Priors, Hierarchy, and Information Asymmetry for Skill Transfer in Reinforcement Learning

Sasha Salter*, **Kristian Hartikainen***, **Walter Goodwin**, **Ingmar Posner**
Applied AI Lab, University of Oxford,
{sasha, kristian, walter, ingmar}@robots.ox.ac.uk

Abstract:
The ability to discover behaviours from past experience and transfer them to new tasks is a hallmark of intelligent agents acting sample-efficiently in the real world. Equipping embodied reinforcement learners with the same ability may be crucial for their successful deployment in robotics. While hierarchical and KL-regularized RL individually hold promise here, arguably a hybrid approach could combine their respective benefits. Key to these fields is the use of information asymmetry to bias which skills are learnt. While asymmetric choice has a large influence on transferability, prior works have explored a narrow range of asymmetries, primarily motivated by intuition. In this paper, we theoretically and empirically show the crucial trade-off, controlled by information asymmetry, between the *expressivity* and *transferability* of skills across sequential tasks. Given this insight, we provide a principled approach towards choosing asymmetry and apply our approach to a complex, robotic block stacking domain, unsolvable by baselines, demonstrating the effectiveness of hierarchical KL-regularized RL, coupled with correct asymmetric choice, for sample-efficient transfer learning.

1 Introduction

While Reinforcement Learning (RL) algorithms recently achieved impressive feats across a range of domains [1, 2, 3], they remain sample inefficient [4, 5] and are therefore of limited use for real-world robotics applications. Intelligent agents during their lifetime discover and reuse skills at multiple 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 lifelong learning RL agents [6] with a similar ability could be vital towards attaining comparable sample efficiency.

To this end, two paradigms have recently been introduced. *KL-regularized* RL [7, 8] presents an intuitive approach for automating skill reuse in multi-task learning. By regularizing 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 skill discovery [9, 10, 11, 12, 13] 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 skills that are transferable across tasks. Both hierarchy and priors offer their own skill abstraction. However, when combined, one can in theory discover and leverage multiple skill abstractions. Whilst prior works [14] 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 more generally refers to an asymmetric masking of information across architectural modules. This masking forces independence to, and ideally generalisation across, the masked dimensions [8]. Therefore, IA crucially biases learnt behaviour and how it transfers across environments. Previous works have motivated their chosen IAs primarily on intuition [14, 8, 9], which, if sub-optimal, limits transfer benefits. We demonstrate that this indeed is the case for the hierarchical KL-regularized method in [14]. A more systematic approach to identifying IA schemes, underpinned by rigorous theory, is therefore required in order to benefit from skills in lifelong learning.

*Co-first authors.

In this paper, we employ hierarchical KL-regularized RL [14] to effectively transfer skills 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 skills across sequential tasks. With this insight, we consider previously unexplored, distinct asymmetries 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, a complex, sparse-reward domain unsolvable by the state-of-the-art baselines, and show that with the right choice of asymmetries, sample-efficient learning can be achieved.

2 Reinforcement Learning for Skill Transfer

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

2.1 KL-Regularized Reinforcement Learning

The typical multi-task KL-regularized RL objective [15, 16, 17, 18] takes the form:

$$\mathcal{O}(\pi, \pi_0) = \mathbb{E}_{\substack{\tau \sim \rho_\pi, \\ k \sim p(\mathcal{K})}} \left[\sum_{t=0}^{\infty} \gamma^t \Big(r_k(\mathbf{x}_t, \mathbf{a}_t) - \alpha_0 \mathrm{D}_{\mathrm{KL}} \left(\pi(\mathbf{a}|\mathbf{x}_t, k) \parallel \pi_0(\mathbf{a}|\mathbf{x}_t) \right) \Big) \right] \tag{1}$$

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 from policy π 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 [7]. We consider the sequential learning paradigm, where skills are learnt from past tasks, $p_{past}(\mathcal{K})$, and leveraged while attempting the current set of tasks, $p_{current}(\mathcal{K})$.

2.2 Hierarchical KL-Regularized RL

While KL-regularized RL has achieved success across various settings [4, 7, 19, 12], recently Tirumala et al. [14] 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 skills relevant to each task. By not conditioning the high-level prior on task-id, Tirumala et al. [14] encourage the reuse of common high-level abstractions across tasks. Tirumala et al. [14] propose the following upper bound for approximating the KL-divergence between hierarchical policy and prior:

$$\mathrm{D}_{\mathrm{KL}} \left(\pi(\mathbf{a}|\mathbf{x}) \parallel \pi_0(\mathbf{a}|\mathbf{x}) \right) \leq \mathrm{D}_{\mathrm{KL}} \left(\pi^H(\mathbf{z}|\mathbf{x}) \parallel \pi_0^H(\mathbf{z}|\mathbf{x}) \right) + \mathbb{E}_{\pi^H} \left[\mathrm{D}_{\mathrm{KL}} \left(\pi^L(\mathbf{a}|\mathbf{x}, \mathbf{z}) \parallel \pi_0^L(\mathbf{a}|\mathbf{x}, \mathbf{z}) \right) \right] \tag{2}$$

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. [14] do not observe benefits from learning it.

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 in-

formation accessible by certain modules. Not conditioning on specific environment aspects forces independence and generalisation across them [8]. In the context of hierarchical KL-regularized RL, the explored asymmetries between the high-level policy, π^H, and prior, π_0^H, have been narrow [14, 19]. Tirumala et al. [14], Pertsch et al. [19] explore auto-regressive 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.

3 Model Architecture and the Expressivity-Transferability Trade-Off

To rigorously investigate the contribution of priors, hierarchy, and information asymmetry on skill transfer, 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 Fig. 1, which introduces *information gating functions* (IGFs) as a means of decoupling IA from architecture. Each component has its own IGF, depicted with colored rectangle. 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 skills they learn. By presenting multiple priors, we enable a comparison with existing literature [14, 19, 20, 21]. With the right masking, one can recover previously investigated asymmetries [14, 19], explore additional ones, and also express purely hierarchical [9] and KL-regularized equivalents [8].

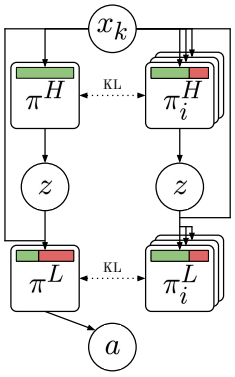


Figure 1: Hierarchical KL-regularized architecture. The hierarchical policy modules π^H and π^L are regularized against their corresponding prior modules π_i^H and π_i^L. The inputs to each module are filtered by an information gating function, depicted with colored rectangle.

While prior works focused on the role of IA on policy (π) regularization for multi-task learning [8], we focus on its influence over the prior's (π_0) ability to handle covariate shift across sequential tasks. 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:

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 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 Appendix B.1. □

In our case, p and q can be interpreted as training and transfer distributions over network inputs (such as π_0^H). Intuitively, Theorem 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. Therefore, IA leads to an impactful trade-off between the *transferability* and *expressivity* of skills. Interestingly, covariate shift is upper-bounded by policy shift: $\mathrm{D_{KL}}\left(p_{\pi_I}(\tau) \parallel p_{\pi_T}(\tau)\right) \geq \mathrm{D_{KL}}\left(p_{\pi_I}(\tau_f) \parallel p_{\pi_T}(\tau_f)\right)$ (using Theorem 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 attained skills.

While IAs influence *what* skills are discovered by hierarchy and priors, an important distinction must be made between *how* each approach *transfers* skills. Hierarchy transfers skills as *hard constraints* on transfer behaviour, by enforcing the reuse of low-level skills (π^L). In contrast, priors transfer skills through *soft constraints* (via KL-regularization), allowing the policy to deviate from these behaviours 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 [14, 9], 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 skill 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 is thus not expected to harm transferability. See Appendix D for further architectural details.

4 Method

As described in Section 3, sequential learning poses distinct challenges: primarily abrupt covariate shifts. As such, we designed experiments to isolate each component's importance under this setting.

4.1 Training Regime

In Tirumala et al. [14, 22], for each choice of IA, they first jointly train separate hierarchical policies and priors in a multi-task setting, and then freeze the shared low-level policy and high-level prior when training on the sequential task. Accordingly, they cannot isolate the effects of distinct IAs on sequential transfer, as distinct IAs lead to distinct low-level policies during multi-task learning. To decouple the effects of IA and hierarchy for sequential transfer, we propose the following regime. **Stage 1** trains a single hierarchical policy π in the multi-task setup whilst regularizing against *multiple* high-level priors with distinct IGFs. Importantly, unlike Tirumala et al. [14], 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 sequential task. See Algorithm 1. Unlike Tirumala et al. [14, 22], during stage 1 we train using variational behavioral cloning (BC) to bypass initial exploration issues. Given we focus on sequential transfer, this choice is unimportant. Nevertheless, our skill learning method can be considered of interest to the sequential learning community. During BC, we apply DAGGER [23], as per Algorithm 2, improving learning rates [24]. For further details refer to Appendix D.

4.2 Variational Behavioral Cloning as KL-Regularized Reinforcement Learning

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

$$\mathcal{O}_{bc}(\pi, \pi_e) = -\mathbb{E}_{\substack{\tau\sim\rho_\pi, \\ k\sim p(\mathcal{K})}} \left[\sum_{t=0}^{\infty} \gamma^t \alpha_e \mathrm{D}_{\mathrm{KL}} \left(\pi(\mathbf{a}|\mathbf{x}_t, k) \parallel \pi_e(\mathbf{a}|\mathbf{x}_t) \right) \right] \tag{3}$$

More generally, Eqs. (1) and (3) can be extended to the setting of multiple priors:

$$\mathcal{O}_{bc}(\pi, \{\pi_i\}^{i\in I}) = \sum_{i\in I} \mathcal{O}_{bc}(\pi, \pi_i) \quad (4) \qquad \mathcal{O}_{rl}(\pi, \{\pi_i\}^{i\subset I}) = E_\pi[R(\tau)] + O_{bc}(\pi, \{\pi_i\}^{i\subset I}) \quad (5)$$

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, $i \in \{0, u\}$, 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 Eq. (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 highly-entropic policies, encouraging exploration and stabilizing hierarchical learning [20]. We use Retrace [25] and double Q-learning [26] to train our critic. Refer to Appendices A and D for full training details and a deeper discussion on variational BC and RL.

5 Experiments

Our experiments are designed to answer the following questions related to sequential task learning: **(1)** Can we benefit from hierarchy and priors to effectively transfer knowledge over multiple levels

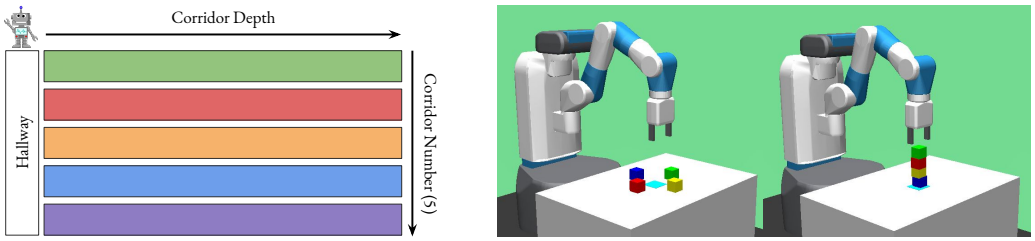


Figure 2: *Environments*. *a) CorridorMaze:* A hard-exploration toy domain. The agent starts in the hallway and must traverse a given number of corridors in a given sequence. The agent completes a given corridor by traversing to its depth and back. *b) Stack:* A hard exploration, robotic, domain. The agent must stack the cubes in a given ordering over the light blue target pad.

of action and temporal abstractions? **(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 in 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?

5.1 Environments

We explore these questions in 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 Fig. 2). Both of these domains exhibit modular behaviours at multiple-levels of abstraction whose discovery could yield transfer benefits for downstream tasks. Refer to Appendix C 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. 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*, rewarding on task completion. Our two 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 skill priors corresponding to sequentially stacking block with respect to their masses, is beneficial. We collect $17.5k$ trajectories from a scripted policy, stacking any two blocks given this requirement. To demonstrate the ability to achieve *generalisable knowledge transfer* and discover sequential coordination behavioural priors, the transfer task requires all blocks be stacked according to mass. Rewards are given per correct individual block stacked.

5.2 Hierarchy and Priors For Knowledge Transfer

We begin by exploring the relative importance of priors and hierarchy for skill 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. [14]. *RecSAC* represents a history-dependent *SAC* [27], 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. [9]. Table 3a 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 skill 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 semi-sparse domain, the smaller the observed benefits, given that rewards are enough to guide the learning. We provide further qualitative analysis of the performance of these methods in Section 5.6.

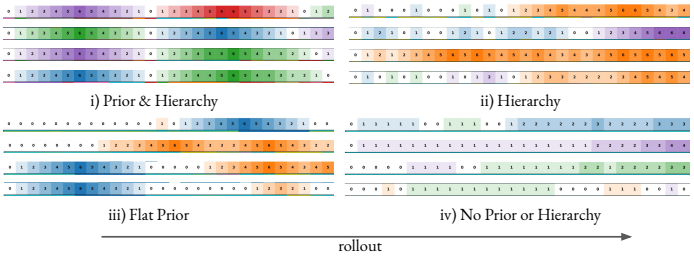


Figure 3: *CorridorMaze 2 corridor*. 4 rollouts shown for each method, with each episode rolled out horizontally. Corridors are colour coded and depth within them is denoted by shade; the darker the deeper. Hallway is white. See text for detailed analysis.

5.3 Information Asymmetry For Knowledge Transfer

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 skills transfer across these instances. The results in Table 3a demonstrate the role that IA has in achieving effective skill 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 array of transfer tasks*.

5.4 Transferability-Expressivity Trade-Off

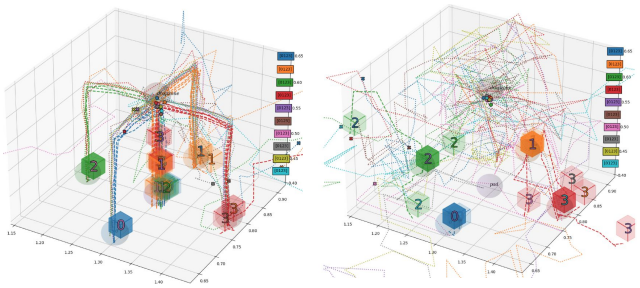
Table 1: Distillation Losses

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)

To understand the effects of IA on transfer, we refer back to Section 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: 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 3b presents the improvement in transfer performance attained 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 1). We thus expect larger improvement in transfer performance for these priors. Table 3b 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 3, conditioning on too little information limits the expressivity of the skills captured. To observe whether this is the case, we compare distillation losses between prior and policy for various IAs. The results are shown in Table 1 (showing mean/standard deviation for 4 seeds). As expected, *reduced IA reduces distillation losses* showing improved ability to distil behaviours present



i) Prior & Hierarchy ii) Hierarchy

Figure 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.

during training. Tables 1, 3a and 3b demonstrate the *transferability-expressivity trade-off*, which is independent of transfer task and more important for sparse domains, present for distinct IAs.

5.5 Hierarchy for Expressivity

Table 2: Hierarchy Ablation

Environment	CorridorMaze	
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

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 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 regularizing against a flat prior, KL-regularization 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 regularization occurs only over the action-space, *APES-H1-KL-a*.

Comparing transfer results for ablations *APES-H1-KL-a* and *APES-H1-KL-flat*, in Table 2, 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 5.6) can be seen to be caused by its inability to capture multi-modal behaviours present in the training tasks. For bottleneck states [28], where multi-modality is needed the most, this leads to regularization against a highly suboptimal unimodal action distribution. Interestingly, by comparing *APES-H1* and *APES-H1-KL-a*, we observe minimal benefits regularizing against latent rather than raw actions, suggesting that with alternate methods for achieving multi-modal priors, hierarchy may not be necessary. Finally, comparing *APES-no_prior* with *RecSAC* , we observe that hierarchy alone yields significantly less transfer benefits for sparse-domain tasks than priors.

5.6 Exploration Analysis

To gain further understanding on the effects of hierarchy and priors, we visualise the policy rollouts early on ($5k$ steps) in the transfer for APES, its ablations, and baselines. For *CorridorMaze* (Fig. 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 skills, 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 (Fig. 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 manipulation skills.

6 Related Work

Hierarchical frameworks have a long history [28]. The option, conditioned, RL literature tackle the semi-MDP setup and explore the benefits that hierarchy and temporal abstraction bring [29, 13, 20].

Table 3: 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

Wulfmeier et al. [9, 13] use hierarchy to enforce knowledge transfer through shared hierarchical modules. However, for lifelong learning, where number of skills increase over time, it is unclear how well these approaches will fare, without priors to narrow skill exploration.

Priors have been used in various fields. In the context of offline-RL, Siegel et al. [30], Wu et al. [31] primarily use priors to tackle value overestimation [32]. In the variational literature, priors have been used to guide latent-space learning [11, 20, 19, 33]. Hausman et al. [11] learn episodic skills, limiting their ability to transfer. Igl et al. [20] 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 literature, priors have been used to guide exploration [19, 8, 30, 34, 7], yet without hierarchy expressivity in learnt behaviours is limited. Recently, concurrent with our research, Tirumala et al. [22] were able to exploit hierarchical priors to transfer entire history-dependent high-level skills. However, their setup was in a high-data regime ($1e9$ data samples for pre-training skills), where covariate shift is less predominant. Unlike the aforementioned works, we consider the POMDP setting, 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. [20], Wulfmeier et al. [9, 13] only employ task or goal asymmetry and Tirumala et al. [14], Merel et al. [10], Galashov et al. [8] use exteroceptive asymmetry. Salter et al. [35], Galashov et al. [8] use asymmetry as a form of achieving robust and generalizable behaviours. Salter et al. [35] also investigate a way of learning asymmetry. We provide principled investigation on the role of asymmetry and point out its crucial role in sequential transfer setting.

7 Conclusion

In this paper, we employ hierarchical KL-regularized RL to efficiently transfer skills across multiple abstraction levels, showing the effectiveness of combining hierarchy and priors. We propose temporal information asymmetry, as an effective method for sequential skill discovery and transfer, and leverage 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 skills across sequential tasks. Our experiments validate the importance of this trade-off for not just extrapolated domains, but also for 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. Future work will include how to learn optimal IAs, given weak assumptions about the distribution of transfer tasks.

References

[1] 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.

[2] 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):529–533, 2015.

[3] 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. *arXiv preprint arXiv:1509.02971*, 2015.

[4] 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.

[5] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

[6] 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.

[7] Y. W. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. *arXiv preprint arXiv:1707.04175*, 2017.

[8] 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.

[9] 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.

[10] 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.

[11] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

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

[13] 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.

[14] 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.

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

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

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

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

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

[35] S. Salter, D. Rao, M. Wulfmeier, R. Hadsell, and I. Posner. Attention-privileged reinforcement learning. In *Conference on Robot Learning*, 2020.

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

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

[38] 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.

A Method

A.1 Training Regime

In this section, we algorithmically describe our training setup. We relate each training phase to the principle equations in the main paper, but note that Appendices A.2 and A.3 outline a more detailed version of these equations that were actually used.

Algorithm 1 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 reinitialized.

 # Behavioral Cloning
Initialize: 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)
    $\pi, \pi_0 \leftarrow$ BC_update(π, π_0, R_{bc}) *# Eq. 4*
end for
 # Reinforcement Learning
Initialize: 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*
    $Q_k \leftarrow$ RL_critic_update(Q_k, π, R_{rl}) *# Eq. 10*
end for

Algorithm 2 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($\mathbf{x}, \mathbf{a}_f$,r$_k, \mathbf{x}'$)
    return R_j, env
end function

A.2 Variational Behavioral Cloning and Reinforcement Learning

Behavioral Cloning (BC) and KL-Regularized 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:

$$O_{bc}(\pi, \{\pi_i\}^{i\in I}) = -\sum_{i\in I} \mathrm{D_{KL}}\left(\pi(\tau) \parallel \pi_i(\tau)\right), \;\; O_{rl}(\pi, \{\pi_i\}^{i\subset I}) = E_\pi(\tau)[R(\tau)] + O_{bc}(\pi, \{\pi_i\}^{i\subset I}) \tag{6}$$

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, [19]. 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 [4], appendix B.3.3 for proof, further explanation, and necessary conditions. During transfer using RL, we do **not** have access to the expert or its demonstrations ($i \subset I := i \in \{0, u\}$).

For hierarchical policies, the KL terms are not easily evaluable. $\mathrm{D_{KL}}\left(\pi(\tau) \parallel \pi_i(\tau)\right) \leq \sum_t \mathbb{E}_{\pi(\tau)}\left[\mathrm{D_{KL}}\left(\pi^H(\mathbf{z}_t|\mathbf{x}_k) \parallel \pi_i^H(\mathbf{z}_t|\mathbf{x}_k)\right) + \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_k)}\left[\mathrm{D_{KL}}\left(\pi^L(\mathbf{a}_t|\mathbf{x}_k, \mathbf{z}_t) \parallel \pi_i^L(\mathbf{a}_t|\mathbf{x}_k, \mathbf{z}_t)\right)\right]\right]$ [2] [14], is a commonly chosen upper bound. 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 Eq. (6) amendable to off-policy training (experience from $\{\pi_e, \pi_b\}$, for

[2] For proof refer to Appendix B.2

BC/RL respectively; π_b representing behavioral 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 attain:

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

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

$$-\mathrm{D}_{\mathrm{KL}}(\pi(\tau)||\pi_e(\tau)) \geq -\sum_{i\in\{0,u,e\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_e(\tau)}(\pi(\tau)||\pi_i(\tau)) \tag{7}$$

$$\mathbb{E}_{\substack{p(\mathcal{K}),\\ \pi_0(\tau)}}\left[\log(O=1|\tau,k)\right] \geq \mathbb{E}_{\pi_b(\tau)}\left[\sum_t \nu^{\pi_b}[t] \cdot r_k(\mathbf{x}_t, \mathbf{a}_t)\right] - \sum_{i\in\{0,u\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_b(\tau)}(\pi(\tau)||\pi_i(\tau)) \tag{8}$$

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 timestep i, between π and arbitrary policy n. $\nu^n[t]$ is the t^{th} element of ν^n; the cumulative IW product at timestep t. Equations 7, 8 are the BC/RL lower bounds used for policy gradients. See Appendix B.3 for a derivation, and necessary 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, maximizes return for both policy and prior, whilst minimizing the disparity between them. Regularising against an uninformative prior, π_u, encourages highly-entropic policies, further aiding at exploration and stabilising learning [20].

In RL, IWs are commonly ignored [3, 4, 5], 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 otherwize), 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, rather than using sampling approximations [36]. For BC, we train for 1 epoch (referring to training in the expectation once over each sample in the replay buffer).

A.3 Critic Learning

The lower bound presented in Eq. (8) is non-differentiable due to rewards being sampled from the environment. Therefore, as is common in the RL literature [2, 3], 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 [37] using the Retrace algorithm [25] to provide a low-variance, low-bias, policy evaluation operator:

$$Q_t^{\text{ret}} := Q'(\mathbf{x}_t, \mathbf{a}_t, k) + \sum_{j=t}^{\infty} \epsilon_j^t \left[r_k(\mathbf{x}_j, \mathbf{a}_j) + \mathbb{E}_{\substack{\pi^H(\mathbf{z}|\mathbf{x}_{j+1},k),\\ \pi^L(\mathbf{a}'|\mathbf{x}_{j+1},\mathbf{z},k)}}\left[Q'(\mathbf{x}_{j+1}, \mathbf{a}', k)\right] - Q'(\mathbf{x}_j, \mathbf{a}_j, k)\right] \tag{9}$$

$$L(Q) = \mathbb{E}_{\substack{p(\mathcal{K}),\\ \pi_b(\tau)}}\left[(Q(\mathbf{x}_t, \mathbf{a}_t, k) - \arg\min_{Q_t^{\text{ret}}}(Q_t^{\text{ret}}))^2\right] \qquad \epsilon_j^t = \gamma^{j-t}\prod_{i=t+1}^{j}\zeta_i^b \tag{10}$$

Where Q_t^{ret} represents the policy return evaluated via Retrace. Q' is the target Q-network, commonly used to stabilize critic learning [2], and is updated periodically with the current Q values. IWs are not ignored here, and are clipped between $[0, 1]$ to prevent exploding gradients, [25]. To further reduce bias and overestimates of our target, Q_t^{ret}, we apply the double Q-learning trick, [26], and concurrently learn two target Q-networks, Q'. Our critic is trained to minimize the loss in Eq. (10), which regularizes the critic against the minimum of the two targets produced by both target networks.

B Theory and Derivations

In this section we provide proofs for the theory introduced in the main paper and in Appendix A.

B.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 and inputs*

$$\begin{aligned} &\mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{b}) \parallel q(\mathbf{b})\right) \geq \mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{c}) \parallel q(\mathbf{c})\right) \\ &\quad \textit{with } \mathbf{b} = (b_0, b_1, ..., b_n) \textit{ and } \mathbf{c} \subset \mathbf{b}. \end{aligned} \tag{11}$$

Proof

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{b}) \parallel q(\mathbf{b})\right) &= \mathbb{E}_{p(\mathbf{b})}\left[\log\left(\frac{p(\mathbf{b})}{q(\mathbf{b})}\right)\right] \\ &= \mathbb{E}_{p(\mathbf{d}|\mathbf{c})\cdot p(\mathbf{c})}\left[\log\left(\frac{p(\mathbf{d}|\mathbf{c})\cdot p(\mathbf{c})}{q(\mathbf{d}|\mathbf{c})\cdot q(\mathbf{c})}\right)\right] \quad \textit{with} \quad \mathbf{d} \in \mathbf{b} \oplus \mathbf{c} \\ &= \mathbb{E}_{p(\mathbf{c})}\left[\mathbb{E}_{p(\mathbf{d}|\mathbf{c})}\left[1\right] \cdot \log\left(\frac{p(\mathbf{c})}{q(\mathbf{c})}\right)\right] + \mathbb{E}_{p(\mathbf{c})}\left[\mathbb{E}_{p(\mathbf{d}|\mathbf{c})}\left[\log\left(\frac{p(\mathbf{d}|\mathbf{c})}{q(\mathbf{d}|\mathbf{c})}\right)\right]\right] \\ &= \mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{c}) \parallel q(\mathbf{c})\right) + \mathbb{E}_{p(\mathbf{c})}\left[\mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{d}|\mathbf{c}) \parallel q(\mathbf{d}|\mathbf{c})\right)\right] \\ &\geq \mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{c}) \parallel q(\mathbf{c})\right) \quad \textit{given} \quad \mathbb{E}_{p(\mathbf{c})}\left[\mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{d}|\mathbf{c}) \parallel q(\mathbf{d}|\mathbf{c})\right)\right] \geq 0 \end{aligned} \tag{12}$$

B.2 Hierarchical KL-Divergence Upper Bound

All proofs in this section ignore multi-task setup for simplicity. Extending to this scenario is trivial.

Upper Bound

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}\left(\pi(\tau) \parallel \pi_i(\tau)\right) \leq \sum_t \mathbb{E}_{\pi(\tau)}[&\mathrm{D}_{\mathrm{KL}}\left(\pi^H(\mathbf{z}_t|\mathbf{x}_t) \,\middle\|\, \pi_i^H(\mathbf{z}_t|\mathbf{x}_t)\right) \\ &+ \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_t)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^L(\mathbf{a}_t|\mathbf{x}_t, \mathbf{z}_t) \,\middle\|\, \pi_i^L(\mathbf{a}_t|\mathbf{x}_t, \mathbf{z}_t)\right)\right]] \end{aligned} \tag{13}$$

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)}[\mathrm{D}_{\mathrm{KL}}\left(\pi(\mathbf{a}_t|\mathbf{x}_t) \parallel \pi_i(\mathbf{a}_t|\mathbf{x}_t)\right) + \\ &\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]] \\ &= \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)}[\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{14}$$

B.3 Policy Gradient Lower Bounds

B.3.1 Importance Weights Derivation

$$\tilde{\mathrm{D}}_{\mathrm{KL}}^{q(\tau)}(\pi(\tau)||\pi_i(\tau)) = ub(\mathrm{D}_{\mathrm{KL}}\left(\pi(\tau) \parallel \pi_i(\tau)\right)) \tag{15}$$

For $\beta_i^z, \beta_i^a = 1$, where $ub(\mathrm{D}_{\mathrm{KL}}\left(\pi(\tau) \parallel \pi_i(\tau)\right))$ corresponds to the hierarchical upper bound introduced in Appendix A.2.

Proof

$$\begin{aligned}
ub(\mathrm{D}_{\mathrm{KL}}\left(\pi(\tau) \parallel \pi_i(\tau)\right)) &= \sum_t \mathbb{E}_{\pi(\tau)}[\mathrm{D}_{\mathrm{KL}}\left(\pi^H(\mathbf{z}_t|\mathbf{x}_t) \parallel \pi_i^H(\mathbf{z}_t|\mathbf{x}_t)\right) \\
&\quad + \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_t)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t) \parallel \pi_i^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t)\right)\right]] \\
&= \sum_t \mathbb{E}_{q(\tau)\cdot\frac{\pi(\tau)}{q(\tau)}}[\mathrm{D}_{\mathrm{KL}}\left(\pi^H(\mathbf{z}_t|\mathbf{x}_t) \parallel \pi_i^H(\mathbf{z}_t|\mathbf{x}_t)\right) \\
&\quad + \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_t)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t) \parallel \pi_i^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t)\right)\right]] \\
&= \sum_t \mathbb{E}_{q(\tau)\cdot\prod_{i=0}^t \frac{\pi(\mathbf{a}_i|\mathbf{x}_i)}{q(\mathbf{a}_i|\mathbf{x}_i)}}[\mathrm{D}_{\mathrm{KL}}\left(\pi^H(\mathbf{z}_t|\mathbf{x}_t) \parallel \pi_i^H(\mathbf{z}_t|\mathbf{x}_t)\right) \\
&\quad + \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_t)}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t) \parallel \pi_i^L(\mathbf{a}_t|\mathbf{x}_t,\mathbf{z}_t)\right)\right]] \\
&= \tilde{\mathrm{D}}_{\mathrm{KL}}^{q(\tau)}(\pi(\tau)||\pi_i(\tau))
\end{aligned} \tag{16}$$

B.3.2 Behavioral Cloning Upper Bound

$$\begin{aligned}
-\mathrm{D}_{\mathrm{KL}}(\pi(\tau)||\pi_e(\tau)) \geq -\sum_{i\in\{0,u,e\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_e(\tau)}(\pi(\tau)||\pi_i(\tau)) \\
\textit{for} \quad \beta_i^z, \beta_i^a \geq 1
\end{aligned} \tag{17}$$

Proof

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}\left(\pi(\tau) \parallel \pi_e(\tau)\right) &\leq \sum_{i\in\{0,u,e\}} \mathrm{D}_{\mathrm{KL}}\left(\pi(\tau) \parallel \pi_i(\tau)\right) \\
&\leq \sum_{i\in\{0,u,e\}} ub(\mathrm{D}_{\mathrm{KL}}\left(\pi(\tau) \parallel \pi_i(\tau)\right)) \\
&= \sum_{i\in\{0,u,e\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{q(\tau)}(\pi(\tau)||\pi_i(\tau)) \quad \textit{for} \quad \beta_i^z, \beta_i^a = 1 \\
&\leq \sum_{i\in\{0,u,e\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{q(\tau)}(\pi(\tau)||\pi_i(\tau)) \quad \textit{for} \quad \beta_i^z, \beta_i^a \geq 1
\end{aligned} \tag{18}$$

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

B.3.3 Reinforcement Learning Upper Bound

$$\begin{aligned}
&\mathbb{E}_{\substack{p(\mathcal{K}),\\ \pi_0(\tau)}}[\log(O=1|\tau,k)] \geq \mathbb{E}_{\pi_b(\tau)}\left[\sum_t \nu^{\pi_b}[t]\cdot r_k(\mathbf{x}_t,\mathbf{a}_t)\right] - \sum_{i\in\{0,u\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_b(\tau)}(\pi(\tau)||\pi_i(\tau)) \\
&\textit{for} \quad \beta_i^z, \beta_i^a \geq 1 \quad \textit{and} \quad r_k < 0
\end{aligned} \tag{19}$$

Proof

$$
\begin{aligned}
\mathbb{E}_{\substack{p(\mathcal{K}),\\ \pi_0(\tau)}}[\log(O=1|\tau,k)] &\geq \mathbb{E}_{\pi_b(\tau)}\left[\sum_t \nu^{\pi_b}[t]\cdot r_k(\mathbf{x}_t,\mathbf{a}_t)\right] - \mathrm{D}_{\mathrm{KL}}\left(\pi(\tau)\parallel \pi_i(\tau)\right) \\
&\geq \mathbb{E}_{\pi_b(\tau)}\left[\sum_t \nu^{\pi_b}[t]\cdot r_k(\mathbf{x}_t,\mathbf{a}_t)\right] - \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_b(\tau)}(\pi(\tau)||\pi_i(\tau)),\ \textit{for}\ \beta_i^z,\beta_i^a=1 \\
&\geq \mathbb{E}_{\pi_b(\tau)}\left[\sum_t \nu^{\pi_b}[t]\cdot r_k(\mathbf{x}_t,\mathbf{a}_t)\right] - \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_b(\tau)}(\pi(\tau)||\pi_i(\tau)),\ \textit{for}\ \beta_i^z,\beta_i^a\geq 1 \\
&\geq \mathbb{E}_{\pi_b(\tau)}\left[\sum_t \nu^{\pi_b}[t]\cdot r_k(\mathbf{x}_t,\mathbf{a}_t)\right] - \sum_{i\in\{0,u\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_b(\tau)}(\pi(\tau)||\pi_i(\tau)), \\
&\quad \textit{for}\quad \beta_i^z,\beta_i^a\geq 1
\end{aligned}
\tag{20}
$$

For proof for line 1, please refer to [4]. The final 2 lines hold due to positive KL-divergences.

C Environments

We continue by covering each environment setup in detail.

C.1 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.

$s \in \{0,l\}^c$, $p(s_0)=\mathbf{0}^c$, $k=$ 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}. Specifically we define the transition function as follows:

$$
f(\mathbf{s}_t^{j_t},\mathbf{a}_t)=\begin{cases}\mathbf{s}_t^{j_t}+1, & \text{if } \psi_{inc}^w(\mathbf{a}_t,j_t).\\ \mathbf{s}_t^{j_t}-1, & \text{elif } \psi_{dec}^w(\mathbf{a}_t,j_t).\\ 0, & \text{otherwise.}\end{cases}
\tag{21}
$$

$\psi_{inc}^w(\mathbf{a}_t,j)=\text{bool}(\mathbf{a}_t \text{ in } [j/c,(j+0.5\cdot w)/c])$, $\psi_{dec}^w(\mathbf{a}_t,j)=\text{bool}(\mathbf{a}_t \text{ in } [j/c,(2\cdot j-0.5\cdot w)/c])$. The smaller the w parameter, the narrower the distribution of actions that lead to transitions. As such, w, together with r_k controls 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.

C.2 Stack

This domain is adapted from the well known gym robotics FetchPickAndPlace-v0 environment [38]. 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 Fig. 2, 3) the number of substeps was increased

Table 4: Feedforward Module, $\pi_{(0)}^{\{H,L\}}$

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

Table 5: Full experimental setup. Describes inputs to each module, level over which KL-regularization occurs, which modules are shared, and which are 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

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 Fig. 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, $r_k^{\text{sparse}}(\mathbf{x}_t, \mathbf{a}_t) = 1 \quad \textit{if} \quad$ correct object has been placed on stack in correct ordering $\quad \textit{else} \quad 0$.

D Experimental Setup

We provide the reader with the experimental setup for all training regimes and environments below. We build off the softlearning code base [5]. Algorithmic details not mentioned in the following sections 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.

D.1 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 comprized of a feedforward module outlined in Table 4. The softlearning repository that we build off [5], applies tanh activation over network outputs, where appropriate, to match the predefined outpute ranges of any given module. The critic is also comprized 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 5 describes the setup for all the experiments in the main paper, including inputs to each module, level over which KL-regularization 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 3b, we additionally reuse π^H (or π^L for RecSAC) across domains, and whose input is $\mathbf{x}_t$. For all the above 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 generalize across task instances.

D.2 Behavioural Cloning

For the BC setup, we use a deterministic, noisy, expert controller to create experience to learn off. We apply DAGGER [23] 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 during training, between policy and expert. Noize 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 [5], so please refer to it for details regarding this interplay.

Table 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

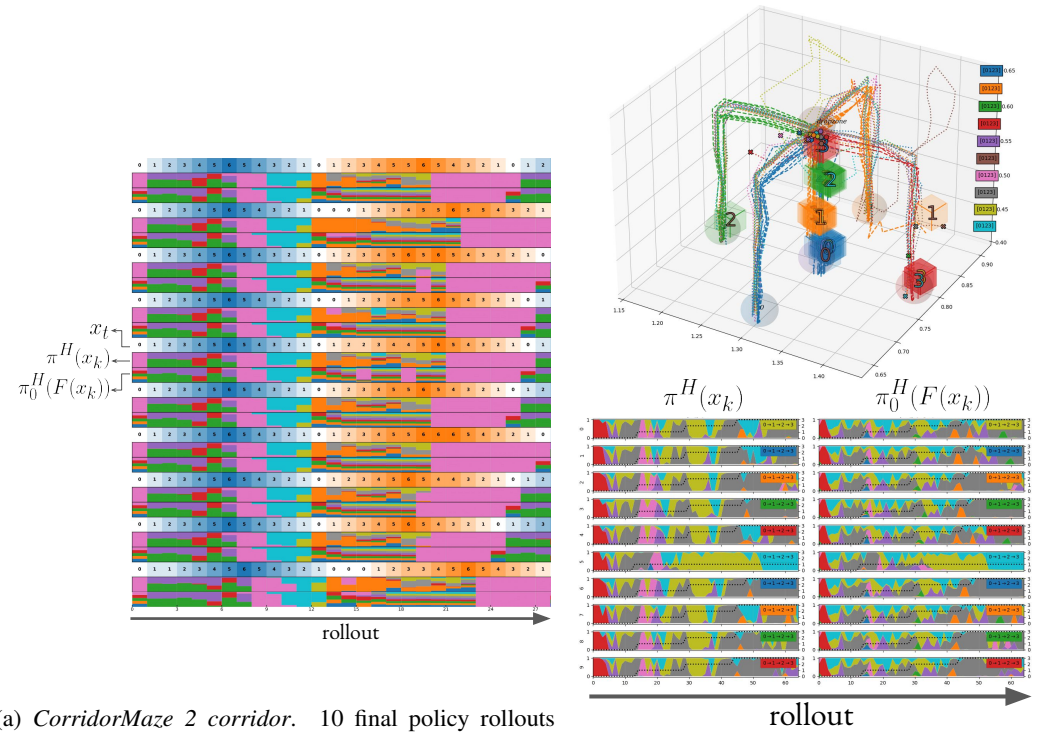
Environment	CorridorMaze		Stack
Reward Type	sparse	semi-sparse	sparse
Transfer task	2 corridor	4 corridor	4 blocks
Q learning rate	$3e^{-6}$	$3e^{-5}$	$3e^{-5}$
π learning rate	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$
$\beta_0^{z/a}$	$1e^{-2}$	$1e^{-1}$	$5e^{-2}$
$\beta_u^{z/a}$	$1e^{-2}$	0	$5e^{-4}$
Q update rate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

Refer to Table 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 in this paper. 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 state the history lengths used for π^H for BC in Table 6a. This value was also used for both 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 paper, 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.

D.3 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



(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 Fig. 5a. Horizontal dashed lines in bottom plots refer to current sub-task (block stack), vertically transitioning upon each completion.

Figure 5: 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 stack next), but where determinism is required for the task at hand.

for the covariate shift experiments in Table 3b). Any modules that are not shared (such as π^H for most experiments), are initialized randomly across tasks. The RL setup is akin to the softlearning repository [5] that we build off. We note any changes in Table 6b. We regularize against the latent or action level for, 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 regularize against, at most, two β hyper-parameters. Hyper-parameter sweeps are performed in the same way as in the previous 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 [25], and perform λ returns rather than n-step. We found the Retrace operator important for sample-efficient learning.

E 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 Figs. 5a and 6, we see that the full method successfully solves each respective task, correctly 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

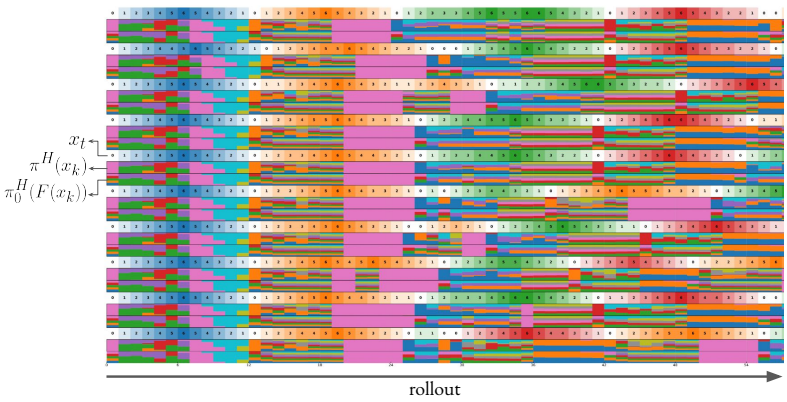


Figure 6: *CorridorMaze 4 corridor*. 10 final policy rollouts (episodes vertically, rollouts horizontally) for most performant method. Task: Blue -> Orange -> Green -> Red corridors. Policy distribution over categorical latent space for π^H and π_0^H plotted (between policy rollouts, denoted by x_t) as vertical histograms, with colour and width denoting category and probability.

the most from each-other at the hallway, the *bottleneck state* [28], 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.

Fig. 5b 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).