. Pascanu, J. Schwarz, G. Desjardins, W. M. Czarnecki, A. Ahuja, Y. W. Teh, et al. Behavior priors for efficient reinforcement learning. *arXiv preprint arXiv:2010.14274*, 2020.

[164] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 23–30. IEEE, 2017.

[165] E. Todorov. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems*, pages 1369–1376. MIT Press, 2007.

[166] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[167] M. Tokic. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *Annual Conference on Artificial Intelligence*, pages 203–210. Springer, 2010.

[168] U. Viereck, A. t. Pas, K. Saenko, and R. Platt. Learning a visuomotor controller for real world robotic grasping using simulated depth images. *arXiv preprint arXiv:1706.04652*, 2017.

[169] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[170] W. Wang, Y. Hu, and S. Scherer. Learning temporal abstraction with information-theoretic constraints for hierarchical reinforcement learning. 2019.

[171] M. A. Wiering and H. Van Hasselt. Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):930–936, 2008.

[172] Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[173] Y. Wu, S. Kasewa, O. Groth, S. Salter, L. Sun, O. P. Jones, and I. Posner. Imagine that! leveraging emergent affordances for tool synthesis in reaching tasks. *International Conference on Machine Learning Workshop on Object-Oriented Learning*, 2020.

[174] M. Wulfmeier, I. Posner, and P. Abbeel. Mutual alignment transfer learning. *arXiv preprint arXiv:1707.07907*, 2017.

[175] M. Wulfmeier, A. Abdolmaleki, R. Hafner, J. T. Springenberg, M. Neunert, T. Hertweck, T. Lampe, N. Siegel, N. Heess, and M. Riedmiller. Compositional Transfer in Hierarchical Reinforcement Learning. (1), 2019. URL `http://arxiv.org/abs/1906.11228`.

[176] M. Wulfmeier, A. Abdolmaleki, R. Hafner, J. T. Springenberg, M. Neunert, T. Hertweck, T. Lampe, N. Siegel, N. Heess, and M. Riedmiller. Regularized hierarchical policies for compositional transfer in robotics. *arXiv preprint arXiv:1906.11228*, 2019.

[177] M. Wulfmeier, D. Rao, R. Hafner, T. Lampe, A. Abdolmaleki, T. Hertweck, M. Neunert, D. Tirumala, N. Siegel, N. Heess, et al. Data-efficient hindsight off-policy option learning. *arXiv preprint arXiv:2007.15588*, 2020.

[178] R. Yang, H. Xu, Y. Wu, and X. Wang. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

[179] A. Zhang, Z. C. Lipton, L. Pineda, K. Azizzadenesheli, A. Anandkumar, L. Itti, J. Pineau, and T. Furlanello. Learning causal state representations of partially observable environments. *arXiv preprint arXiv:1906.10437*, 2019.

[180] A. Zhang, C. Lyle, S. Sodhani, A. Filos, M. Kwiatkowska, J. Pineau, Y. Gal, and D. Precup. Invariant causal prediction for block mdps. In *International Conference on Machine Learning*, pages 11214–11224. PMLR, 2020.

[181] F. Zhang, J. Leitner, B. Upcroft, and P. Corke. Vision-based reaching using modular deep networks: from simulation to the real world. *arXiv preprint arXiv:1610.06781*, 2016.

[182] S. Zhang and S. Whiteson. DAC: The Double Actor-Critic Architecture for Learning Options. (NeurIPS), 2019. URL `http://arxiv.org/abs/1904.12691`.

[183] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019.

Chapter 8

Appendix

8.1 Chapter 3 Additional Details

8.1.1 Environments

1. *NavWorld*: In this sparse reward, 2D environment, the goal is for the circular agent to reach the triangular target in the presence of distractor objects. Distractor objects have 4 or more sides (to a maximum of 8) and apart from changing the visual appearance of the environment cannot affect the agent. The state-space consists of the $[x, y]$ locations of all objects. The observation-space comprises RGB images of dimension $(60 \times 60 \times 3)$. The action-space corresponds to the velocity of the agent. The agent only obtains a sparse reward of $+1$ if the particle is within ε of the target, after which the episode is terminated prematurely. The maximum episodic length is 20 steps, and all object locations are randomised between episodes.
2. *JacoReach*: In this 3D environment the goal of the agent is to move the Kinova arm such that the distance between its hand and the diamond-shaped object is minimised. The state-space consists of the quaternion position and velocity of each joint as well as the Cartesian positions of each object. The observation-space comprises RGB images and is of dimension $(100 \times 100 \times 3)$. The action-space consists of the desired relative quaternion positions of each joint (excluding the digits) with respect to their current positions. Mujoco uses a PD controller to execute 20 steps that minimises the error between each joint's actual and target positions. The agent's reward is the negative squared Euclidean distance between the Kinova hand and

diamond object plus an additional discrete reward of $+5$ if it is within ε of the target. The episode is terminated early if the target is reached. All objects are out of reach of the arm and equally far from its base. Between episodes the locations of the objects are randomised along an arc of fixed radius with respect to the base of the Kinova arm. The maximum episodic length is 20 agent steps. Distractor objects are sampled from a subset of ShapeStacks[42] objects, specifically ovoid and pentagonal prism objects.

3. *Walker2D*: In this 2D modified Deepmind Control Suite environment[157] with a continuous action-space the goal of the agent is to walk forward as far as possible within 300 steps. We introduce a limit to episodic length as we found that in practice this helped stabilise learning across all tested algorithms. The observation-space comprises of 2 stacked RGB images and is of dimension $(40 \times 40 \times 6)$. Images are stacked so that velocity of the walker can be inferred. The state-space consists of quaternion position and velocities of all joints. The absolute positions of the walker along the x-axis is omitted such that the walker learns to become invariant to this. The action-space setup is the same as *JacoReach*. The reward is the same as defined in Tassa et al.[157] and consists of two multiplicative terms: one encouraging moving forward beyond a given speed, the other encouraging the torso of the walker to remain as upright as possible. The episode is terminated early if the walker's torso falls beyond either $[-1, 1]$ radians with the vertex or $[0.8, 2.0]$m along the z axis.

8.1.2 Randomisation Procedure

We outline the randomisation procedure taken for each environment during training:

1. *NavWorld*: Randomisation occurs at the start of every episode. We randomise the location, orientation and colour of every object as well as the colour of the background. We therefore hope that our agent can become invariant to these aspects of the environment.
2. *JacoReach*: Randomisation occurs at the start of every episode. We randomise the textures and materials of every object, Kinova arm and background. We randomise the locations of each object along an arc of fixed radius with respect to the base of the Kinova arm. Materials vary in reflectance, specularity, shininess and repeated textures. Textures vary between the following: noisy (where RGB noise of a given colour is superimposed on top of another base colour), gradient (where the colour varies linearly between two predefined colours), uniform (only one

colour). Camera location and orientation are also randomised. The camera is randomised along a spherical sector of a sphere of varying radius whilst always facing the Kinova arm at its centre. We hope that our agent can become invariant to these randomised aspects of the environment.

3. *Walker2D*: Randomisation occurs at the start of every episode as well as after every 50 agent steps. We introduce additional randomisation between episodes due to their increased duration. Due to the MDP setup, intra-episodic randomisation is not an issue. Materials, textures, camera location and orientation, are randomised in the same procedure as for *JacoReach*. The camera is set up to always face the upper torso of the walker.

8.1.3 Implementation Details

Table 8.1: Model architecture. FC() and Conv() represent a fully connected and convolutional network. The arguments of FC() and Conv() take the form [nodes] and [channels, square kernel size, stride] for each hidden layer respectively.

Domain	NavWorld and JacoReach	Walker2D
State Actor	FC([256])	FC([256])
Obs Actor	Conv([[18,7,1],[32,5,1],[32,3,1]])	Conv([[18,8,2],[32,5,1],[16,3,1],[4,3,1]])
State Critic	FC([64,64])	FC([400,300])
Obs Critic	FC([64,64])	FC([400,300])
State Attention	FC([256])	FC([256])
Obs Attention	Conv([[32,8,1],[32,5,1],[64,3,1]])	Conv([[32,8,1],[32,5,1],[64,3,1]])
Replay Size	10^4	2×10^5

In this section, we provide more details on our training setup. Refer to Table 8.1 for the model architecture for each component of APRiL and the asymmetric DDPG baseline. *Obs Actor* and *Obs Critic* setups are the same for both APRiL and the asymmetric DDPG baseline. *Obs Actor* model structure comprises of the convolutional layers (without padding) defined in Table 8.1 followed by one fully connected layer with 256 hidden units, FC([256]). The state-mapping asymmetric DDPG baseline has almost the same architecure as *Obs Actor*, except there is one additional fully connected layer, directly after the convolutional layers that has the same dimensions as the environment state-space. When training this intermediate layer on the L_2 state regressor loss, the state targets are normalised using a running mean and standard deviation, similar to DDPG, to ensure each dimension is evenly weighted and to stabilise targets. The DDPG baseline has the same policy architecture as the other baselines except now the critic is image-based and has the same structure as the actor. All layers use ReLU activations and layer normalisation unless otherwise stated. Each actor network is followed by a tanh activation and rescaled to match the limits of the environment's action-space.

The *State Attention* module includes the fully connected layer defined in Table 8.1 followed by a softmax operation. The *Obs Attention* module has the convolutional layers (with padding to ensure constant dimensionality) outlined in Table 8.1 followed by a fully connected convolutional layer (Conv($[1,1,1]$)) with a sigmoid activation to ensure the outputs vary between 0 and 1. The output of this module is tiled in order to match the dimensionality of the observation-space.

During each iteration of APRiL (for both A_o and A_s) we perform 50 optimization steps on mini-batches of size 64 from the replay buffer. The target actor and critic networks are updated with a Polyak averaging of 0.999. We use Adam optimiser with learning rate of 10^{-3}, 10^{-4} and 10^{-4} for critic, actor and attention networks. We use default TensorFlow values for the other hyperparameters. The discount factor, entropy weighting and self-supervised learning hyperparameters are $\gamma = 0.99$, $\beta = 0.0008$ and $\nu = 1$. To stabilise learning, all input states are normalised by running averages of the means and standard deviations of encountered states. Both actors employ adaptive parameter noise[117] exploration strategy with initial std of 0.1, desired action std of 0.1 and adoption coefficient of 1.01. The settings for the baseline are kept the same as for APRiL where appropriate.

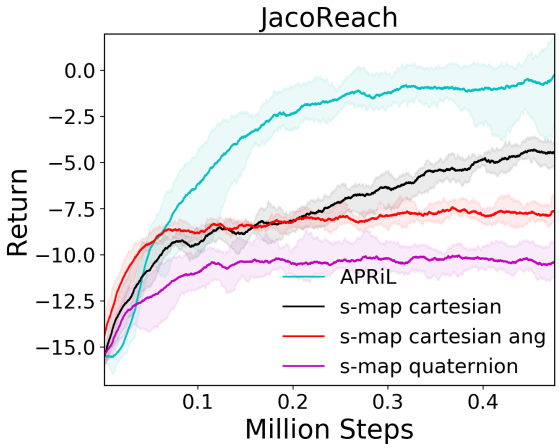


Figure 8.1: We compare learning of APRiL with variants of s-map asym-DDPG. For **s-map cartesian**, **s-map cartesian ang** and **s-map quaternion**, regressed states are Cartesian position, Cartesian position and rotation, and quaternions respectively (for Jaco arm - distractors are always Cartesian).

8.1.4 Attention Visualisation

Figures 8.2 and 8.3 show APRiL's attention maps for policy rollouts on each environment and held-out domain. Attention attends to the task-relevant objects and generalises favourably.

8.1.5 State Mapping Asymmetric DDPG Ablation Study

We found that for *JacoReach*, the choice of state-space to regress to drastically affected the performance of the s-map asym-DDPG baseline. In particular, we observed that if we kept the regressor

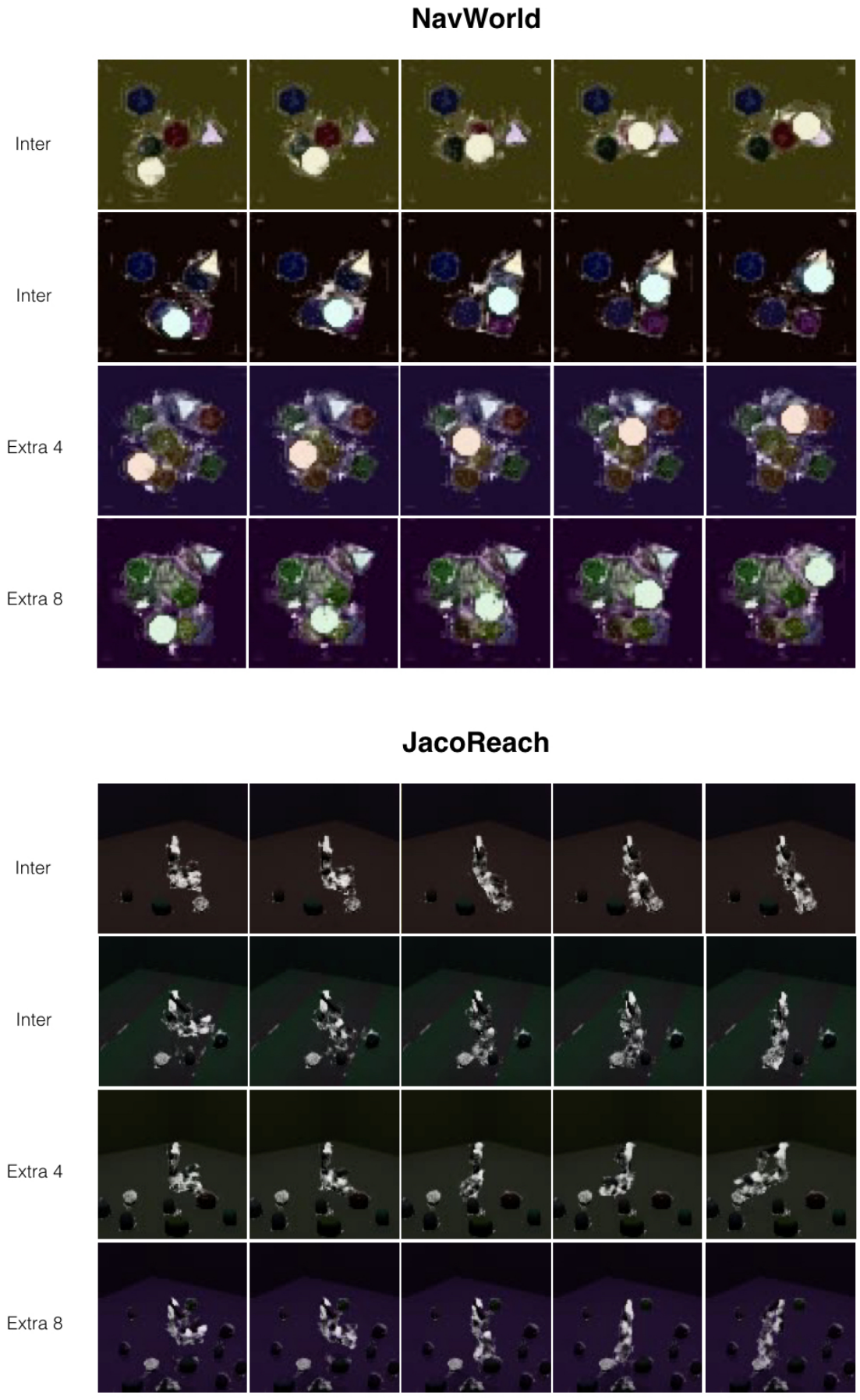


Figure 8.2: APRiL's attention maps for policy rollouts on NavWorld and JacoReach. White and black signify high and low attention values respectively. Attention is correctly paid only to the relevant objects (and Jaco links), even for the extrapolated domains. Refer to Section 3.5.4 for more details.

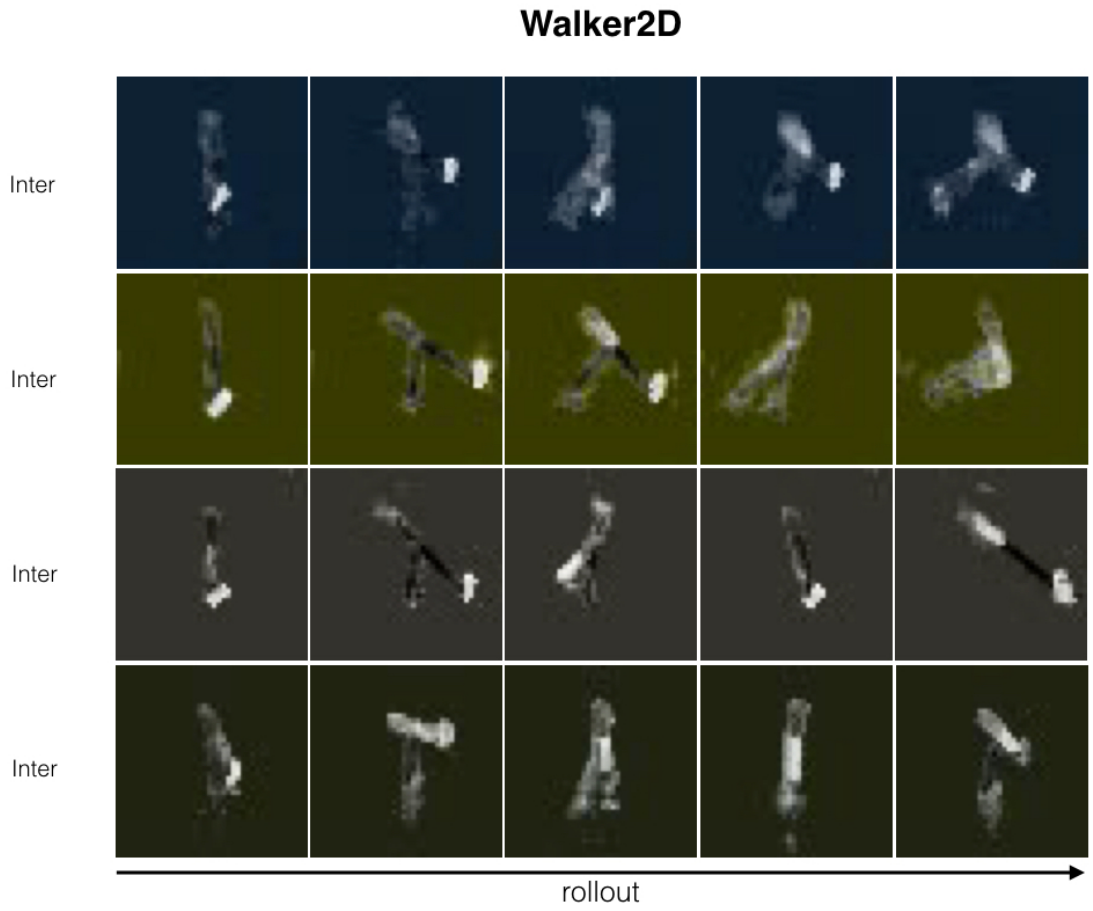


Figure 8.3: APRiL's attention maps for policy rollouts on Walker domain. White and black signify high and low attention values respectively. Attention varies based on the state of the walker. When the walker is upright, high attention is paid to lower limbs. When walking, even attention is paid to every other limb. When about to collapse, high attention is paid to the foot and upper torso. Refer to Section 3.5.4 for more details.

state as quaternions (for Jaco arm links; this is our default state-space setup), that the performance was considerably worse than regressing to Cartesian positions and rotations, and significantly worse than simply regressing to Cartesian positions (see Figure 8.1). Figure 8.4 demonstrates that it is the inability to accurately regress to quaternions and Cartesian rotations that leads to inferior policy performance for these two s-map asym-DDPG ablations. Zhou et al.[183] similarly observed that quaternions are hard for neural networks to regress and showed that it was due to their representations being discontinuous. It is this reason why regressing **only** to Cartesian positions performed best.

However, even with a representation which is better suited for learning, the agent's performance is still significantly below APRiL (see Figure 8.1). Given that the state-space agent used under the APRiL framework learns efficiently for this domain, this suggests that the remainder of the s-map asymmetric DDPG policy (layers dependent on the state-space predictor) is rather sensitive to inaccuracies in the regressor. Different methods for using privileged information, as given by APRiL's attention mechanism, provide more robust performance.

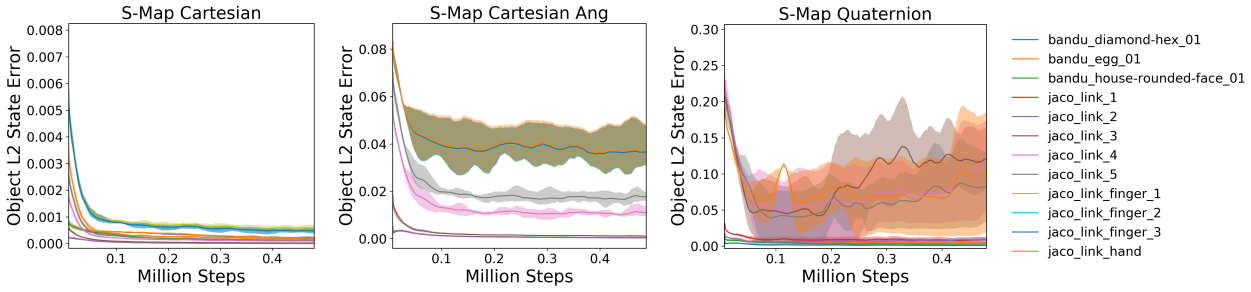


Figure 8.4: S-Map Asym-DDPG normalised state prediction errors. We compare individual object L_2 regressor losses (mean loss over states corresponding to a given object) between **s-map cartesian**, **s-map cartesian ang** and **s-map quaternion**. The object keys are on the right. **S-map quaternion** and **s-map cartesian ang** struggle to regress to quaternions and Cartesian rotations and hence policy performance is restricted.

8.2 Chapter 4 Additional Details

8.2.1 Technical Details

Implementation Details and Architecture

The policies for all tasks are modelled as MLPs unless images are used. In this case, convolutional layers, shared between all sub-policies, are used to extract state features. In continuous domains, action probabilities are modelled as $a \sim \mathcal{N}(\mu, \sigma = 1)$ where μ is the output of the MLP. In discrete domains, action probabilities are modelled using categorical distributions. We found that having separate networks for the domain actions and the a_{STOP} action, leads to better performance, especially for the Dial domain. See Table 8.2 for implementation details of the architectures used in each experiment where *Core* represents the convolutional architecture that learns a feature representation of the input space before feeding it into the stop and action policies.

For larger domains we found that dropout on the a_{STOP} policy greatly improved results. It serves as regulariser as well as to ensure optimising over a broad range of paths as various alignment paths are sampled when units in the network are dropped. For further performance, we exponentially decrease the dropout rate. All models are implemented in TensorFlow[1].

Experiment	State Dim	Core	Stop Policy	Action Policy	Output Dim
Nav World	8	-	FC [100]	FC [100]	2
Craft	1076	-	FC [400,100]	FC [400,100]	7
Dial	39	-	FC [300,200,100]	FC [300,200,100]	9
Dial (Images)	[112,112,3]	conv[10,5,3] kernel[5,5,3] stride[2,2,1]	FC [400,300]	FC [400,300]	9

Table 8.2: Specification of the architectures used in each experiment. For experiment *Dial (Images)* conv[], kernel[] and stride[] represent the number of channels, dimension of square kernels and 2D strides per convolutional layer respectively

Data Collection

As mentioned in the Chapter 4, data collection took place using DART[77] in order to compensate for the issue of covariate shift and compounding errors during policy deployment. For the Dial domain we add noise to each joint, proportional to the maximum allowed torque, which was manually tuned. During demonstrations we add a varying degree of noise to better cover the state-space resulting in

more robust policies that are better able to handle sub-optimality in the test domain.

8.2.2 Detailed Results

This section covers more detailed results for different domains and algorithms. For the NavWorld scenario, we report results on non-0-shot settings. For the Dial domain, we additionally address the sub-task accuracy, i.e how many sub-tasks were completed out of the total attempted. Finally, we detail alignment accuracy values for the methods considered in the Chapter 4. The metric describes the percentage of correctly aligned sub-policies on a set of hold-out tasks.

NavWorld Domain

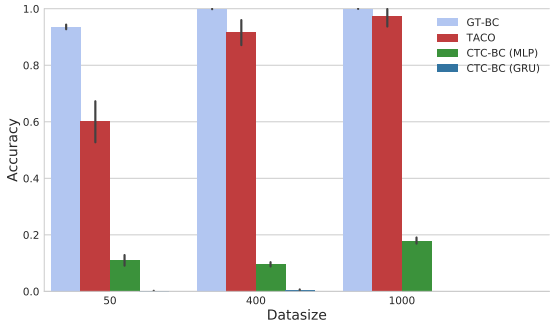


Figure 8.5: Mean task accuracy on NavWorld. $L = L_{test} = 3$. Reporting mean value across 100 tasks and 100 agents. Black error bars represent min/max achieved mean task accuracy between agents.

As is seen in Figure 8.5, TACO not only vastly outperforms CTC-BC (MLP and GRU), both of which favour poorly in NavWorld, but as well asymptotically approaches the fully supervised GT-BC with increasing number of training trajectories.

Joint-State Dial Domain

The sub-task accuracy is displayed in Figure 8.6.

Sequence Alignment Accuracy

Alignment accuracy is measured as the % of agreement between the ground truth sub-task sequence of length T and the most likely sequence predicted by either of the three alignment architectures considered in Chapter 4, TACO, CTC-BC (MLP), CTC-BC (GRU). The most likely sequence is given by taking the arg max of the forward variables at each time-step for either CTC or TACO. For GT-BC, we pass the learned policies through TACO alignment without learning. We compute the

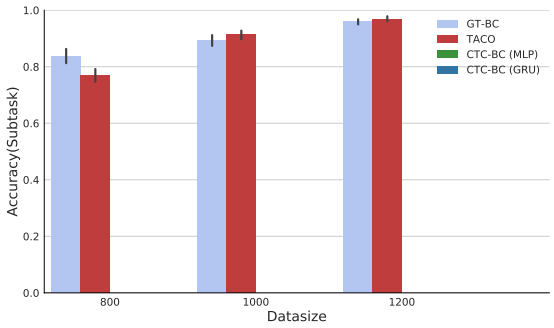


Figure 8.6: Mean sub-task accuracy for the Dial domain. Black error bars represent min/max achieved values. We measure the number of sub-tasks succeeded over the total given, for 5 independent evaluations of 20 tasks of task length 5. Due to the complexity of the task and bad alignment performance, CTC based methods are unable to complete any sub-tasks.

alignment accuracy for 100 hold-out test sequences, which come from the same distribution as the training data. The results across all the experiment domains are stated in Table 8.3.

	Domain			
Algorithm	Nav-World	Craft	Dial	Dial (Image)
TACO	**95.3**	95.6	**98.9**	**99.0**
CTC-BC (MLP)	89.0	41.4	31.4	84.6
CTC-BC (GRU)	80.0	57.1	28.6	48.8
GT-BC (aligned with TACO)	94.6	**99.4**	98.7	98.2

Table 8.3: Alignment accuracy of each algorithm for all domains. TACO always outperforms CTC emphasising the importance of maximising the joint likelihood of task sequences and actions.

8.2.3 CTC Probabilistic Sub-Policy Training

While the naive extension of CTC (Section 4.4.1) trains sub-policies based on the a single alignment by taking the arg max of the forward variables $\alpha_t(l)$ for every time-step, this paragraph addresses the possible probabilistic assignment of active sub-policies.

To obtain the probabilistic weighting based optimisation objective in (8.1), we first define the probability distribution $p_t(l)$ at time-step t over all sub-policies l in a sketch based on the normalised CTC forward variables in (8.2). However, as the ground-truth targets in the trajectory ρ only exist for the regular actions, we first compute the stop action probability targets based on the CTC forward variables.

$$\theta^*_{k=1,..,K} = \arg\max_{\theta_k} \mathbb{E}_{\rho_k}[\sum_{t=1}^{T}\sum_{l=1}^{L} \log p_t(l)\pi_{\theta_k}(a_t^+|s_t)] \tag{8.1}$$

$$\text{where} \quad p_t(l) = \frac{\alpha_t(l)}{\sum_{l_i=1}^{L}\alpha_t(l_i)}. \tag{8.2}$$

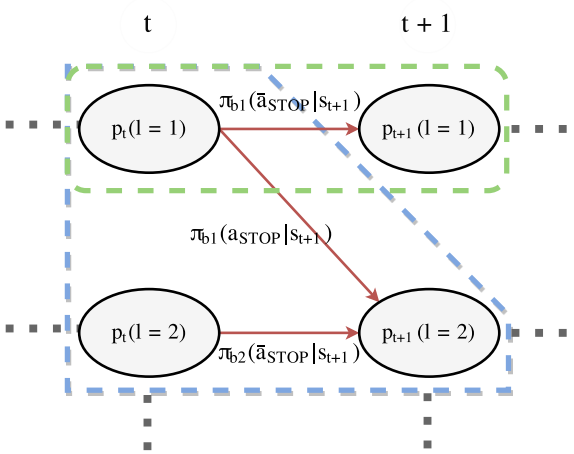


Figure 8.7: CTC-based computation of stop action targets. The green and blue areas respectively depict the relations for $l = 1$ in (8.3) and $l > 1$ in (8.4).

To determine probabilistic targets for the stop actions, we associate the edges in Figure 8.7 between nodes of the same or subsequent sub-policies respectively with $\bar{a}_{stop}$ and a_{stop}. For $l = 1$, nodes depend only on a single relevant edge from the same sub-policy at the previous time-step, while for all nodes with $l > 1$ we take into account edges from the same sub-policy and the previous sub-policy at the previous time-step.

$$p_{t+1}(1) = p_t(1) \cdot \pi_1(\bar{a}_{stop}|s_{t+1})] \tag{8.3}$$

$$p_{t+1}(l+1) = p_t(l+1) \cdot \pi_{l+1}(\bar{a}_{stop}|s_{t+1}) + p_t(l) \cdot \pi_l(a_{stop}|s_{t+1}) \tag{8.4}$$

$$\pi_l(a_{stop}|s_t) = 1 - \pi_l(\bar{a}_{stop}|s_t) \tag{8.5}$$

Based on (8.3), (8.4) and (8.5), we can derive the targets in (8.6) for $t = 1, \ldots, T-1$. Starting with the targets for $l = 1$ we can compute the targets for $l+1$ based on the targets for l and the forward variables $\alpha_t(l)$.

$$\pi_l(\bar{a}_{stop}|s_{t+1}) \begin{cases} \frac{p_t(l)}{p_{t+1}(l)} & \text{if } l = 1, \\ \\ \frac{p_{t+1}(l+1)}{p_t(l+1)} - \frac{p_t(l) \cdot (1 - \pi_l(\bar{a}_{stop}|s_{t+1}))}{p_t(l+1)} & \text{if } l > 1. \end{cases} \tag{8.6}$$

8.3 Chapter 5 Additional Details

8.3.1 Method

Training Regime

In this section, we algorithmically describe our training setup. We relate each training phase to the principle equations introduced in Chapter 5, but note that Section 8.3.1 outlines a more detailed version of these equations that were actually used.

Algorithm 2 APES training regime

 # Full training and transfer regime. For BC, gradients are prevented from flowing from π_0 to π. In practice $\pi_0 = \{\pi_i\}_{i\in\{0,\dots,N\}}$, multiple trained priors. During transfer, π_H, is reinitialised.

 # Behavioural Cloning
Initialise: policy π, prior π_0, replay R_{bc}, DAGGER rate r, environment *env*
for Number of BC training steps **do**
    R_{bc}, *env* $\leftarrow$ *collect*(π, R_{bc}, *env*, True, r)
    π, $\pi_0 \leftarrow$ *BC_update*(π, π_0, R_{bc}) *# Eq. 5.4*
end for
 # Reinforcement Learning
Initialise: high level policy π^H, critics $Q_{k\in\{1,2\}}$, replay R_{rl}, transfer environment env_t
for Number of RL training steps **do**
    R_{rl}, $env_t \leftarrow$ *collect*(π, R_{rl}, env_t)
    $\pi^H \leftarrow$ *RL_policy_update*(π, π_0, R_{rl}) *# Eq. 5.5*
    $Q_k \leftarrow$ *RL_critic_update*(Q_k, π, R_{rl}) *# Eq. 8.11*
end for

Algorithm 3 collect

 # Collects experience from either π_i or π_e, applying DAGGER at a given rate if instructed, and updates R_j, env accordingly.
function COLLECT(π_i, R_j, *env*, *dag*=False, r=1)
    $\mathbf{x} \leftarrow$ *env*.observation()
    $\pi_e \leftarrow$ *env*.expert()
    $\mathbf{a}_i \leftarrow \pi_i(\mathbf{x})$
    $\mathbf{a}_e \leftarrow \pi_e(\mathbf{x})$
    $\mathbf{a} \leftarrow$ Bernoulli([$\mathbf{a}_i$, $\mathbf{a}_e$], [r, 1 - r])
    $\mathbf{x}', r_k, env \leftarrow env$.step($\mathbf{a}$)
    if *dag* **then**
        $\mathbf{a}_f \leftarrow \mathbf{a}_e$
    else
        $\mathbf{a}_f \leftarrow \mathbf{a}_i$
    end if
    $R_j \leftarrow R_j$.update($\mathbf{x}, \mathbf{a}_f, \mathrm{r}_k, \mathbf{x}'$)
    return R_j, *env*
end function

Variational Behavioural Cloning and Reinforcement Learning

Behavioural Cloning (BC) and KL-Regularised RL, when considered from the variational inference perspective, share many similarities. These similarities become even more apparent when dealing with hierarchical models. A particularly unifying choice of objective functions for BC and RL that fit with off-policy, generative, hierarchical RL, desirable for sample efficiency, are:

$$O_{bc}(\pi, \{\pi_i\}^{i\in I}) = -\sum_{i\in I} \mathrm{D}_{\mathrm{KL}}\left(\pi(\tau) \parallel \pi_i(\tau)\right), \quad O_{rl}(\pi, \{\pi_i\}^{i\subset I}) = E_{\pi(\tau)}[R(\tau)] + O_{bc}(\pi, \{\pi_i\}^{i\subset I}) \quad (8.7)$$

O_{bc}, corresponds to the KL-divergence between trajectories from the policy, π, and various priors, π_i. For BC, $i \in \{0, u, e\}$, denote the learnt, uniform, and expert priors. For BC, in practice, we train multiple priors in parallel: $\pi_0 = \{\pi_i\}_{i\in\{0,\dots,N\}}$. We leave this notation out for the remainder of this section for simplicity. When considering only the expert prior, this is the reverse KL-divergence, opposite to what is usually evaluated in the literature[115]. O_{rl} refers to a lower bound on the expected optimality of each prior $\log p_{\pi_i}(O=1)$; O denoting the event of achieving maximum return (return referred to as $R(.)$); refer to Abdolmaleki et al.[3] and Section 8.3.2 for this proof, a further explanation, and necessary conditions. During transfer, we do **not** have access to the expert ($i \subset I := i \in \{0, u\}$).

For hierarchical policies, the KL terms are not easily evaluable. $\mathrm{D}_{\mathrm{KL}}(\pi(\tau) \parallel \pi_i(\tau)) \leq \sum_t \mathbb{E}_{\pi(\tau)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^H(\mathbf{z}_t|\mathbf{x}_k) \parallel \pi_i^H(\mathbf{z}_t|\mathbf{x}_k)\right) + \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_k)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^L(\mathbf{a}_t|\mathbf{x}_k,\mathbf{z}_t) \parallel \pi_i^L(\mathbf{a}_t|\mathbf{x}_k,\mathbf{z}_t)\right)\right]\right]$ [1] is a commonly chosen upper bound[162]. If sharing modules, e.g. $\pi_i^L = \pi^L$, or using non-hierarchical networks, this bound can be simplified (removing the second or first terms respectively). To make both Equation (8.7) amendable to off-policy training (experience from $\{\pi_e, \pi_b\}$, for BC/RL respectively; π_b representing behavioural policy), we introduce importance weighting (IW), removing off-policy bias at the expense of higher variance. Combining all the above with additional individual term weighting hyperparameters, $\{\beta_i^z, \beta_i^a\}$, we obtain:

$$\tilde{\mathrm{D}}_{\mathrm{KL}}^{q(\tau)}(\pi(\tau)||\pi_i(\tau)) := \mathbb{E}_{q(\tau)}\left[\sum_t \nu^q[t] \cdot (\beta_i^z \cdot C_{i,h}(\mathbf{z}_t|\mathbf{x}_k) + \beta_i^a \cdot \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_k)}\left[C_{i,l}(\mathbf{a}_t|\mathbf{x}_k,\mathbf{z}_t)\right])\right]$$

$$\zeta_i^n = \frac{\mathbb{E}_{\pi^H(z_i|\mathbf{x}_i,k)}\left[\pi^L(\mathbf{a}_i|\mathbf{x}_i,\mathbf{z}_i,k)\right]}{n(\mathbf{a}_i|\mathbf{x}_i,k)},\ \nu^n = \left[\zeta_1^n, \zeta_1^n\zeta_2^n, \dots, \prod_{i=1}^{\tau_t}\zeta_i^n\right],\ C_{\mu,\varepsilon}(y) = \log\left(\frac{\pi_\varepsilon^r(y)}{\pi_{\mu,\varepsilon}^r(y)}\right)$$

$$-\mathrm{D}_{\mathrm{KL}}(\pi(\tau)||\pi_e(\tau)) \geq -\sum_{i\in\{0,u,e\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_e(\tau)}(\pi(\tau)||\pi_i(\tau)) \tag{8.8}$$

$$\mathbb{E}_{\substack{p(\mathscr{K}),\\ \pi_0(\tau)}}[\log(O=1|\tau,k)] \geq \mathbb{E}_{\pi_b(\tau)}\left[\sum_t \nu^{\pi_b}[t] \cdot r_k(\mathbf{x}_t,\mathbf{a}_t)\right] - \sum_{i\in\{0,u\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_b(\tau)}(\pi(\tau)||\pi_i(\tau)) \tag{8.9}$$

Where $\tilde{\mathrm{D}}_{\mathrm{KL}}^{q(\tau)}(\pi(\tau)||\pi_i(\tau))$ (for $\{\beta_i^z, \beta_i^a\} = 1$) is an unbiased estimate for the aforementioned upper bound, using q's experience. ζ_i^n is the IW for time-step i, between π and arbitrary policy n. $\nu^n[t]$ is the t^{th} element of ν^n; the cumulative IW product at time-step t. Equations (8.8) and (8.9) are the BC/RL lower bounds used for policy gradients. See Section 8.3.2 for a derivation, and necessary

[1] For proof refer to Section 8.3.2

conditions, of these bounds. For BC, this bounds the KL-divergence between hierarchical and expert policies, π, π_e. For RL, this bounds the expected optimality, for the learnt prior policy, π_0. Intuitively, maximising this particular bound, maximises return for both policy and prior, whilst minimising the disparity between them. Regularising against an uninformative prior, π_u, encourages highly-entropic policies, further aiding at exploration and stabilising learning[61].

In RL, IWs are commonly ignored[84;3;46], thereby considering each sample equally important. This is also convenient for BC, as IWs require the expert probability distribution: not usually provided. We did not observe benefits of using them and therefore ignore them too. We employ module sharing ($\pi_i^L = \pi^L$; unless stated otherwise), and freeze certain modules during distinct phases, and thus never employ more than 2 hyperparameters, β, at any given time, simplifying the hyperparameter optimisation. These weights balance an exploration/exploitation trade-off. We use a categorical latent space, explicitly marginalising over latents, rather than using sampling approximations[63]. For BC, we train for 1 epoch (referring to training, in the expectation, once per sample in the replay buffer).

Critic Learning

The lower bound presented in Equation (8.9) is non-differentiable due to rewards being sampled from the environment. Therefore, as is common in the RL literature[96;84], we approximate the return of policy π with a critic, Q. To be sample efficient, we train in an off-policy manner with TD-learning[155] using the Retrace algorithm[98] to provide a low-variance, low-bias, policy evaluation operator:

$$Q_t^{\text{ret}} := Q'(\mathbf{x}_t, \mathbf{a}_t, k) + \sum_{j=t}^{\infty} \varepsilon_j^t \left[r_k(\mathbf{x}_j, \mathbf{a}_j) + \mathbb{E}_{\substack{\pi^H(\mathbf{z}|\mathbf{x}_{j+1},k),\\ \pi^L(\mathbf{a}'|\mathbf{x}_{j+1},\mathbf{z},k)}} \left[Q'(\mathbf{x}_{j+1}, \mathbf{a}', k) \right] - Q'(\mathbf{x}_j, \mathbf{a}_j, k) \right] \tag{8.10}$$

$$L(Q) = \mathbb{E}_{\substack{p(\mathscr{K}),\\ \pi_b(\tau)}} \left[(Q(\mathbf{x}_t, \mathbf{a}_t, k) - \arg\min_{Q_t^{\text{ret}}}(Q_t^{\text{ret}}))^2 \right] \qquad \varepsilon_j^t = \gamma^{j-t} \prod_{i=t+1}^{j} \zeta_i^b \tag{8.11}$$

Where Q_t^{ret} represents the policy return evaluated via Retrace. Q' is the target Q-network, commonly used to stabilise critic learning[96], and is updated periodically with the current Q values. IWs are not ignored here, and are clipped between $[0, 1]$ to prevent exploding gradients[98]. To further reduce bias and overestimates of our target, Q_t^{ret}, we apply the double Q-learning trick[54], and concurrently learn two target Q-networks, Q'. Our critic is trained to minimise the loss in Equation (8.11), which

regularises the critic against the minimum of the two targets produced by both target networks.

8.3.2 Theory and Derivations

In this section, we provide proofs for the theory introduced in Chapter 5 and Section 8.3.1.

Theorem 1

Theorem 1. *The more random variables a network depends on, the larger the covariate shift (input distributional shift, here represented by KL-divergence) encountered across sequential tasks. That is, for distributions p, q*

$$\begin{gathered}\mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{b}) \parallel q(\mathbf{b})\right) \geq \mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{c}) \parallel q(\mathbf{c})\right) \\ \textit{with } \mathbf{b} = (b_0, b_1, ..., b_n) \textit{ and } \mathbf{c} \subset \mathbf{b}.\end{gathered} \tag{8.12}$$

Proof

$$\begin{aligned}\mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{b}) \parallel q(\mathbf{b})\right) &= \mathbb{E}_{p(\mathbf{b})}\left[\log\left(\frac{p(\mathbf{b})}{q(\mathbf{b})}\right)\right] \\ &= \mathbb{E}_{p(\mathbf{d}|\mathbf{c})\cdot p(\mathbf{c})}\left[\log\left(\frac{p(\mathbf{d}|\mathbf{c})\cdot p(\mathbf{c})}{q(\mathbf{d}|\mathbf{c})\cdot q(\mathbf{c})}\right)\right] \quad \textit{with} \quad \mathbf{d} \in \mathbf{b} \oplus \mathbf{c} \\ &= \mathbb{E}_{p(\mathbf{c})}\left[\mathbb{E}_{p(\mathbf{d}|\mathbf{c})}\left[1\right]\cdot\log\left(\frac{p(\mathbf{c})}{q(\mathbf{c})}\right)\right] + \mathbb{E}_{p(\mathbf{c})}\left[\mathbb{E}_{p(\mathbf{d}|\mathbf{c})}\left[\log\left(\frac{p(\mathbf{d}|\mathbf{c})}{q(\mathbf{d}|\mathbf{c})}\right)\right]\right] \\ &= \mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{c}) \parallel q(\mathbf{c})\right) + \mathbb{E}_{p(\mathbf{c})}\left[\mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{d}|\mathbf{c}) \parallel q(\mathbf{d}|\mathbf{c})\right)\right] \\ &\geq \mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{c}) \parallel q(\mathbf{c})\right) \quad \textit{given} \quad \mathbb{E}_{p(\mathbf{c})}\left[\mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{d}|\mathbf{c}) \parallel q(\mathbf{d}|\mathbf{c})\right)\right] \geq 0\end{aligned} \tag{8.13}$$

Theorem 2

Theorem 2. *The more random variables a network depends on, the greater its ability to distil knowledge in the expectation (output distributional shift between network and target distribution, here represented by the expected KL-divergence). That is, for target distribution p and network q with outputs a and possible inputs* $\mathbf{b}$*,* $\mathbf{c}$*,* $\mathbf{d}$*, such that* $\mathbf{b} = (b_0, b_1, ..., b_n)$ *and* $\mathbf{d} \subset \mathbf{c} \subset \mathbf{b}$

$$\mathbb{E}_{q(\mathbf{e}|\mathbf{d})}\left[\mathrm{D}_{\mathrm{KL}}\left(p(a|\mathbf{b}) \parallel q(a|\mathbf{c})\right)\right] \leq \mathrm{D}_{\mathrm{KL}}\left(p(a|\mathbf{b}) \parallel q(a|\mathbf{d})\right) \quad \textit{with} \quad \mathbf{e} \in \mathbf{d} \oplus \mathbf{c} \tag{8.14}$$

Proof

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}\left(p(a|\mathbf{b}) \parallel q(a|\mathbf{d})\right) &= \mathbb{E}_{p(a|\mathbf{b})}\left[\log\left(\frac{p(a|\mathbf{b})}{q(a|\mathbf{d})}\right)\right] \\
&= \mathbb{E}_{p(a|\mathbf{b})}\left[\log p(a|\mathbf{b}) - \log \mathbb{E}_{q(\mathbf{e}|\mathbf{d})}\left[q(a|\mathbf{c})\right]\right] \quad \textit{with} \quad \mathbf{e} \in \mathbf{d} \oplus \mathbf{c} \\
&\geq \mathbb{E}_{p(a|\mathbf{b})\cdot q(\mathbf{e}|\mathbf{d})}\left[\log\left(\frac{p(a|\mathbf{b})}{q(a|\mathbf{c})}\right)\right] \quad \text{given Jensen's Inequality} \\
&= \mathbb{E}_{q(\mathbf{e}|\mathbf{d})}\left[\mathrm{D}_{\mathrm{KL}}\left(p(a|\mathbf{b}) \parallel q(a|\mathbf{c})\right)\right]
\end{aligned} \tag{8.15}$$

Hierarchical KL-Divergence Upper Bound

Most of the following proofs ignore multi-task setting. Multi-task extensions are trivial.

Upper Bound

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}\left(\pi(\tau) \parallel \pi_i(\tau)\right) \leq \sum_t \mathbb{E}_{\pi(\tau)}\big[&\mathrm{D}_{\mathrm{KL}}\left(\pi^H(\mathbf{z}_t|\mathbf{x}_t) \,\middle\|\, \pi_i^H(\mathbf{z}_t|\mathbf{x}_t)\right) \\
&+ \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_t)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t) \,\middle\|\, \pi_i^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t)\right)\right]
\end{aligned} \tag{8.16}$$

Proof

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}\left(\pi(\tau) \parallel \pi_i(\tau)\right) &= \mathbb{E}_{\pi(\tau)}\left[\log\left(\frac{\pi(\tau)}{\pi_i(\tau))}\right)\right] \\
&= \mathbb{E}_{\pi(\tau)}\left[\log\left(\frac{p(\mathbf{s}_0)\cdot\prod_t p(\mathbf{s}_{t+1}|\mathbf{x}_t,\mathbf{a}_t)\cdot\pi(\mathbf{a}_t|\mathbf{x}_t)}{p(\mathbf{s}_0)\cdot\prod_t p(\mathbf{s}_{t+1}|\mathbf{x}_t,\mathbf{a}_t)\cdot\pi_i(\mathbf{a}_t|\mathbf{x}_t)}\right)\right] \\
&= \mathbb{E}_{\pi(\tau)}\left[\log\left(\prod_t \frac{\pi(\mathbf{a}_t|\mathbf{x}_t)}{\pi_i(\mathbf{a}_t|\mathbf{x}_t)}\right)\right] \\
&= \sum_t \mathbb{E}_{\pi(\tau)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi(\mathbf{a}_t|\mathbf{x}_t) \parallel \pi_i(\mathbf{a}_t|\mathbf{x}_t)\right)\right] \\
&\leq \sum_t \mathbb{E}_{\pi(\tau)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi(\mathbf{a}_t|\mathbf{x}_t) \parallel \pi_i(\mathbf{a}_t|\mathbf{x}_t)\right) + \right. \\
&\left. \mathbb{E}_{\pi(\mathbf{a}_t|\mathbf{x}_t)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi(\mathbf{z}_t|\mathbf{x}_t,\mathbf{a}_t) \parallel \pi(\mathbf{z}_t|\mathbf{x}_t,\mathbf{a}_t)\right)\right]\right] \\
&= \sum_t \mathbb{E}_{\pi(\tau)}\left[\mathbb{E}_{\pi(\mathbf{a}_t,\mathbf{z}_t|\mathbf{x}_t)}\left[\log\left(\frac{\pi(\mathbf{a}_t|\mathbf{x}_t)}{\pi_i(\mathbf{a}_t|\mathbf{x}_t)}\right) + \log\left(\frac{\pi(\mathbf{z}_t|\mathbf{x}_t,\mathbf{a}_t)}{\pi_i(\mathbf{z}_t|\mathbf{x}_t,\mathbf{a}_t)}\right)\right]\right] \\
&= \mathbb{E}_{\pi(\tau)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi(\mathbf{a}_t,\mathbf{z}_t|\mathbf{x}_t) \parallel \pi_i(\mathbf{a}_t,\mathbf{z}_t|\mathbf{x}_t)\right)\right] \\
&= \sum_t \mathbb{E}_{\pi(\tau)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^H(\mathbf{z}_t|\mathbf{x}_t) \,\middle\|\, \pi_i^H(\mathbf{z}_t|\mathbf{x}_t)\right)\right. \\
&\left. + \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_t)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t) \,\middle\|\, \pi_i^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t)\right)\right]\right]
\end{aligned} \tag{8.17}$$

Policy Gradient Lower Bounds

Importance Weights Derivation

$$\tilde{\mathrm{D}}_{\mathrm{KL}}^{q(\tau)}(\pi(\tau)||\pi_i(\tau)) = ub(\mathrm{D}_{\mathrm{KL}}(\pi(\tau) \parallel \pi_i(\tau))) \tag{8.18}$$

For $\beta_i^z, \beta_i^a = 1$, where $ub(\mathrm{D}_{\mathrm{KL}}(\pi(\tau) \parallel \pi_i(\tau)))$ corresponds to the hierarchical upper bound introduced in Section 8.3.1.

Proof

$$\begin{aligned}
ub(\mathrm{D}_{\mathrm{KL}}(\pi(\tau) \parallel \pi_i(\tau))) &= \sum_t \mathbb{E}_{\pi(\tau)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^H(\mathbf{z}_t|\mathbf{x}_t) \,\middle\|\, \pi_i^H(\mathbf{z}_t|\mathbf{x}_t)\right)\right. \\
&\quad \left. + \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_t)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t) \,\middle\|\, \pi_i^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t)\right)\right]\right. \\
&= \sum_t \mathbb{E}_{q(\tau)\cdot\frac{\pi(\tau)}{q(\tau)}}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^H(\mathbf{z}_t|\mathbf{x}_t) \,\middle\|\, \pi_i^H(\mathbf{z}_t|\mathbf{x}_t)\right)\right. \\
&\quad \left. + \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_t)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t) \,\middle\|\, \pi_i^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t)\right)\right]\right. \\
&= \sum_t \mathbb{E}_{q(\tau)\cdot\prod_{i=0}^t \frac{\pi(\mathbf{a}_i|\mathbf{x}_i)}{q(\mathbf{a}_i|\mathbf{x}_i)}}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^H(\mathbf{z}_t|\mathbf{x}_t) \,\middle\|\, \pi_i^H(\mathbf{z}_t|\mathbf{x}_t)\right)\right. \\
&\quad \left. + \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_t)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t) \,\middle\|\, \pi_i^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t)\right)\right]\right. \\
&= \tilde{\mathrm{D}}_{\mathrm{KL}}^{q(\tau)}(\pi(\tau)||\pi_i(\tau))
\end{aligned} \tag{8.19}$$

Behavioural Cloning Upper Bound

$$\begin{gathered}
-\mathrm{D}_{\mathrm{KL}}(\pi(\tau)||\pi_e(\tau)) \geq -\sum_{i\in\{0,u,e\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_e(\tau)}(\pi(\tau)||\pi_i(\tau)) \\
for \quad \beta_i^z, \beta_i^a \geq 1
\end{gathered} \tag{8.20}$$

Proof

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}(\pi(\tau) \parallel \pi_e(\tau)) &\leq \sum_{i\in\{0,u,e\}} \mathrm{D}_{\mathrm{KL}}(\pi(\tau) \parallel \pi_i(\tau)) \\
&\leq \sum_{i\in\{0,u,e\}} ub(\mathrm{D}_{\mathrm{KL}}(\pi(\tau) \parallel \pi_i(\tau))) \\
&= \sum_{i\in\{0,u,e\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{q(\tau)}(\pi(\tau)||\pi_i(\tau)) \quad for \quad \beta_i^z, \beta_i^a = 1 \\
&\leq \sum_{i\in\{0,u,e\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{q(\tau)}(\pi(\tau)||\pi_i(\tau)) \quad for \quad \beta_i^z, \beta_i^a \geq 1
\end{aligned} \tag{8.21}$$

The last line holds as each weighted term in $\tilde{\mathrm{D}}_{\mathrm{KL}}^{q(\tau)}(\pi(\tau)||\pi_i(\tau))$ are positive.

Reinforcement Learning Upper Bound

$$\mathbb{E}_{\substack{p(\mathscr{K}),\\ \pi_0(\tau)}}[\log(O=1|\tau,k)] \geq \mathbb{E}_{\pi_b(\tau)}\left[\sum_t \nu^{\pi_b}[t]\cdot r_k(\mathbf{x}_t,\mathbf{a}_t)\right] - \sum_{i\in\{0,u\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_b(\tau)}(\pi(\tau)||\pi_i(\tau)) \\ for \quad \beta_i^z,\beta_i^a \geq 1 \quad and \quad r_k < 0 \tag{8.22}$$

Proof

$$\begin{aligned}
\mathbb{E}_{\substack{p(\mathscr{K}),\\ \pi_0(\tau)}}[\log(O=1|\tau,k)] &\geq \mathbb{E}_{\pi_b(\tau)}\left[\sum_t \nu^{\pi_b}[t]\cdot r_k(\mathbf{x}_t,\mathbf{a}_t)\right] - \mathrm{D}_{\mathrm{KL}}\left(\pi(\tau)\parallel\pi_i(\tau)\right) \\
&\geq \mathbb{E}_{\pi_b(\tau)}\left[\sum_t \nu^{\pi_b}[t]\cdot r_k(\mathbf{x}_t,\mathbf{a}_t)\right] - \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_b(\tau)}(\pi(\tau)||\pi_i(\tau)),\ for\ \beta_i^z,\beta_i^a = 1 \\
&\geq \mathbb{E}_{\pi_b(\tau)}\left[\sum_t \nu^{\pi_b}[t]\cdot r_k(\mathbf{x}_t,\mathbf{a}_t)\right] - \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_b(\tau)}(\pi(\tau)||\pi_i(\tau)),\ for\ \beta_i^z,\beta_i^a \geq 1 \\
&\geq \mathbb{E}_{\pi_b(\tau)}\left[\sum_t \nu^{\pi_b}[t]\cdot r_k(\mathbf{x}_t,\mathbf{a}_t)\right] - \sum_{i\in\{0,u\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_b(\tau)}(\pi(\tau)||\pi_i(\tau)), \\
&for \quad \beta_i^z,\beta_i^a \geq 1
\end{aligned} \tag{8.23}$$

Refer to Abdolmaleki et al.[3] for line 1 proof. The final 2 lines hold due as KL(.) ≥ 0.

8.3.3 Environments

We continue by covering each environment setup in detail.

CorridorMaze

Intuitively, the agent starts at the intersection of corridors, at the origin, and must traverse corridors, aligned with each dimension of the state-space, in a given ordering. This requires the agent to reach the end of the corridor (which we call half-corridor cycle), and return back to the origin, before the corridor is considered complete.

The environment is defined by: $s \in \{0,l\}^c$, $p(s_0) = \mathbf{0}^c$, $k =$ one-hot task encoding, $a \in [0,1]$, $r_k^{\text{semi-sparse}}(\mathbf{x}_t,\mathbf{a}_t) = 1$ *if* agent has correctly completed the entire or half-corridor cycle *else* 0, $r_k^{\text{sparse}}(\mathbf{x}_t,\mathbf{a}_t) = 1$ *if* task complete *else* 0. Task is considered complete when a desired ordering of corridors have been traversed. $c = 5$ represents the number of corridors in our experiments. $l = 6$, the lengths of each corridor. States transition according to deterministic transition function $s_{t+1}^{j_t} = f(\mathbf{s}_t^{j_t},\mathbf{a}_t)$. j_t corresponds to the index of the current corridor that the agent is in (i.e. $j_t = 0$

if the agent is in corridor 0 at timestep t). $\mathbf{s}^i$ corresponds to the i^{th} dimension of the state. States transition incrementally or decrementally down a corridor, and given state dimension s^i, if actions fall into corresponding transition action bins ψ_{inc}, ψ_{dec}. We define the transition function as:

$$f(\mathbf{s}_t^{j_t}, \mathbf{a}_t) = \begin{cases} \mathbf{s}_t^{j_t} + 1, & \text{if } \psi_{inc}^w(\mathbf{a}_t, j_t). \\ \mathbf{s}_t^{j_t} - 1, & \text{elif } \psi_{dec}^w(\mathbf{a}_t, j_t). \\ 0, & \text{otherwise.} \end{cases} \tag{8.24}$$

$\psi_{inc}^w(\mathbf{a}_t, j) = \text{bool}(\mathbf{a}_t \text{ in } [j/c, (j+0.5\cdot w)/c])$, $\psi_{dec}^w(\mathbf{a}_t, j) = \text{bool}(\mathbf{a}_t \text{ in } [j/c, (2\cdot j - 0.5\cdot w)/c])$. The smaller the w parameter, the narrower the distribution of actions that lead to transitions. As such, w, together with r_k, control the exploration difficulty of task k. We set $w = 0.9$. We constrain the state transitions to not transition outside of the corridor boundaries. Furthermore, if the agent is at the origin, $s = \mathbf{0}^c$ (at the intersection of corridors), then the transition function is ran for all values of j_t, thereby allowing the agent to transition into any corridor.

Stack

This domain is adapted from the well known gym robotics FetchPickAndPlace-v0 environment[118]. The following modifications were made: 1) 3 additional blocks were introduced, with different colours, and a goal pad, 2) object spawn locations were not randomized and were instantiated equidistantly around the goal pad, see Figure 5.2, 3) the number of substeps was increased from 20 to 60, as this reduced episodic lengths, 4) a transparent hollow rectangular tube was placed around the goal pad, to simplify the stacking task and prevent stacked objects from collapsing due to structural instabilities, 5) the arm was always spawned over the goal pad, see figure Figure 5.2, 6) the state space corresponded to gripper position and grasp state, as well as the object positions and relative positions with respect to the arm. Velocities were omitted as access to such information may not be realistic for real robotic systems. $k =$ one-hot task encoding, $r_k^{\text{sparse}}(\mathbf{x}_t, \mathbf{a}_t) = 1 \quad if \quad$ correct object has been placed on stack in correct ordering $\quad else \quad 0$.

8.3.4 Experimental Setup

We provide the reader with the experimental setup for all training regimes and environments below. We build off the softlearning code base[46]. Algorithmic details not mentioned in the following sec-

Table 8.4: Feedforward Module, $\pi_{(0)}^{\{H,L\}}$

hidden layers	$(512, 512)$
hidden layer activation	relu
output activation	linear

tions are omitted as are kept constant with the original code base. For all experiments, we sample *batch size* number of **entire episodes** of experience during training.

Model Architectures

We continue by outlining the shared model architectures across domains and experiments. Each policy network (e.g. $\pi^H, \pi^L, \pi_0^H, \pi_0^L$) is comprised of a feedforward module outlined in Table 8.4. The softlearning repository that we build off[46], applies tanh activation over network outputs, where appropriate, to match the predefined outpute ranges of any given module. The critic is also comprised of the same feedforward module, but is not hierarchical. To handle historical inputs, we tile the inputs and flatten, to become one large input 1-dimensional array. We ensure the input always remains of fixed size by appropriately left padding zeros. For π^H, π_0^H we use a categorical latent space of size 10. We found this dimensionality sufficed for expressing the diverse behaviours exhibited in our domains. Table 8.5 describes the setup for all the experiments, including inputs to each module, level over which KL-regularisation occurs (z or a), which modules are shared (e.g. π^L and π_0^L), and which modules are reused across sequential tasks. For the covariate shift designed experiments in Table 5.1b, we additionally reuse π^H (or π^L for RecSAC) across domains, and whose input is $\mathbf{x}_t$. For all the experiments, any reused modules are not given access to task-dependent information, namely task-id (k) and exteroceptive information (cube locations for Stack domain). This design choice ensures reused modules generalise across task instances.

Behavioural Cloning

For the BC setup, we use a deterministic, noisy, expert controller to create experience to learn off. We apply DAGGER[129] during data collection and training of policy π as we found this aided at achieving a high success rate at the BC tasks. Our DAGGER setup intermittently during data collection, with a predefined rate, samples an action from π instead of π_e, but still saves BC target action a_t as the one that would have been taken by the expert for $\mathbf{x}_k$. This setup helps mitigate covariate shift

Table 8.5: Full experimental setup. Describes inputs to each module, level over which KL-regularisation occurs, which modules are shared and reused across training and transfer tasks.

Name	π_0^H	π_0^L	π^L	π^H	KL level	$\pi^L = \pi_0^L$	reused modules
APES-H20	$\mathbf{x}_{t-20:t}$	$\mathbf{s}_t$	$\mathbf{s}_t$	$\mathbf{x}_k$	z	✓	π_0^H, π_0^L, π^L
APES-H10	$\mathbf{x}_{t-10:t}$	$\mathbf{s}_t$	$\mathbf{s}_t$	$\mathbf{x}_k$	z	✓	π_0^H, π_0^L, π^L
APES-H1	$\mathbf{x}_{t-1:t}$	$\mathbf{s}_t$	$\mathbf{s}_t$	$\mathbf{x}_k$	z	✓	π_0^H, π_0^L, π^L
APES-S	$\mathbf{s}_t$	$\mathbf{s}_t$	$\mathbf{s}_t$	$\mathbf{x}_k$	z	✓	π_0^H, π_0^L, π^L
APES-no_prior	_	_	$\mathbf{s}_t$	$\mathbf{x}_k$	_	_	π^L
Hier-RecSAC	_	_	$\mathbf{s}_t$	$\mathbf{x}_k$	_	_	_
RecSAC	_	_	$\mathbf{x}_k$	_	_	_	_
APES-H1-KL-a	$\mathbf{x}_{t-1:t}$	$\mathbf{s}_t$	$\mathbf{s}_t$	$\mathbf{x}_k$	a	✗	π_0^H, π_0^L
APES-H1-flat	_	$\mathbf{x}_{t-1:t}$	$\mathbf{s}_t$	$\mathbf{x}_k$	a	✗	π_0^L

during training, between policy and expert. Noise levels were chosen to be small enough so that the expert still succeeded at the task. We trained our policies for one epoch (once over each collected data sample in the expectation). It may be possible to be more sample efficient, by increasing the ratio of gradient steps to data collection, but we did not explore this direction. The interplay we use between data collection and training over the collected experience, is akin to the RL paradigm. We build off the softlearning code base[46], so please refer to it for details regarding this interplay.

Table 8.6: Full Training Setup

(a) Behavioural Cloning Setup

Environment	CorridorMaze	Stack
$\pi_{(i)}$ learning rate	$3e^{-4}$	$3e^{-4}$
z categorical size	10	10
π^H history depth	24	5
β_u^z	$1e^{-3}$	$1e^{-3}$
β_u^a	$1e^{-2}$	$1e^{-2}$
β_e^a	1	1
β_0^z	1	1
β_0^a	1	1
DAGGER rate	0.1	0.1
batch size	128	128
episodic length	24	35

(b) Reinforcement Learning Setup

Environment	CorridorMaze		Stack
Reward Type	sparse	semi-sparse	sparse
Transfer task	2 corridor	4 corridor	4 blocks
Q learning rate	$3e^{-6}$	$3e^{-5}$	$3e^{-5}$
π learning rate	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$
$\beta_0^{z/a}$	$1e^{-2}$	$1e^{-1}$	$5e^{-2}$
$\beta_u^{z/a}$	$1e^{-2}$	0	$5e^{-4}$
Q update rate	$6e^{-4}$	$6e^{-4}$	$6e^{-4}$
Retrace λ	0.99	0.99	0.99
batch size	128	128	128
episodic length	30	60	65

Refer to Table 8.6a for BC algorithmic details. It is important to note here that, although we report five β hyper-parameter values, there are only two degrees of freedom. As we stop gradients flowing from π_0 to π, choice of β_0 is unimportant (as long as it is not 0) as it does not influence the interplay between gradients from individual loss terms. We set these values to 1. β_e^a's absolute value is also unimportant, and only its relative value compared to β_u^z and β_u^a matters. We also set β_e^a to 1. For the remaining two hyper-parameters, β_u^z, β_u^a, we performed a hyper-parameter sweep over three orders

of magnitude, three values across each dimension, to obtain the reported optimal values. In practice $\pi_0 = \{\pi_i\}_{i\in\{0,\dots,N\}}$, multiple trained priors each sharing the same β_0 hyper-parameters. Four seeds were ran, as for all experiments reported. We observed very small variation in learning across the seeds, and used the best performing seed to bootstrap learning off for all latter experiments. We separately also performed a hyper-parameter sweep over π learning rate, in the same way as before. We did not perform a sweep for batch size. We found for both BC and RL setups, that conditioning on entire history for π^H was not always necessary, and sometimes hurt performance. We report history lengths for π^H for BC in Table 8.6a. This value was also used for π^H and Q for the RL setup.

We prevent gradient flow from π_0 to π, to ensure as fair a comparison between ablations as possible: each prior distils knowledge from the same, high performing, policy π and dataset. If we simultaneously trained multiple π and π_0 pairs (for each distinct prior), it is possible that different learnt priors would influence the quality of each policy π which knowledge is distilled off. In this analysis, we are not interested in investigating how priors affect π during BC, but instead how priors influence what knowledge can be distilled and transferred. We observed prior KL-distillation loss convergence across tasks and seeds, ensuring a fair comparison.

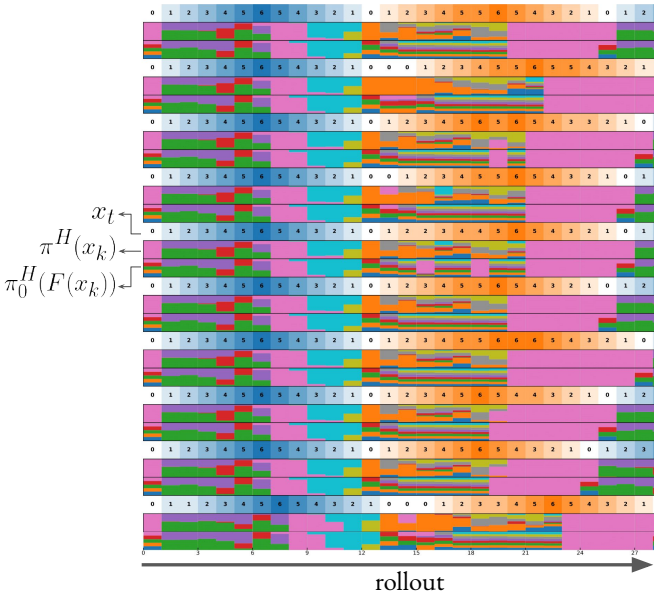
Reinforcement Learning

During this stage of training we freeze the prior and low-level policy (if applicable, depending on the ablation). In general, any reused modules across sequential tasks are frozen (apart from π^H for the covariate shift experiments in Table 5.1b). Any modules that are not shared (such as π^H for most experiments) are initialised randomly across tasks. The RL setup is akin to the softlearning repository[46] that we build off. We note any changes in Table 8.6b. We regularise against the latent or action level, depending on the ablation, whether or not our models are hierarchical, share low level policies, or use pre-trained modules (low-level policy and prior). Therefore, we only ever regularise against, at most, two β hyper-parameters. Hyper-parameter sweeps are performed in the same way as in the BC experiments. We did not sweep over *Retrace* λ, *batch size*, or *episodic length*. For Retrace, we clip the importance weights between $[0, 1]$, like in the original paper[98], and perform λ returns rather than n-step. We found the Retrace operator important for sample-efficient learning.

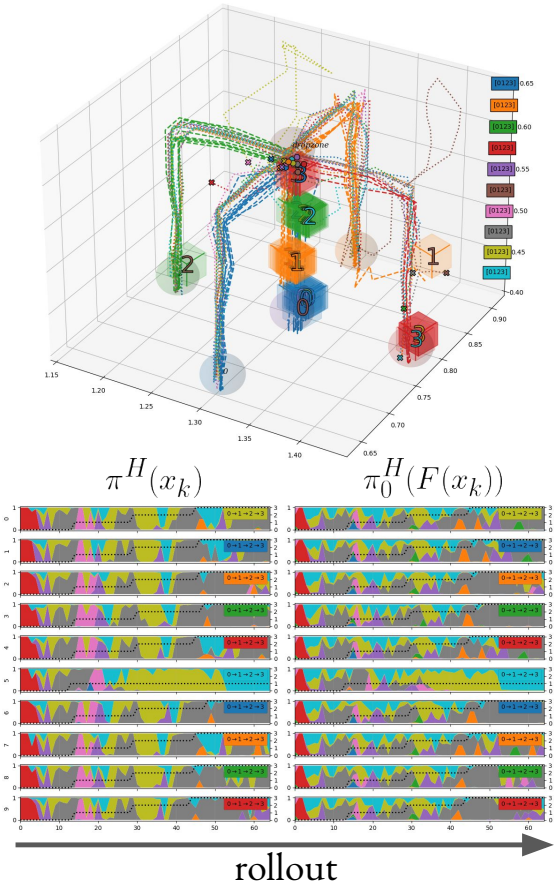
8.3.5 Final Policy Rollouts and Analysis

In this section, we show final policy performance (in terms of episodic rollouts) for the most performant method, *APES-H1*, across each transfer domain. We additionally display the categorical probability distributions for $\pi^H(\mathbf{x}_k)$ and $\pi_0^H(F(\mathbf{x}_k))$ across each rollout to analyse the behaviour of each. $F(.)$ denotes the chosen *information gating function* for the prior. For *CorridorMaze 2 and 4 corridor*, seen in Figures 8.8a and 8.9, we see that the full method successfully solves each respective task, traversing the correct ordering of corridors. The categorical distributions for these domains remain relatively entropic. In general, latent categories cluster into those that lead the agent deeper down a corridor, and those that return the agent to the hallway. Policy and prior align their categorical distributions in general, as expected. Interestingly, however, the two categorical distributions deviate the most from each other at the hallway, the *bottleneck state*[156], where prior multimodality (for hierarchical π_0) exists most (e.g. which corridor to traverse next). In this setting, the policy needs to deviate from the multimodal prior, and traverse only the optimal next corridor. We also observe, for the hallway, that the prior allocates one category to each of the five corridors. Such behaviour would not be possible with a flat prior.

Figure 8.8b plots the same information for *Stack 4 blocks*. APES-H1 successfully solves the transfer task, stacking all blocks according to their masses. Similar categorical latent-space trends exist for this domain as the previous. Most noteworthy is the behaviour of both policy and prior at the *bottleneck state*, the location above the block stack, where blocks are placed. This location is visited five times within the episode: at the start $\mathbf{s}_0$, and four more times upon each stacked block. Interestingly, for this state, the prior becomes increasingly less entropic upon each successive visit. This suggests that the prior has learnt that the number of feasible *high-level* actions (corresponding to which block to stack next), reduces upon each visit, as there remains fewer lighter blocks to stack. It is also interesting that for $\mathbf{s}_0$, the red categorical value is more favoured than the rest. Here, the red categorical value corresponds to moving towards cube 0, the heaviest cube. This behaviour is as expected, as during BC, this cube was stacked first more often than the others, given its mass. For this domain, akin to *CorridorMaze*, the policy deviates most from the prior at the bottleneck state, as here it needs to behave deterministically (regrading which block to stack next).



(a) *CorridorMaze 2 corridor.* 10 final policy rollouts (episodes vertically, rollouts horizontally) for most performant method. Task: Blue -> Orange corridors. Policy distribution over categorical latent space for π^H and π_0^H plotted (between policy rollouts, denoted by x_t) as vertical histograms, with colour and width denoting category and probability. Colour here does not correlate to corridor.



(b) *Stack 4 blocks. Top)* Policy rollouts for most performant method. Task: stack blocks in order 0 -> 1 -> 2 -> 3. *Bottom)* policy distributions akin to Figure 8.8a. Horizontal dashed lines in bottom plots refer to current sub-task (block stack), vertically transitioning upon each completion.

Figure 8.8: Final transfer performance for most performant method. Tasks are solved. Displaying latent distribution for both policy and prior. For both domains, policy deviates most from prior at the bottleneck state (hallway/stacking zone, for CorridorMaze/Stack), where prior multimodality exists (e.g. which corridor/block to tackle next), but where determinism is required for the task at hand.

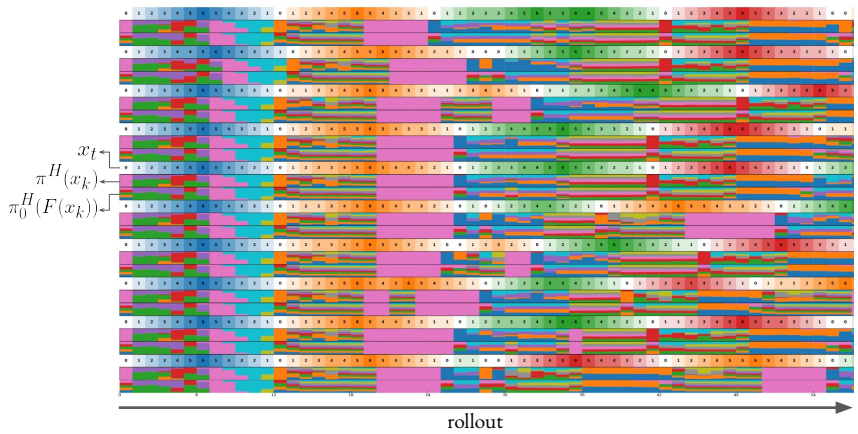


Figure 8.9: *CorridorMaze 4 corridor.* 10 final policy rollouts (episodes vertically, rollouts horizontally) for most performant method. Task: Blue -> Orange -> Green -> Red corridors. Policy distribution over categorical latent space for π^H and π_0^H plotted (between policy rollouts, denoted by x_t) as vertical histograms, with colour and width denoting category and probability.

8.4 Chapter 6 Additional Details

We provide the reader with experimental details required to reproduce the results in Chapter 6.

8.4.1 Architecture

Table 8.7: Feedforward Module, $\pi^{\{C,L\}}, \beta, \tilde{p}^T, Q$

hidden layers	$(256, 256)$
hidden layer activation	elu
output activation	linear

We continue by outlining the model architectures across domains and experiments. Each policy network (π^C, π^L, β) and the option-transition model $\tilde{p}^T$ are comprised of a feedforward module outlined in Table 8.7. π^C then has a component layer of size 128 for each option. We apply a tanh or softmax activations over network outputs, where appropriate, to match the predefined output ranges of any given module. The critic, Q, is also a feedforward module. For π^C we use a categorical latent space of size 4 (number of options). We find this dimensionality suffices for expressing the diverse behaviours in our domains. $\tilde{p}^T$ is also a gaussian mixture model with 4 heads. On the transfer experiments we freeze π^L, β and train a newly instantiated π^C to recompose prior skills. $\tilde{p}^T$ is not used during online transfer.

8.4.2 Offline Skill Discovery

Table 8.8: Offline Skill Discovery Setup

Environment	Maze2D	AntMaze
$\pi^{\{C,L\}}, \beta$ learning rate	$3e^{-4}$	$3e^{-4}$
$\tilde{p}^T$ learning rate	$3e^{-4}$	$3e^{-4}$
number of options	4	4
α predictability weight	1.0	0.2
$m \log(\tilde{p}^T)$ offset	13	103
batch size	256	128
sequence length	100	100

In Equation (6.5) we introduce MO2's objective O_{mo2} for discovering bottleneck options. In practice, we actually use the following slightly modified objective:

$$\max_{\pi^L,\pi^C,\beta} O_{mo2}(\pi^L,\pi^C,\beta) = \mathbb{E}_{\substack{s_t,a_t,h_t\sim D \\ o\sim\pi^H(\cdot|h_t),a\sim\pi^L(\cdot|s_t,o) \\ s_f\sim\tilde{p}^T(\cdot|s_t,o,a)}} [\alpha\beta(s_t|h_t)\log(\tilde{p}^T(s_f|s_t,o,a)-m)+\log(\pi(a_t,o|h_t))] \tag{8.25}$$

m and α are hyperparameters. m is set to the maximum possible $\tilde{p}^T(s_f|s_t,o,a)$ value and is a function of the minimum permitted covariance Σ of our model $\tilde{p}^T$ (σ set to $1e^{-3}$ for $\Sigma=\sigma\mathbb{I}$). As such, m ensures O_{pred} is always negative, thus encouraging minimal switches that align with bottleneck states, as outlined in Figure 6.2. α is set using grid search (specifically $[1e^{-1},2e^{-1},4e^{-1},6e^{-1},8e^{-1},1e^{0}]$), and weighs the importance of the two objects O_{pred} and O_{bc}. The offline data collected by behavioural policy, $b(a|h)$, consists of tuples of experience in the form (s_t,a_t).

We train the options framework ($\pi^{\{C,L\}},\beta$) and option-transition model $\tilde{p}^T$ in parallel using Equation (8.25) and Equation (6.6) respectively. Algorithmic hyperparameter details are outlined in Table 8.8. We run four seeds. During transfer, we select the seed that traverses the maze deepest. We note, however, that there is minimal difference between each seed. *Batch size* refers to the number of independently sampled trajectory snippets from the offline dataset, each of length *sequence length*. We sample these batches uniformly, according to Wulfmeier et al.[177]. We run experiments until O_{mo2} convergence ($1e^6$ gradients). For the baselines, *HO2 (offline), HO2-lim (offline)*, we set α to zero, yet still train $\tilde{p}^T$ for evaluation purposes. The other hyperparameters are kept constant. Finally, for *HO2-lim (offline)*, we run a sweep over N (specifically $[1,5,10]$), the number of permitted option switches, when training on Equation (17) from the Appendix of Wulfmeier et al.[177]. We report results for $N=1$, as this is the experiment that achieves the lowest switch rate.

8.4.3 Online Transfer Learning

During this stage of training we freeze the option-policies (π^L,β). The categorical option controller, π^C is initialised randomly on the transfer task (inline with Wulfmeier et al.[177]) and is trained to reorder options to solve the task. We note that while some skill transfer approaches[134;115] note performance gains by additionally transferring π^C, we did not observe this. The RL setup is akin to the CMPO[102]. We note any changes in Table 8.9. Hyperparameter sweeps are performed for *data collection to gradients ratio* (data-grad ratio) (specifically $[5e^1,5e^2,5e^3]$) for each method as we find

Table 8.9: Online Transfer Learning Setup

Environment	Maze2D	AntMaze
Q learning rate	$1e^{-4}$	$1e^{-4}$
π^C learning rate	$1e^{-4}$	$1e^{-4}$
number options	4	4
ε	1	1
ε_{KL}	1	1
data-grad ratio	5000	50
batch size	256	256
sequence length	10	10

this makes a significant difference. We report the most performant setting for each approach. For HO2-scratch, we train an HO2 agent from scratch, using the same architecture as the other methods, and use the default setting/hyperparameters in Wulfmeier et al.[177].

MDP learning

For this setup, the agent acts in the environment at the option-level. Learning occurs at the original MDP level, however. The following, per time-step, tuple of experiences are stored in the replay-buffer: $\{s_t, o_t, r_t, s_{t+1}\}$. We use these samples to train the critic and policy, using CMPO. We perform temporal-difference TD(0) learning[155] for training the critic. Crucially, this occurs per time-step, at the original temporal resolution of the MDP. As such, the option's temporal abstractions are not used to perform more efficient TD learning across long horizons.

Semi-MDP learning

For this setup, the agent acts in the environment at the option-level. Learning also occurs at the option-level (over the option-induced semi-MDP). The following tuple of experiences are stored in the replay-buffer: $\{s_t, o_t, g_t, s_{t+k}\}$, where k is specified by the option's sampled termination condition in the environment. s_t represents the option's o_t initiation state, s_{t+k} the termination state, and $g_t = \sum_{i=t}^{t+k} \gamma^{i-t} r_i$ the discounted cumulative rewards during its execution. We use these samples to train the critic and policy, using CMPO. We perform temporal-difference TD(0) learning[155] for training the critic. Crucially, this occurs at the option-level, over the semi-MDP. As such, the option's temporal abstractions are used to perform more efficient TD learning across long horizons.

8.4.4 Environments

In this section, we highlight the modifications made to the standard D4RL[35] domains for our transfer learning experiments.

Maze2D

The original Maze2D transfer domain spawns the agent (at the start of the episode) uniformly across the maze whilst keeping the target goal location static. The agent is rewarded upon goal reaching. The agent sometimes spawns close to the goal, other times far away. As such, credit-assignment and exploration are not particularly problematic as the spawn locations provide a form of curriculum for learning values and exploring the environment across all starting locations in the maze. Therefore, this domain does not necessitate temporal-abstractions for acting and learning. To increase the difficulty, we modify the environment such that the agent always spawns at the furthest distance from the goal (with respect to steps required to reach target). Now, abstractions are more important for credit-assignment and exploration. Nevertheless, the state-action space for this domain is low dimensional, and the episodic horizon is not long, meaning the task is still not particularly sparse. Additionally, to keep with the modular task setting (as detailed in Section 6.1), we terminate the episode early when the goal is reached. Below we detail the maze layout for our setting:

```
// Modified Maze2D
// R = spawn location, G = goal location, 1 = walls, 0 = empty space

Maze2D = [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
          [1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1],
          [1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1],
          [1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1],
          [1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1],
          [1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1],
          [1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1],
          [1, R, 0, 1, 0, 0, 0, 1, 0, 0, G, 1],
          [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]
```

AntMaze

For this domain, we experimentally find that reaching the original target location is tricky for all methods. While MO2 strongly outperforms all other approaches, its success rate is still low (within the allocated online training budget). We suspect this may be an issue with the number of offline trajectories reaching the goal location. All our methods discover skills by maximising the log-likelihood of offline data. If the goal is rarely reached offline, neither method will favour discovering skills that reach it. As such, we change the goal location to a slightly closer location, more representative of offline behaviours (which we visually inspected). We used the following maze layout:

```
// Modified AntMaze
// R = spawn location, G = goal location, 1 = walls, 0 = empty space

AntMaze = [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
           [1, R, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1],
           [1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1],
           [1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1],
           [1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1],
           [1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1],
           [1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1],
           [1, 0, 0, 1, 0, 0, G, 1, 0, 0, 0, 1],
           [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]
```

8.4.5 Proofs and Discussions

We continue by highlighting why MO2's objective leads to bottleneck state discovery. We provide proofs coupled with explanations.

Why does Equation (6.3) lead to bottleneck options?

We continue by demonstrating why Equation (6.3) leads to the minimal number of low entropic, transition-state distributions, able to reconstruct offline behaviours. These are the properties of bottleneck states for modular MDPs (see the *temporal optimal compression* scenario in Figure 6.2). To achieve this we first demonstrate why maximising Equation (6.3) is equivalent to minimising the

conditional entropy of plans. $\mathbb{H}_\pi[s_{0:N}|o_{0:N},a_{0:N}]$ represents the conditional entropy of plans over sequential option-transition states $s_{0:N}$ conditioned on sequential options $o_{0:N}$ and their corresponding first initiated action $a_{0:N}$.

$$\begin{aligned}
\mathbb{H}_\pi[s_{0:N}|o_{0:N},a_{0:N}] &= -\mathbb{E}_{s_{0:N},o_{0:N},a_{0:N}\sim\pi}[\log(p_\pi(s_{0:N}|o_{0:N},a_{0:N}))] \\
&= -\mathbb{E}_{s_{0:N},o_{0:N},a_{0:N}\sim\pi}[\prod_{n=0}^{N}\log(p^T(s_{n+1}|s_n,o_n,a_n))] \\
&= -\sum_{n=0}^{N}\mathbb{E}_{s_{0:N},o_{0:N},a_{0:N}\sim\pi}[\log(p^T(s_{n+1}|s_n,o_n,a_n))] \\
&= -\sum_{n=0}^{N}\mathbb{E}_{s_n,o_n,a_n,s_{n+1}\sim\pi}[\log(p^T(s_{n+1}|s_n,o_n,a_n))] \\
&= \sum_{n=0}^{N}\mathbb{H}_\pi[s_{n+1}|s_n,o_n,a_n] \\
&= -O_{pred} \text{ (Equation (6.3))}
\end{aligned} \tag{8.26}$$

We subscript $\mathbb{H}$ with π to emphasise that this entropy (and plans) is dependent on our option-policy π that defines the option-transition model $p^T(s_{n+1}|s_n,o_n,a_n)$ and how options and actions are chosen (from $\pi^C(o|s)$ and $\pi^L(a|s,o)$). $s_{0:N},o_{0:N},a_{0:N}\sim\pi$ represents that the expectation over sequential transition states, options and actions are sampled from our policy π acting in the environment (i.e. from $p_\pi(s_{0:N},o_{0:N},a_{0:N})$).

The second line of Equation (8.26) holds due to the product rule and independence between s_{n+1} and all other variables after conditioning on s_n,o_n,a_n, given our graphical model of the agent π and environment. The fourth line holds due to the integral of all extraneous variables, that p^T is not dependent on, integrating to 1. The final line is equivalent to the fourth by the definition of Equation (6.3).

We therefore demonstrate that maximising the O_{pred} objective in Equation (6.3) is equivalent to minimising the conditional entropy of plans over sequential option-transition states $\mathbb{H}_\pi[s_{0:N}|o_{0:N},a_{0:N}]$ as well as the cumulative sum of individual conditional option-transition entropies $\sum_{n=0}^{N}\mathbb{H}_\pi[s_{n+1}|s_n,o_n,a_n]$. If individual entropies are enforced to be positive (achievable as shown in Equation (8.25)) then to minimise Equation (8.26) transitions should be as sparse as possible as to minimise the accumulation of positive option-transition entropies.

If we assume for now that $\pi(a|h)=b(a|h)$ (with b representing the behavioural distribution that

created the offline data), and constant environment dynamics, then the distribution over trajectories induced by π equates to the offline data distribution over the *source* domains. As such, Equation (8.26) encourages a decomposition of offline behaviours into compressed temporal-abstractions between sequential decision states able to reconstruct offline behaviours by their recomposition (using options) with high confidence (low $\mathbb{H}_\pi[s_{0:N}|o_{0:N}, a_{0:N}]$). Additionally, individual option-transitions entropies $\mathbb{H}_\pi[s_{n+1}|s_n, o_n, a_n]$ should be low, thus aligning with bottleneck states as depicted in Figure 6.2. Altogether, Equation (6.3) leads to the *temporal optimal compression (bottleneck state alignment)* scenario in Figure 6.2.

In practice, $\pi(a|h) \neq b(a|h)$. However, the O_{bc} objective in Equations (6.5) and (8.25) encourages both to align with each other. The α hyperparameter in Equation (8.25) is set low enough such that the O_{pred} component of MO2's objective does not hinder π's ability to align with b. Thus, over time, as π distils the offline behaviours, Equations (6.5) and (8.25) will discover bottleneck options able to reconstruct the offline modular behaviours across the *source* MDPs.

Under what assumptions does optimising O_{pred} from Equation (6.3) = Equation (6.4)?

Equations (6.5) and (8.25) do not use the form of O_{pred} from Equation (6.3) but instead the alternate definition in Equation (6.4). We continue by highlighting under what assumptions optimising either form is equivalent.

Our final proof relies on two concepts. Firstly, instead of considering individual n^{th} option initiation state distributions $p_\pi(s_n)$ across the episode, we consider the stationary option initiation state distribution $p_\pi(s_i)$ by marginalising over the transition number dimension n:

$$p_\pi(s_i) = E_{n\sim\pi}[p_\pi(s_n)] \tag{8.27}$$

$p_\pi(s_i)$ denotes the marginal initiation state distribution, independent of transition number n. The distribution over n is dependent on π which is why $n \sim \pi$ in the expectation. The second concept that we rely on is how to obtain the same marginal initiation state distribution by marginalising over the t rather than n. As highlighted in Section 6.3.1, we have a closed form solution to a related quantity $\beta(s_t|h_t)$ that represents the probability that s_t is initial given history h_t. Given this quantity we can

calculate $p_\pi(s_i)$ by marginalising across t and h_t as follows:

$$p_\pi(s_i) = E_{t\sim U\{0,T\},h_t\sim\pi}[\beta(s_t|h_t)p_\pi(s_t|h_t)] \tag{8.28}$$

$U\{0,T\}$ denotes a uniform categorical distribution between 0 and T, the episodic length. $p_\pi(s_t|h_t)$ denotes the probability state distribution at time-step t given history h_t, for policy π. Equation (8.28) without the $\beta(s_t|h_t)$ term corresponds to the stationary state visitation distribution of policy π (e.g. $p_\pi(s) = E_{t\sim U\{0,T\},h_t\sim\pi}[p_\pi(s_t|h_t)]$ which is independent of t). The additional weights weigh how probable any s_t is given h_t according to our option-policy π. We continue by using Equations (8.27) and (8.28) to show to relation between Equations (6.3) and (6.4):

$$\begin{aligned}
O_{pred}\text{ (Equation (6.3))} &= \sum_{n=0}^{N} \mathbb{E}_{s_n,o_n,a_n,s_{n+1}\sim\pi}[\log(p^T(s_{n+1}|s_n,o_n,a_n))] \\
&= N\mathbb{E}_{\substack{s_i\sim\pi \\ o\sim\pi^C(\cdot|s_i),a\sim\pi^L(\cdot|s_i,o) \\ s_f\sim p^T(\cdot|s_i,o,a)}}[\log(p^T(s_f|s_i,o,a))] \\
&\propto \mathbb{E}_{\substack{s_i\sim\pi \\ o\sim\pi^C(\cdot|s_i),a\sim\pi^L(\cdot|s_i,o) \\ s_f\sim p^T(\cdot|s_i,o,a)}}[\log(p^T(s_f|s_i,o,a))] \\
&= \mathbb{E}_{\substack{t\sim U\{0,T\},s_t,h_t\sim\pi \\ o\sim\pi^H(\cdot|h_t),a\sim\pi^L(\cdot|s_t,o) \\ s_f\sim p^T(\cdot|s_t,o,a)}}[\beta(s_t|h_t)\log(p^T(s_f|s_t,o,a))] \\
&= \mathbb{E}_{\substack{s_t,h_t\sim D \\ o\sim\pi^H(\cdot|h_t),a\sim\pi^L(\cdot|s_t,o) \\ s_f\sim p^T(\cdot|s_t,o,a)}}[\beta(s_t|h_t)\log(p^T(s_f|s_t,o,a))] \text{ if } \pi(a|h) = b(a|h) \\
&:= O_{pred}\text{ (Equation (6.4))}
\end{aligned} \tag{8.29}$$

The second line of Equation (8.29) is equivalent to the first by marginalising over the depth dimension n, akin to Equation (8.27), and then grouping into marginal initial transition state distribution $s_i \sim \pi$ of options and their conditional terminal state distribution, $s_f \sim p^T(\cdot|s_i,o,a)$. $s_i \sim \pi$ is shorthand for sampling from $p_\pi(s_i)$. In the expectation, we explicitly report how options and actions are sampled. N represents the expected number of transitions across episodes. The third line is proportional to the second by a factor of N. The fourth line is obtained by applying Equation (8.28) and grouping $p_\pi(s_t|h_t)$ into the expectation (i.e. $s_t \sim \pi$).

In practice, we do not have access to s_t and h_t samples from option-policy π and thus cannot calculate

the expectation in the fourth line. Nevertheless, if we assume $\pi(a|h) = b(a|h)$, then we can use offline data samples (from D) instead. As such, we obtain the fifth line from Equation (8.29). We omit $t \sim U\{0,T\}$ for simplicity and to keep notation consistent with Section 6.3.1. However, we are still sampling t as aforementioned. The final line holds by definition. We have therefore proved that Equation (6.3) $\propto$ Equation (6.4) under the assumption that $\pi(a|h) = b(a|h)$. Proportionality does not influence optimisation and therefore both are equivalent from a learning perspective. As mentioned in Section 8.4.5, $\pi(a|h) \neq b(a|h)$. However, O_{bc} encourages both to align over time, ensuring O_{pred} leads to bottleneck options.