Sub-policy Adaptation for Hierarchical Reinforcement Learning

Alexander C. Li*, **Carlos Florensa***, **Ignasi Clavera**, **Pieter Abbeel**
Department of Computer Science
University of California, Berkeley
{alexli1, florensa, iclavera, pabbeel}@berkeley.edu

Abstract

Hierarchical Reinforcement Learning is a promising approach to long-horizon decision-making problems with sparse rewards. Unfortunately, most methods still decouple the lower-level skill acquisition process and the training of a higher level that controls the skills in a new task. Treating the skills as fixed can lead to significant sub-optimality in the transfer setting. In this work, we propose a novel algorithm to discover a set of skills, and continuously adapt them along with the higher level even when training on a new task. Our main contributions are two-fold. First, we derive a new hierarchical policy gradient, as well as an unbiased latent-dependent baseline. We introduce Hierarchical Proximal Policy Optimization (HiPPO), an on-policy method to efficiently train all levels of the hierarchy simultaneously. Second, we propose a method of training time-abstractions that improves the robustness of the obtained skills to environment changes. Code and results are available at `sites.google.com/view/hippo-rl`.

1 Introduction

Reinforcement learning (RL) has made great progress in a variety of domains, from playing games such as Pong and Go [1, 2] to automating robotic locomotion [3, 4, 5], dexterous manipulation [6, 7], and perception [8, 9]. Yet, most work in RL is still learning a new behavior from scratch when faced with a new problem. This is particularly inefficient when dealing with tasks that are hard to solve due to sparse rewards or long horizons, or when solving many related tasks.

A promising technique to overcome this limitation is Hierarchical Reinforcement Learning (HRL) [10, 11]. In this paradigm, policies have several modules of abstraction, so the reuse of a subset of the modules becomes easier. The most common case consists of temporal abstraction [12, 13], where a higher-level policy (manager) takes actions at a lower frequency, and its actions condition the behavior of some lower level skills or sub-policies. When transferring knowledge to a new task, most prior works fix the skills and train a new manager on top. Despite having a clear benefit in kick-starting the learning in the new task, having fixed skills can considerably cap the final performance on the new task [11]. Little work has been done on adapting pre-trained sub-policies to be optimal for a new task.

In this paper, we develop a new framework for adapting all levels of temporal hierarchies simultaneously. First, we derive an efficient approximated hierarchical policy gradient. The key insight is that, despite the decisions of the manager being unobserved latent variables from the point of view of the Markovian environment, from the perspective of the sub-policies they can be considered as part of the observation. This provides a type of decoupling of the gradient with respect to the manager and the sub-policies parameters that greatly simplifies the policy gradient computation, in a principled way. It also justifies theoretically a technique used in other prior works [14]. Second, we introduce a sub-policy specific baseline for our hierarchical policy gradient. We show using this baseline is unbiased, and our experiments reveal faster convergence, suggesting efficient gradient variance

Preprint. Under review.

reduction. Then we introduce a more stable way of using this gradient, Hierarchical Proximal Policy Optimization (HiPPO). This helps us take more conservative steps in our policy space [15], necessary in hierarchies because of the interdependence of each layer. Finally we also evaluate the benefit of varying the time-commitment to the sub-policies, and show it helps both in terms of final performance and zero-shot adaptation to similar tasks.

2 Related Work

The key points in HRL are how the different levels of the hierarchy are defined, trained, and then re-used. In this work, we are interested in approaches that allow us to build temporal abstractions by having a higher level taking decisions at a slower frequency than a lower-level. There has been growing interest in HRL for the past few decades [10, 12], but only recently has it been applied to high-dimensional continuous domains as we do in this work [16, 17].

To obtain the lower level policies, or skills, most methods exploit some additional assumptions, like access to demonstrations [18, 19, 20, 21], policy sketches [22], or task decomposition into sub-tasks [23, 24]. Other methods use a different reward for the lower level, often constraining it to be a "goal reacher" policy, where the signal from the higher level is the goal to reach [25, 26, 27]. These methods are very promising for state-reaching tasks, but might require access to goal-reaching reward systems not defined in the original MDP, and are more limited when training on tasks beyond state-reaching. Our method does not require any additional supervision, and the obtained skills are not constrained to be goal-reaching.

When transferring skills to a new environment, most HRL methods keep them fixed and simply train a new higher-level on top [28, 29]. Other work allows for building on previous skills by constantly supplementing the set of skills with new ones [30], but they require a hand-defined curriculum of tasks, and the previous skills are never fine-tuned. Our algorithm allows for seamless adaptation of the skills, showing no trade-off between leveraging the power of the hierarchy and the final performance in a new task. Other methods use invertible functions as skills [31], and therefore a fixed skill can be fully over-written when a new layer of hierarchy is added on top. This kind of "fine-tuning" is promising, although they do not apply it to temporally extended skills as we are interested in here.

One of the most general frameworks to define temporally extended hierarchies is the options framework [10], and it has recently been applied to continuous state spaces [32]. One of the most delicate parts of this formulation is the termination policy, and it requires several regularizers to avoid skill collapse [33, 34]. This modification of the objective may be difficult to tune and affects the final performance. Instead of adding such penalties, we propose having skills of a random length, not controlled by the agent during training of the skills. The benefit is two-fold: no termination policy to train, and more stable skills that transfer better. Furthermore, these works only used discrete action MDPs. We lift this assumption, and show good performance of our algorithm in complex locomotion tasks.

The closest work to ours in terms of final algorithm is the one proposed by Frans et al. [14]. Their method can be included in our framework, and hence benefits from our new theoretical insights. We also introduce two modifications that are shown to be highly beneficial: the random time-commitment explained above, and the notion of an information bottleneck to obtain skills that generalize better.

3 Preliminaries

We define a discrete-time finite-horizon discounted Markov decision process (MDP) by a tuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma, H)$, where $\mathcal{S}$ is a state set, $\mathcal{A}$ is an action set, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ is the transition probability distribution, $\gamma \in [0, 1]$ is a discount factor, and H the horizon. Our objective is to find a stochastic policy π_θ that maximizes the expected discounted reward within the MDP, $\eta(\pi_\theta) = \mathbb{E}_\tau[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)]$. We denote by $\tau = (s_0, a_0, ...,)$ the entire state-action trajectory, where $s_0 \sim \rho_0(s_0)$, $a_t \sim \pi_\theta(a_t|s_t)$, and $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$.

In this work, we tackle the problem of learning a hierarchical policy and efficiently adapting all the levels in the hierarchy to perform a new task. Usually, hierarchical policies are composed by a higher level, or manager $\pi_{\theta_h}(z_t|s_t)$, and a lower level, or sub-policy $\pi_{\theta_l}(a_{t'}|z_t, s_{t'})$. The higher level does not take actions in the environment directly, but rather outputs a command, or latent variable $z_t \in \mathcal{Z}$,

that conditions the behavior of the lower level. In this line of work, the manager typically operates at a lower frequency than the sub-policies. Specifically, the manager just observes the environment every p time-steps. When the manager receives a new observation it decides which low level policy to commit to for p environment steps by the means of a latent code z. Figure 1 depicts this set-up where the high level frequency p is a random variable, which is a contribution of this paper as described in future sections.

4 Problem Statement

Hierarchical Reinforcement Learning (HRL) has the potential to leverage previously acquired knowledge to accelerate learning in related tasks. For instance, if an agent has learned to navigate in a certain environment, a good hierarchical policy could decompose the task into different sub-policies, each sub-policy advancing the agent in different directions. Then, in a new navigation problem, the manager of the hierarchical policy can make use of each of the sub-policies, effectively improving exploration when those are executed for an extended period of time. As a result, longer horizon tasks and tasks that require multiple skills can be solved in an efficient manner.

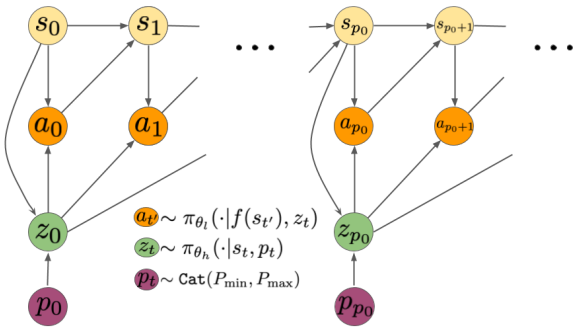

Figure 1: Temporal hierarchy studied in this paper. A latent code z_t is sampled from the manager policy $\pi_{\theta_h}(z_t|s_t)$ every p time-steps, using the current observation s_{kp}. The actions a_t are sampled from the sub-policy $\pi_{\theta_l}(a_t|s_t, z_{kp})$ conditioned on the same latent code from timestep $t = kp$ to timestep $(k+1)p - 1$

Prior works have been focused on learning a manager that combines these sub-policies, but they do not further train the sub-policies when learning a new task. However, preventing the skills from learning results in sub-optimal behavior in new tasks. This effect is exacerbated when the skills were learned in a task agnostic way or in a different environment. In this paper, we present a HRL method that learns all levels of abstraction in the hierarchical policy: the manager learns to make use of the low-level skills, while the skills are continuously adapted to attain maximum performance in the given task. We derive a policy gradient update for hierarchical policies that monotonically improves the performance. Furthermore, we demonstrate that our approach prevents sub-policy collapse behavior, when the manager ends up using just one skill, observed in previous approaches.

5 Efficient Hierarchical Policy Gradients

When using a hierarchical policy, the intermediate decision taken by the higher level is not directly applied in the environment. This consideration makes it unclear how it should be incorporated into the Markovian framework of RL: should it be treated as an observed variable, like an action, or as a latent? The answer to this question impacts the methods applicable to HRL and how the gradient of the RL objective with respect to the parameters of a hierarchical policy is computed.

In this section, we first prove that one framework is an approximation of the other under mild assumptions. Then, we derive an unbiased baseline for the HRL setup that reduces its variance. Thirdly, we introduce the notion of information bottleneck and trajectory compression, which proves critical for learning reusable skills. Finally, with these findings, we present our method, Hierarchical Proximal Policy Optimization (HiPPO), an on-policy algorithm for hierarchical policies that monotonically improves the RL objective, allowing learning at all levels of the policy and preventing sub-policy collapse.

5.1 Approximate Hierarchical Policy Gradient

Policy gradient algorithms are based on the likelihood ratio trick [35] to estimate the gradient of returns with respect to the policy parameters as

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_\tau \big[\nabla_\theta \log P(\tau) R(\tau)\big] \approx \frac{1}{N} \sum_{i=1}^{n} \nabla_\theta \log P(\tau_i) R(\tau_i). \tag{1}$$

In the context of HRL, a hierarchical policy with a manager $\pi_{\theta_h}(z_t|s_t)$ selects every p time-steps one of n sub-policies to execute. These sub-policies, indexed by $z \in [n]$, can be represented as a single conditional probability distribution over actions $\pi_{\theta_l}(a_t|z_t, s_t)$. This allows us to not only use a given set of sub-policies, but also leverage skills learned with Stochastic Neural Networks (SNNs) [11]. Under this framework, the probability of a trajectory $\tau = (s_0, a_0, s_1, \ldots, s_H)$ can be written as

$$P(\tau) = \Bigg(\prod_{k=0}^{H/p} \Big[\sum_{j=1}^{n} \pi_{\theta_h}(z_j|s_{kp}) \prod_{t=kp}^{(k+1)p-1} \pi_{\theta_l}(a_t|s_t, z_j) \Big] \Bigg) \Bigg[P(s_0) \prod_{t=1}^{H} P(s_{t+1}|s_t, a_t) \Bigg]. \tag{2}$$

The mixture action distribution, which presents itself as an additional summation over skills, prevents additive factorization when taking the logarithm, as in Eq. 1. This can yield considerable numerical instabilities due to the product of the p sub-policy probabilities. For instance, in the case where all the skills are distinguishable all the sub-policies probabilities but one will have small values, resulting in an exponentially small value. In the following Lemma, we derive an approximation of the policy gradient, whose error tends to zero as the skills become more diverse, and draw insights on the interplay of the manager actions.

Lemma 1. *If the skills are sufficiently differentiated, then the latent variable can be treated as part of the observation to compute the gradient of the trajectory probability. Let $\pi_{\theta_h}(z|s)$ and $\pi_{\theta_l}(a|s, z)$ be Lipschitz functions w.r.t. their parameters, and assume that $0 < \pi_{\theta_l}(a|s, z_j) < \epsilon \; \forall j \neq kp$, then*

$$\nabla_\theta \log P(\tau) = \sum_{k=0}^{H/p} \nabla_\theta \log \pi_{\theta_h}(z_{kp}|s_{kp}) + \sum_{t=0}^{H} \nabla_\theta \log \pi_{\theta_l}(a_t|o_t, z_{kp}) + \mathcal{O}(nH\epsilon^{p-1}) \tag{3}$$

Proof. See Appendix. □

Our assumption can be seen as having diverse skills. Namely, for each action there is just one sub-policy that gives it high probability. In this case, the latent variable can be treated as part of the observation to compute the gradient of the trajectory probability. Many algorithms to extract lower-level skills are based on promoting diversity among the skills [11, 36], therefore usually satisfying our assumption. We further analyze how well this assumption holds in our experiments section.

5.2 Unbiased Sub-policy Baseline

The policy gradient estimate obtained when applying the log-likelihood ratio trick as derived above is known to have large variance. A very common approach to mitigate this issue without biasing the estimate is to subtract a baseline from the returns [37]. The more precise the learned baseline can be, the more effective this technique is. It is well known that such baselines can be made state-dependent without incurring any bias. However, it is still unclear how to formulate a baseline for all the levels in a hierarchical policy, since an action dependent baseline does introduce bias in the gradient. Here, we show how, under the assumptions of Lemma 1, we can formulate an unbiased latent dependent baseline for the approximate gradient (Eq. 4).

Lemma 2. *For any functions $b_h : \mathcal{S} \to \mathbb{R}$ and $b_l : \mathcal{S} \times \mathcal{Z} \to \mathbb{R}$ we have:*

$$\mathbb{E}_\tau [\sum_{k=0}^{H/p} \nabla_\theta \log P(z_{kp}|s_{kp}) b_h(s_{kp})] = 0$$

$$\mathbb{E}_\tau [\sum_{t=0}^{H} \nabla_\theta \log \pi_{s,\theta}(a_t|s_t, z_{kp}) b_l(s_t, z_{kp})] = 0$$

Proof. See Appendix. □

Now we apply Lemma 1 and Lemma 2 to Eq. 1. By using the corresponding value functions as the function baseline, the return can be replaced by the Advantage function [3], and we obtain the following gradient expression:

$$\hat{g} = \mathbb{E}_\tau \Big[(\sum_{k=0}^{H/p} \nabla_\theta \log \pi_{\theta_h}(z_{kp}|s_{kp}) A(s_{kp}, z_{kp})) + (\sum_{t=0}^{H} \nabla_\theta \log \pi_{\theta_l}(a_t|s_t, z_{kp}) A(s_t, a_t, z_{kp})) \Big]$$

This hierarchical policy gradient estimate has lower variance than without baselines, but using it for policy optimization through stochastic gradient descent still yields an unstable algorithm. In the next section, we further improve the stability and sample efficiency of the policy optimization by incorporating techniques from Proximal Policy Optimization [15].

5.3 Hierarchical Proximal Policy Optimization

Using an appropriate step size in policy space is critical for stable policy learning. Modifying the policy parameters in some directions may have a minimal impact on the distribution over actions, whereas small changes in other directions might change its behavior drastically and hurt training efficiency. Trust region policy optimization (TRPO) uses a constraint on the KL-divergence between the old policy and the new policy to prevent this issue [3]. Unfortunately, hierarchical policies are generally represented by complex distributions without closed form expressions for the KL-divergence. Therefore, to improve the stability of our hierarchical policy gradient we turn towards Proximal Policy Optimization (PPO) [15]. PPO is a more flexible and compute-efficient algorithm. In a nutshell, it replaces the KL-divergence constraint with a cost function that achieves the same trust region benefits, but only requires the computation of the likelihood. Letting $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, the PPO objective is:

$$L^{CLIP}(\theta) = \mathbb{E}_t \min \left\{ r_t(\theta) A_t,\ \texttt{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) A_t \right\}$$

Since the likelihood ratio $r_t(\theta)$ comes from importance sampling, we can adapt our approximated hierarchical policy gradient with the same approach. Letting $r_{h,kp}(\theta) = \frac{\pi_{\theta_h}(z_{kp}|s_{kp})}{\pi_{\theta_{h,old}}(z_{kp}|s_{kp})}$ and $r_{l,t}(\theta) = \frac{\pi_{\theta_l}(a_t|s_t, z_{kp})}{\pi_{\theta_{l,old}}(a_t|s_t, z_{kp})}$, and using the super-index $\texttt{clip}$ to denote the clipped objective version, we obtain the new surrogate objective:

$$\begin{aligned} L^{CLIP}_{HiPPO}(\theta) = \mathbb{E}_\tau \Big[& \sum_{k=0}^{H/p} \min \left\{ r_{h,kp}(\theta) A(s_{kp}, z_{kp}), r^{\texttt{clip}}_{h,kp}(\theta) A(s_{kp}, z_{kp}) \right\} \\ & + \sum_{t=0}^{H} \min \left\{ r_{l,t}(\theta) A(s_t, a_t, z_{kp}), r^{\texttt{clip}}_{l,t}(\theta) A(s_t, a_t, z_{kp}) \right\} \Big] \end{aligned}$$

We call this algorithm Hierarchical Proximal Policy Optimization (HiPPO). Next, we introduce two critical additions: a switching of the time-commitment between skills, and an information bottleneck at the lower-level. Both are detailed in the following subsections.

5.4 Varying Time-commitment

Most hierarchical methods either consider a fixed time-commitment to the lower level skills [11, 14], or implement the complex options framework [12, 32]. In this work we propose an in-between, where the time-commitment to the skills is a random variable sampled from a fixed distribution $\texttt{Categorical}(T_{\min}, T_{\max})$ just before the manager takes a decision. This modification does not hinder final performance, and we show it improves zero-shot adaptation to a new task. This approach to sampling rollouts is detailed given in Algorithm 1.

Algorithm 1 `Collect Rollout`

Input: skills $\pi_{\theta_l}(a|s,z)$, manager $\pi_{\theta_h}(z|s)$, time-commitment bounds $P_{\min}$ and $P_{\max}$, horizon H, and bottleneck function $o = f(s)$
Reset environment: $s_0 \sim \rho_0$, $t = 0$.
while $t < H$ **do**
  Sample time-commitment $p \sim \texttt{Cat}([P_{\min}, P_{\max}])$
  Sample skill $z_t \sim \pi_{\theta_h}(\cdot|s_t)$
  for $t' = t \dots (t+p)$ **do**
    Sample action $a_{t'} \sim \pi_{\theta_l}(\cdot|f(s_{t'}), z_t)$
    Observe new state $s_{t'+1}$ and reward $r_{t'}$
  end for
  $t \leftarrow t + p$
end while
Output: $(s_0, z_0, a_0, o_1, a_1, \dots, s_H, z_H, a_H, o_{H+1})$

Algorithm 2 HiPPO

Input: skills $\pi_{\theta_l}(a|s,z)$, manager $\pi_{\theta_h}(z|s)$, horizon H, learning rate α
while not done **do**
  for actor = 1, 2, ..., N **do**
    Obtain trajectory with `Collect Rollout`
    Estimate advantages $\hat{A}(a_{t'}, o_{t'}, z_t)$ and $\hat{A}(z_t, s_t)$
  end for
  $\theta \leftarrow \theta + \alpha \nabla_\theta L^{CLIP}_{HiPPO}(\theta)$
end while

5.5 Information Bottleneck through Masking

If we apply the above HiPPO algorithm in the general case, there is little incentive to either learn or maintain a diverse set of skills. We claim this can be addressed via two simple additions:

- Let z only take a finite number of values
- Provide a masked observation to the skills $o_t = f(s_t)$

The masking function f restricts the information about the task, such that a single skill cannot perform the full task. Skill collapse is a common problem in hierarchical methods, requiring the use of regularizers [32, 33, 34]. There are some natural choices of f discussed in the robotics literature, like the agent-space and problem-space split [38, 11], that hide all task-related information and only allow the sub-policies to see proprioceptive information. With this setup, all the missing information needed to perform the task must come from the sequence of latent codes passed to the skills. We can interpret this as a lossy compression, whereby the manager encodes the relevant problem information into $\log n$ bits sufficient for the next p timesteps. The full algorithm is detailed in Algorithm 2.

6 Experiments

We design the experiments to answer the following questions: 1) How does HiPPO compare against a flat policy when learning from scratch? 2) Does it lead to more robust policies? 3) How well does it adapt already learned skills? and 4) Does our skill diversity assumption hold in practice?

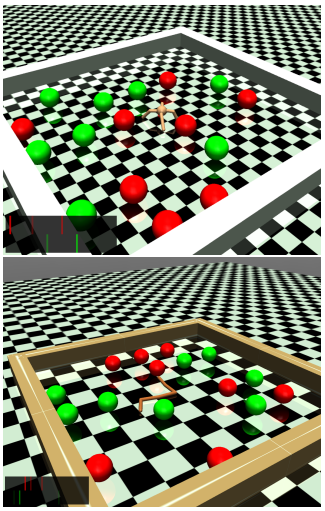

Figure 2: Snake and Ant are the two agents that we evaluate in the Gather environments.

6.1 Tasks

To answer the posed questions, we evaluate our new algorithms on a variety of robotic navigation tasks. Each task is a different robot trying to solve the Gather environment [39], depicted in Figure 2, in which the agent must collect apples (green balls, +1 reward) while avoiding bombs (red balls, -1 reward). This is a challenging hierarchical task with sparse rewards that requires agents to simultaneously learn perception, locomotion, and higher-level planning capabilities. We use 2 different types of robots within this environment. Snake is a 5-link robot with a 17-dimensional observation space and 4-dimensional action space; and Ant a quadrupedal robot with a 27-dimensional observation space and 8-dimensional action space. Both can move and rotate in all directions, and Ant faces the added challenge of avoiding falling over irrecoverably.

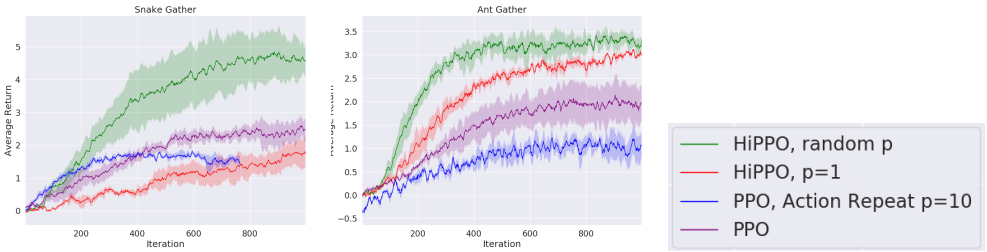

Figure 3: Comparison of Flat PPO, HiPPO, and HiPPO with randomized period learning from scratch on different environments.

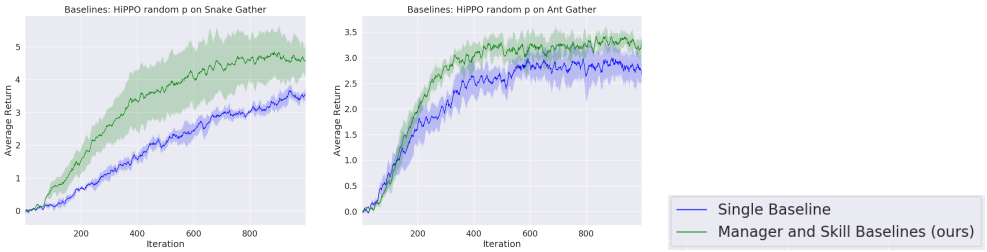

Figure 4: Effect of using a Skill baseline as defined in Section 5.2

6.2 Learning from Scratch

In this section, we study the benefit of using the HiPPO algorithm instead of standard PPO on a flat policy [15]. The results, shown in Figure 3, demonstrate that training from scratch with HiPPO leads faster learning and better performance than flat PPO. Furthermore, the benefit of HiPPO does not just come from having temporally correlated exploration, as PPO with action repeat converges at a performance level well below our method. Finally, Figure 4 shows the effectiveness of using the presented baseline.

6.3 Robustness to Dynamics Perturbations

We try several different modifications to the base Snake Gather and Ant Gather environments. One at a time, we change the body mass, dampening of the joints, body inertia, and friction characteristics of both robots. The results, presented in Table 1, show that HiPPO with randomized period $\texttt{Categorical}([T_{\min}, T_{\max}])$ not only learns faster initially on the original task, but it is also able to better handle these dynamics changes. In terms of the percent change in policy performance between the training environment and test environment, it is able to outperform HiPPO with fixed period on 6 out of 8 related tasks without even taking any gradient steps. Our hypothesis is that the randomized period teaches the policy to adapt to wide variety of scenarios, while its information bottleneck is able to keep separate its representations for planning and locomotion, so changes in dynamics aren't able to simultaneously affect both.

Gather	Algorithm	Initial	Mass	Dampening	Inertia	Friction
Snake	Flat PPO	2.72	3.16 (+16%)	2.75 (+1%)	2.11 (-22%)	2.75 (+1%)
	HiPPO, $p = 10$	4.38	3.28 (-25%)	3.27 (-25%)	3.03 (-31%)	3.27 (-25%)
	HiPPO random p	**5.11**	**4.09** (-20%)	**4.03** (-21%)	**3.21** (-37%)	**4.03** (-21%)
Ant	Flat PPO	2.25	2.53 (+12%)	2.13 (-5%)	2.36 (+5%)	1.96 (-13%)
	HiPPO, $p = 10$	**3.84**	3.31 (-14%)	**3.37** (-12%)	2.88 (-25%)	**3.07** (-20%)
	HiPPO random p	3.22	**3.37** (+5%)	2.57 (-20%)	**3.36** (+4%)	2.84 (-12%)

Table 1: Zero-shot transfer performance of flat PPO, HiPPO, and HiPPO with randomized period. The performance in the initial environment is shown, as well as the average performance over 25 rollouts in each new modified environment.

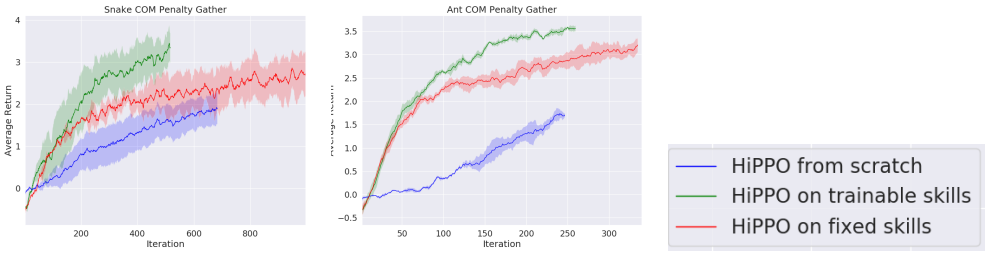

Figure 5: Benefit of adapting some given skills when the preferences of the environment are different from those of the environment where the skills were originally trained.

6.4 Adaptation of Pre-Trained Skills

For this task, we take 6 pre-trained subpolicies encoded by a Stochastic Neural Network [40] that were trained in a diversity-promoting environment [11]. We fine-tune them with HiPPO on the Gather environment, but with an extra penalty on the velocity of the Center of Mass. This can be understood as a preference for cautious behavior. This requires adjustment of the sub-policies, which were trained with a proxy reward encouraging them to move as far as possible (and hence quickly). Fig. 5 shows the difference between fixing the sub-policies and only training a manager with PPO vs using HiPPO to simultaneously train a manager and fine-tune the skills. The two initially learn at the same rate, but HiPPO's ability to adjust to the new dynamics allows it to reach a higher final performance.

6.5 Skill Diversity Assumption

In Lemma 1, we assumed that the sub-policies present ought to be diverse. This allowed us to derive a more efficient and numerically stable gradient. In this section, we empirically test the validity of our assumption, as well as the quality of our approximation. For this we run, on Snake Gather and Ant Gather, the HiPPO algorithm both from scratch and on some pretrained skills as described in the previous section. In Table 2, we report the average maximum probability under other sub-policies, corresponding to ϵ from the assumption. We observe that in all settings this is on the order of magnitude of 0.1. Therefore, under the $p = 10$ that we use in our experiments, the term we neglect has a factor $\epsilon^{p-1} = 10^{-10}$. It is not surprising then that the average cosine similarity between the full gradient and the approximated one is almost 1, as also reported in Table 2. We only ran two random seeds of these experiments, as the results seemed pretty consistent, and they are more computationally challenging to run.

Gather	**Algorithm**	**Cosine Similarity**	$\max_{z' \neq z_{kp}} \pi_{\theta_l}(a_t \vert o_t, z')$
Snake	Adapt given skills	0.98 ± 0.01	0.09 ± 0.04
	HiPPO	0.97 ± 0.03	0.12 ± 0.03
Ant	Adapt given skills	0.96 ± 0.04	0.11 ± 0.05
	HiPPO	0.94 ± 0.03	0.13 ± 0.05

Table 2: Empirical evaluation of Lemma 1. On the right column we evaluate the quality of our assumption by computing what is the average largest probability of a certain action under other skills. On the left column we report cosine similarity between our approximate gradient and the gradient computed using Eq. 2 without approximation.

7 Conclusions and Future Work

In this paper, we examined how to effectively adapt hierarchical policies. We began by deriving a hierarchical policy gradient and approximation of it. We then proposed a new method, HiPPO, that can stably train multiple layers of a hierarchy. The adaptation experiments suggested that we can optimize pretrained skills for downstream environments, and learn emergent skills without any unsupervised pre-training. We also explored hierarchy from an information bottleneck point of view, demonstrating that HiPPO with randomized period can learn from scratch on sparse-reward and long time horizon tasks, while outperforming non-hierarchical methods on zero-shot transfer.

There are many enticing avenues of future work. For instance, replacing the manually designed bottleneck with a variational autoencoder with an information bottleneck could further improve HiPPO's performance and extend the gains seen here to other tasks. Also, as HiPPO provides a policy architecture and gradient expression, we could explore using meta-learning on top of it in order to learn better skills that are more useful on a distribution of different tasks.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei a Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[2] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Van Den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 10 2017. ISSN 14764687. doi: 10.1038/nature24270. URL http://arxiv.org/abs/1610.00633.

[3] John Schulman, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust Region Policy Optimization. *International Conference in Machine Learning*, 2015.

[4] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin Riedmiller, and David Silver. Emergence of Locomotion Behaviours in Rich Environments. 7 2017. URL http://arxiv.org/abs/1707.02286.

[5] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic Goal Generation for Reinforcement Learning Agents. *International Conference in Machine Learning*, 2018. URL http://arxiv.org/abs/1705.06366.

[6] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse Curriculum Generation for Reinforcement Learning. *Conference on Robot Learning*, pages 1–16, 2017. ISSN 1938-7228. doi: 10.1080/00908319208908727. URL http://arxiv.org/abs/1707.05300.

[7] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, and Alex Ray. Learning Dexterous In-Hand Manipulation. pages 1–27.

[8] Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual Reinforcement Learning with Imagined Goals. *Adavances in Neural Information Processing Systems*, 2018.

[9] Carlos Florensa, Jonas Degrave, Nicolas Heess, Jost Tobias Springenberg, and Martin Riedmiller. Self-supervised Learning of Image Embedding for Continuous Control. In *Workshop on Inference to Control at NeurIPS*, 2018. URL http://arxiv.org/abs/1901.00943.

[10] Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999. URL http://www-anw.cs.umass.edu/~barto/courses/cs687/Sutton-Precup-Singh-AIJ99.pdf.

[11] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic Neural Networks for Hierarchical Reinforcement Learning. *International Conference in Learning Representations*, pages 1–17, 2017. ISSN 14779129. doi: 10.1002/rcm.765. URL http://arxiv.org/abs/1704.03012.

[12] Doina Precup. Temporal abstraction in reinforcement learning, 1 2000. URL https://scholarworks.umass.edu/dissertations/AAI9978540.

[13] Peter Dayan and Geoffrey E. Hinton. Feudal Reinforcement Learning. *Advances in Neural Information Processing Systems*, page 271–278, 1993. ISSN 0143991X. doi: 10.1108/IR-08-2017-0143. URL http://www.cs.toronto.edu/~fritz/absps/dh93.pdf.

[14] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta Learning Shared Hierarchies. *International Conference in Learning Representations*, pages 1–11, 2018. ISSN 14639076. doi: 10.1039/b203755f. URL http://arxiv.org/abs/1710.09767.

[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. 2017. URL https://openai-public.s3-us-west-2.amazonaws.com/blog/2017-07/ppo/ppo-arxiv.pdf.

[16] Tejas D Kulkarni, Karthik R Narasimhan, Ardavan Saeedi CSAIL, and Joshua B Tenenbaum BCS. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. *Advances in Neural Information Processing Systems*, pages 1–13, 2016.

[17] Christian Daniel, Herke van Hoof, Jan Peters, Gerhard Neumann, Thomas Gärtner, Mirco Nanni, Andrea Passerini, and Celine B Robardet Christian Daniel ChristianDaniel. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104(104), 2016. doi: 10.1007/s10994-016-5580-x.

[18] Hoang M Le, Nan Jiang, Alekh Agarwal, Miroslav Dud, and Yue Hal. Hierarchical Imitation and Reinforcement Learning. *International Conference in Machine Learning*, 2018.

[19] Josh Merel, Arun Ahuja, Vu Pham, Saran Tunyasuvunakool, Siqi Liu, Dhruva Tirumala, Nicolas Heess, and Greg Wayne. Hierarchical visuomotor control of humanoids. *International Conference in Learning Representations*, 2019. URL http://arxiv.org/abs/1811.09656.

[20] Pravesh Ranchod, Benjamin Rosman, and George Konidaris. Nonparametric Bayesian Reward Segmentation for Skill Discovery Using Inverse Reinforcement Learning. 2015. ISSN 21530866. doi: 10.1109/IROS.2015.7353414.

[21] Arjun Sharma, Mohit Sharma, Nicholas Rhinehart, and Kris M Kitani. Directed-Info GAIL: Learning Hierarchical Policies from Unsegmented Demonstrations using Directed Information. *International Conference in Learning Representations*, 2018. URL http://arxiv.org/abs/1810.01266.

[22] Jacob Andreas, Dan Klein, and Sergey Levine. Modular Multitask Reinforcement Learning with Policy Sketches. *International Conference in Machine Learning*, 2017. URL http://github.com/.

[23] Mohammad Ghavamzadeh and Sridhar Mahadevan. Hierarchical Policy Gradient Algorithms. *International Conference in Machine Learning*, 2003. URL http://chercheurs.lille.inria.fr/~ghavamza/my_website/Publications_files/icml03.pdf.

[24] Sungryull Sohn, Junhyuk Oh, and Honglak Lee. Multitask Reinforcement Learning for Zero-shot Generalization with Subtask Dependencies. *Advances in Neural Information Processing Systems*, 2018.

[25] Ofir Nachum, Honglak Lee, Shane Gu, and Sergey Levine. Data-Efficient Hierarchical Reinforcement Learning. *Advances in Neural Information Processing Systems*, 2018.

[26] Andrew Levy, Robert Platt, and Kate Saenko. Hierarchical Reinforcement Learning with Hindsight. *International Conference on Learning Representations*, 5 2019. URL http://arxiv.org/abs/1805.08180.

[27] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal Networks for Hierarchical Reinforcement Learning. *International Conference in Machine Learning*, 2017. URL https://arxiv.org/pdf/1703.01161.pdf.

[28] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an Embedding Space for Transferable Robot Skills. *International Conference in Learning Representations*, pages 1–16, 2018.

[29] Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, David Silver, and Google Deepmind. Learning and Transfer of Modulated Locomotor Controllers. 2016. URL https://arxiv.org/abs/1610.05182.

[30] Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement Learning. *International Conference in Learning Representations*, 3:1–13, 2018. doi: 10.1109/MWC.2016.7553036.

[31] Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent Space Policies for Hierarchical Reinforcement Learning. *Internation Conference in Machine Learning*, 2018. URL http://arxiv.org/abs/1804.02808.

[32] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The Option-Critic Architecture. *AAAI*, pages 1726–1734, 2017. URL http://arxiv.org/abs/1609.05140.

[33] Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When Waiting is not an Option : Learning Options with a Deliberation Cost. *AAAI*, 9 2017. URL http://arxiv.org/abs/1709.04571.

[34] Alexander Vezhnevets, Volodymyr Mnih, John Agapiou, Simon Osindero, Alex Graves, Oriol Vinyals, and Koray Kavukcuoglu Google DeepMind. Strategic Attentive Writer for Learning Macro-Actions. *Advances in Neural Information Processing Systems*, 2016.

[35] Ronald J Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4):229–256, 1992.

[36] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is All You Need: Learning Skills without a Reward Function. *International Conference in Learning Representations*, 2019. URL http://arxiv.org/abs/1802.06070.

[37] Jan Peters and Stefan Schaal. Natural Actor-Critic. *Neurocomputing*, 71(7-9):1180–1190, 2008. ISSN 09252312. doi: 10.1016/j.neucom.2007.11.026.

[38] George Konidaris and Andrew Barto. Building portable options: Skill transfer in reinforcement learning. *International Joint Conference on Artificial Intelligence*, pages 895–900, 2007. ISSN 10450823. doi: 10.1158/1078-0432.CCR-05-1323.

[39] Yan Duan, Xi Chen, John Schulman, and Pieter Abbeel. Benchmarking Deep Reinforcement Learning for Continuous Control. *International Conference in Machine Learning*, 2016. URL http://arxiv.org/abs/1604.06778.

[40] Yichuan Tang and Ruslan Salakhutdinov. Learning Stochastic Feedforward Neural Networks. *Advances in Neural Information Processing Systems*, 2:530–538, 2013. doi: 10.1.1.63.1777.

A Hyperparameters and Architectures

For all experiments, both PPO and HiPPO used learning rate 3×10^{-3}, clipping parameter $\epsilon = 0.1$, 10 gradient updates per iteration, a batch size of 100,000, and discount $\gamma = 0.999$. HiPPO used $n = 6$ sub-policies. Ant Gather has a horizon of 5000, while Snake Gather has a horizon of 8000 due to its larger size. All runs used three random seeds. HiPPO uses a manager network with 2 hidden layers of 32 units, and a skill network with 2 hidden layers of 64 units. In order to have roughly the same number of parameters for each algorithm, flat PPO uses a network with 2 hidden layers with 256 and 64 units respectively. For HiPPO with randomized period, we resample $p \sim \text{Uniform}\{5, 15\}$ every time the manager network outputs a latent, and provide the number of timesteps until the next latent selection as an input into both the manager and skill networks. The single baselines and skill-dependent baselines used a MLP with 2 hidden layers of 32 units to fit the value function. The skill-dependent baseline receives, in addition to the full observation, the active latent code and the time remaining until the next skill sampling.

B Proofs

Lemma 1. If the skills are sufficiently differentiated, then the latent variable can be treated as part of the observation to compute the gradient of the trajectory probability. Concretely, if $\pi_{\theta_h}(z|s)$ and $\pi_{\theta_l}(a|s,z)$ are Lipschitz in their parameters, and $0 < \pi_{\theta_l}(a_t|s_t, z_j) < \epsilon \; \forall j \neq kp$, then

$$\nabla_\theta \log P(\tau) = \sum_{k=0}^{H/p} \nabla_\theta \log \pi_{\theta_h}(z_{kp}|s_{kp}) + \sum_{t=1}^{p} \nabla_\theta \log \pi_{\theta_l}(a_t|o_t, z_{kp}) + \mathcal{O}(nH\epsilon^{p-1}) \tag{4}$$

Proof. From the point of view of the MDP, a trajectory is a sequence $\tau = (s_0, a_0, s_1, a_1, \ldots, a_{H-1}, s_H)$. Let's assume we use the hierarchical policy introduced above, with a higher-level policy modeled as a parameterized discrete distribution with n possible outcomes $\pi_{\theta_h}(z|s) = Categorical_{\theta_h}(n)$. We can expand $P(\tau)$ into the product of policy and environment dynamics terms, with z_j denoting the jth possible value out of the n choices,

$$P(\tau) = \Bigg(\prod_{k=0}^{H/p} \Big[\sum_{j=1}^{n} \pi_{\theta_h}(z_j|s_{kp}) \prod_{t=kp}^{(k+1)p-1} \pi_{\theta_l}(a_t|s_t, z_j) \Big] \Bigg) \Bigg[P(s_0) \prod_{t=1}^{H} P(s_{t+1}|s_t, a_t) \Bigg]$$

Taking the gradient of $\log P(\tau)$ with respect to the policy parameters $\theta = [\theta_h, \theta_l]$, the dynamics terms disappear, leaving:

$$\begin{aligned}
\nabla_\theta \log P(\tau) &= \sum_{k=0}^{H/p} \nabla_\theta \log \Big(\sum_{j=1}^{n} \pi_{\theta_l}(z_j|s_{kp}) \prod_{t=kp}^{(k+1)p-1} \pi_{s,\theta}(a_t|s_t, z_j) \Big) \\
&= \sum_{k=0}^{H/p} \frac{1}{\sum_{j=1}^{n} \pi_{\theta_h}(z_j|s_{kp}) \prod_{t=kp}^{(k+1)p-1} \pi_{\theta_l}(a_t|s_t, z_j)} \sum_{j=1}^{n} \nabla_\theta \Big(\pi_{\theta_h}(z_j|s_{kp}) \prod_{t=kp}^{(k+1)p-1} \pi_{\theta_l}(a_t|s_t, z_j) \Big)
\end{aligned}$$

The sum over possible values of z prevents the logarithm from splitting the product over the p-step sub-trajectories. This term is problematic, as this product quickly approaches 0 as p increases, and suffers from considerable numerical instabilities. Instead, we want to approximate this sum of products by a single one of the terms, which can then be decomposed into a sum of logs. For this we study each of the terms in the sum: the gradient of a sub-trajectory probability under a specific latent $\nabla_\theta \Big(\pi_{\theta_h}(z_j|s_{kp}) \prod_{t=kp}^{(k+1)p-1} \pi_{\theta_l}(a_t|s_t, z_j) \Big)$. Now we can use the assumption that the skills are easy to distinguish, $0 < \pi_{\theta_l}(a_t|s_t, z_j) < \epsilon \; \forall j \neq kp$. Therefore, the probability of the sub-trajectory under a latent different than the one that was originally sampled $z_j \neq z_{kp}$, is upper bounded by ϵ^p. Taking the gradient, applying the product rule, and the Lipschitz continuity of the policies, we obtain that for all $z_j \neq z_{kp}$,

$$\nabla_\theta\Big(\pi_{\theta_h}(z_j|s_{kp})\prod_{t=kp}^{(k+1)p-1}\pi_{\theta_l}(a_t|s_t,z_j)\Big) = \nabla_\theta\pi_{\theta_h}(z_j|s_{kp})\prod_{t=kp}^{(k+1)p-1}\pi_{\theta_l}(a_t|s_t,z_j) +$$
$$\sum_{t=kp}^{(k+1)p-1}\pi_{\theta_h}(z_j|s_{kp})\big(\nabla_\theta\pi_{\theta_l}(a_t|s_t,z_j)\big)\prod_{\substack{t=kp\\t'\neq t}}^{(k+1)p-1}\pi_{\theta_l}(a_{t'}|s_{t'},z_j)$$
$$= \mathcal{O}(p\epsilon^{p-1})$$

Thus, we can across the board replace the summation over latents by the single term corresponding to the latent that was sampled at that time.

$$\nabla_\theta \log P(\tau) = \sum_{k=0}^{H/p}\frac{1}{\pi_{\theta_h}(z_{kp}|s_{kp})\prod_{t=kp}^{(k+1)p-1}\pi_{\theta_l}(a_t|s_t,z_{kp})}\nabla_\theta\Big(P(z_{kp}|s_{kp})\prod_{t=kp}^{(k+1)p-1}\pi_{\theta_l}(a_t|s_t,z_{kp})\Big) + \frac{nH}{p}\mathcal{O}(p\epsilon^{p-1})$$
$$= \sum_{k=0}^{H/p}\nabla_\theta\log\Big(\pi_{\theta_h}(z_{kp}|s_{kp})\prod_{t=kp}^{(k+1)p-1}\pi_{\theta_l}(a_t|s_t,z_{kp})\Big) + \mathcal{O}(nH\epsilon^{p-1})$$
$$= \mathbb{E}_\tau\Big[\Big(\sum_{k=0}^{H/p}\nabla_\theta\log\pi_{\theta_h}(z_{kp}|s_{kp}) + \sum_{t=1}^{H}\nabla_\theta\log\pi_{\theta_l}(a_t|s_t,z_{kp})\Big)\Big] + \mathcal{O}(nH\epsilon^{p-1})$$

Interestingly, this is exactly $\nabla_\theta P(s_0, z_0, a_0, s_1, \dots)$. In other words, it's the gradient of the probability of that trajectory, where the trajectory now includes the variables z as if they were observed.

□

Lemma 2. For any functions $b_h : \mathcal{S} \to \mathbb{R}$ and $b_l : \mathcal{S} \times \mathcal{Z} \to \mathbb{R}$ we have:

$$\mathbb{E}_\tau[\sum_{k=0}^{H/p}\nabla_\theta\log P(z_{kp}|s_{kp})b(s_{kp})] = 0$$

$$\mathbb{E}_\tau[\sum_{t=0}^{H}\nabla_\theta\log\pi_{s,\theta}(a_t|s_t,z_{kp})b(s_t,z_{kp})] = 0$$

Proof. We can use the law of iterated expectations as well as the fact that the interior expression only depends on s_{kp} and z_{kp}:

$$\mathbb{E}_\tau[\sum_{k=0}^{H/p}\nabla_\theta\log P(z_{kp}|s_{kp})b(s_{kp})] = \sum_{k=0}^{H/p}\mathbb{E}_{s_{kp},z_{kp}}[\mathbb{E}_{\tau\setminus s_{kp},z_{kp}}[\nabla_\theta\log P(z_{kp}|s_{kp})b(s_{kp})]]$$
$$= \sum_{k=0}^{H/p}\mathbb{E}_{s_{kp},z_{kp}}[\nabla_\theta\log P(z_{kp}|s_{kp})b(s_{kp})]$$

Then, we can write out the definition of the expectation and undo the gradient-log trick to prove that the baseline is unbiased.

$$\begin{aligned}
\mathbb{E}_\tau[\sum_{k=0}^{H/p} \nabla_\theta \log \pi_{\theta_h}(z_{kp}|s_{kp})b(s_{kp})] &= \sum_{k=0}^{H/p} \int_{(s_{kp},z_{kp})} P(s_{kp}, z_{kp}) \nabla_\theta \log \pi_{\theta_h}(z_{kp}|s_{kp})b(s_{kp}) dz_{kp} ds_{kp} \\
&= \sum_{k=0}^{H/p} \int_{s_{kp}} P(s_{kp})b(s_{kp}) \int_{z_{kp}} \pi_{\theta_h}(z_{kp}|s_{kp}) \nabla_\theta \log \pi_{\theta_h}(z_{kp}|s_{kp}) dz_{kp} ds_{kp} \\
&= \sum_{k=0}^{H/p} \int_{s_{kp}} P(s_{kp})b(s_{kp}) \int_{z_{kp}} \pi_{\theta_h}(z_{kp}|s_{kp}) \frac{1}{\pi_{\theta_h}(z_{kp}|s_{kp})} \nabla_\theta \pi_{\theta_h}(z_{kp}|s_{kp}) dz_{kp} ds_{kp} \\
&= \sum_{k=0}^{H/p} \int_{s_{kp}} P(s_{kp})b(s_{kp}) \nabla_\theta \int_{z_{kp}} \pi_{\theta_h}(z_{kp}|s_{kp}) dz_{kp} ds_{kp} \\
&= \sum_{k=0}^{H/p} \int_{s_{kp}} P(s_{kp})b(s_{kp}) \nabla_\theta 1 ds_{kp} \\
&= 0
\end{aligned}$$

□

Subtracting a state- and subpolicy- dependent baseline from the second term is also unbiased, i.e.

$$\mathbb{E}_\tau[\sum_{t=0}^{H} \nabla_\theta \log \pi_{s,\theta}(a_t|s_t, z_{kp})b(s_t, z_{kp})] = 0$$

We'll follow the same strategy to prove the second equality: apply the same law of iterated expectations trick, express the expectation as an integral, and undo the gradient-log trick.

$$\begin{aligned}
&\mathbb{E}_\tau[\sum_{t=0}^{H} \nabla_\theta \log \pi_{\theta_l}(a_t|s_t, z_{kp})b(s_t, z_{kp})] \\
&\quad= \sum_{t=0}^{H} \mathbb{E}_{s_t,a_t,z_{kp}}[\mathbb{E}_{\tau \setminus s_t,a_t,z_{kp}}[\nabla_\theta \log \pi_{\theta_m}(a_t|s_t, z_{kp})b(s_t, z_{kp})]] \\
&\quad= \sum_{t=0}^{H} \mathbb{E}_{s_t,a_t,z_{kp}}[\nabla_\theta \log \pi_{\theta_l}(a_t|s_t, z_{kp})b(s_{kp}, z_{kp})] \\
&\quad= \sum_{t=0}^{H} \int_{(s_t,z_{kp})} P(s_t, z_{kp})b(s_t, z_{kp}) \int_{a_t} \pi_{\theta_l}(a_t|s_t, z_{kp}) \nabla_\theta \log \pi_{\theta_l}(a_t|s_t, z_{kp}) da_t dz_{kp} ds_t \\
&\quad= \sum_{t=0}^{H} \int_{(s_t,z_{kp})} P(s_t, z_{kp})b(s_t, z_{kp}) \nabla_\theta 1 dz_{kp} ds_t \\
&\quad= 0
\end{aligned}$$