Sub-policy Adaptation for Hierarchical Reinforcement Learning

Anonymous Authors[1]

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, and introduce Hierarchical Proximal Policy Optimization (HiPPO) 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.

1. Introduction

Humans are able to do multi-task learning exceptionally well, reusing and adapting skills between settings. Yet, transfer learning in Reinforcement Learning (RL) remains an open problem. RL has made great progress in a variety of domains, from playing games such as Pong and Go (Mnih et al., 2015; Silver et al., 2017) to automating robotic locomotion (Schulman et al., 2015; Heess et al., 2017; Florensa et al., 2018b), dexterous manipulation (Florensa et al., 2017b; Andrychowicz et al.), and perception (Nair et al., 2018; Florensa et al., 2018a). Yet most work in RL is still learning a new behavior from scratch for every problem. This is particularly inefficient when dealing with tasks hard to solve, with sparse rewards and long horizons, or when solving many related tasks.

A promising technique to overcome this limitation is Hierarchical Reinforcement Learning (HRL). In this paradigm, policies have several layers of abstraction, and hence the reuse of one layer becomes easier. The most common case consists of temporal abstraction, 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 work 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 in the new task. Little work has been done on adapting pretrained subpolicies to be optimal for a new task.

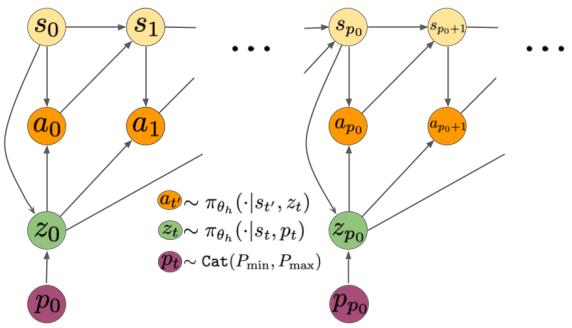


Figure 1. Temporal hierarchy studied in this paper. A latent code z_t is sampled from the manager policy $\pi_{\theta_h}(z_t|s_t)$ every p time-steps, using the current observation s_{kp}. The actions a_t are sampled from the sub-policy $\pi_{\theta_l}(a_t|s_t, z_{kp})$ conditioned on the same latent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

In this paper, we develop several new techniques for finetuning temporal hierarchies. First, we develop an approximated hierarchical policy gradient. Then we introduce a more stable way of using this gradient, Hierarchical Proximal Policy Optimization (HiPPO). This helps us take good steps in our policy space, 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. Preliminaries

We define a discrete-time finite-horizon discounted Markov decision process (MDP) by a tuple $M =$

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

$(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma, H)$, 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 a transition probability distribution, $\gamma \in [0, 1]$ is a discount factor, and H is 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 $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$.

3. Problem Statement

In this work we tackle the problem of training a hierarchical policy, and efficiently adapting the different parts of the policy to perform a new task. Usual 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 doesn't take actions in the environment directly, but rather outputs a command, or latent variable z, that conditions the behavior of the lower level. In our work we use a temporal abstraction, meaning that the manager only observes the environment every p time-steps, and in between higher level actions the latent code z conditioning the lower level is kept fixed, as depicted in Fig. 1.

One of the most interesting benefits of Hierarchical Reinforcement Learning (HRL) is the potential to leverage previously acquired knowledge to accelerate learning in similar tasks. Specifically we focus on the re-use of the lower level. A very common approach is to keep the sub-policies fixed, and train a new manager on top, which encourages meaningful exploration. Nevertheless, it also brings considerable suboptimality on the final performance because these sub-policies were trained on another task. Our goal is to propose a method that can continuously adapt the skills, while preventing the sub-policy collapse observed in other methods when training all levels of the hierarchy together. For a detailed literature review see the Appendix.

4. Methodology

From the Markovian environment perspective, the trajectories are sequences of states and actions $\tau = (s_0, a_0, s_1, \ldots, s_H)$. 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 this variable should be accounted for: as an observed variable like the action, or as a latent? Our first Lemma states that the former is an approximation of the later. This finding is then incorporated into the state-of-the-art on-policy algorithm PPO (Schulman et al., 2017). Finally, we introduce the notion of information bottleneck and trajectory compression that will be critical for learning reusable skills.

4.1. Approximate Hierarchical Policy Gradient

In this subsection, we want to avoid numerical instability from a product of many action probabilities.

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*

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

Proof. See Appendix. Many algorithms to extract lower-level skills are based on promoting diversity among the skills (Florensa et al., 2017a; Eysenbach et al., 2019), therefore usually satisfying our assumption! □

With this result we can already use REINFORCE (Williams, 1992) to compute a policy gradient $g = \mathbb{E}_\tau\left[\nabla_\theta \log P(\tau) r(\tau)\right]$. In the next sub-section we will show how state-latent conditioned baselines can be subtracted without biasing the gradient. This will be crucial for variance reduction later on.

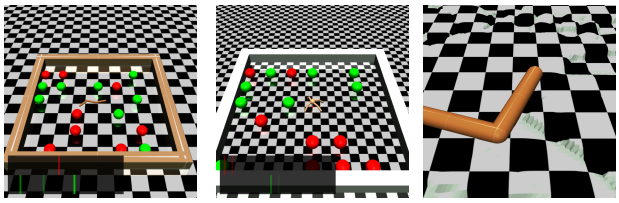
4.2. Hierarchical Proximal Policy Optimization

Hierarchical policy gradient still faces many of the same challenges as vanilla policy gradient, such as taking good step sizes in the policy space. Certain gradient updates will change the policy very little, while others can hurt policy training by changing its behavior drastically.

Trust region policy optimization (TRPO) uses constraints on the KL-divergence between the old policy and the new policy to avoid this (Schulman et al., 2015). Unfortunately, hierarchical policies are generally represented by complex distributions without closed form expressions for the KL-divergence. Instead, we modify a method called Proximal Policy Optimization (Schulman et al., 2017), which replaces the KL-divergence constraint with a cost function that accomplishes the same objective, and 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 directly 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 $r_{l,t}(\theta) = \frac{\pi_{\theta_l}(a_t|s_t, z_{kp})}{\pi_{\theta_{l,old}}(a_t|s_t, z_{kp})}$,



(a) SwimmerGather environment. (b) AntGather environment. (c) Rough terrain in SwimmerGather

Figure 2. SwimmerGather and AntGather environments. SwimmerGatherHField differs from SwimmerGather only in that its floor is rough and uneven.

and using the super-index `clip` to denote their clipped version, we obtain the new surrogate objective:

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

We call this algorithm HiPPO. It is similar to the one proposed by Frans et al. (2018), although they only use it in a meta-learning setup. Therefore their method can also benefit from our theoretical framework and insight. Furthermore, we include two critical additions: the addition of an information bottleneck at the lower-level, and a switching of the time-commitment to the skills. Both are detailed in the following subsections.

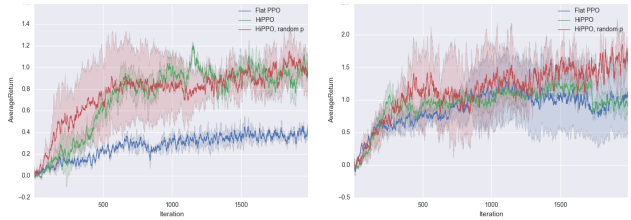
4.3. Information Bottleneck and Compression

If we apply the above HiPPO gradient 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 has to restrict the information about the task to be performed, such that a single skill cannot perform the full task. Skill collapse is a common problem in hierarchical methods, requiring the use of regularizers (Bacon et al., 2017; Harb et al., 2017; Vezhnevets et al., 2016). There are some natural choices of f discussed in the robotics literature, like the agent-space and problem-space split (Konidaris & Barto, 2007; Florensa et al., 2017a), that only allows the sub-policies to see the proprioceptive information and hides all task-related 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 type of compression.

Figure 3. Learning curves for HiPPO vs PPO on the SwimmerGatherHField environment. Both use the same initial skills, trained using a mutual information bonus on a smooth terrain.



(a) Swimmer Gather (b) Ant Gather

Figure 4. Comparison of Flat PPO, HiPPO, and HiPPO with randomized period learning from scratch on different environments.

4.4. Varying Time-commitment

Most hierarchical methods either consider a fixed time-commitment to the lower level skills, or implement the complex option framework (Precup, 2000). In this work we propose an in-between, where the time-commitment to the skills is a random variable sampled just before the manager takes a decision. We show that training with this modification yields better final performance, and improves zero-shot adaptation to a new task. The full algorithm is detailed in the Appendix.

5. Experiments

5.1. Tasks

We evaluate our new algorithms on a variety of robotic navigation tasks. Each task is a different robot trying to solve the Gather environment (Duan et al., 2016), 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. Swimmer is a 2-joint snake-like robot depicted in Fig. 2a, with a 15 dimensional observation space and 2 dimensional action space. Ant is a quadrupedal robot, depicted in Fig. 2b, 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.

Gather	Algorithm	Initial	Mass	DOF	Inertia	Friction	Crippled Leg
Swimmer	Flat PPO	0.42	0.23 (-46%)	0.33 (-20%)	0.15 (-64%)	0.30 (-29%)	
	HiPPO	**1.02**	0.54 (-47%)	0.72 (-29%)	0.46 (-55%)	0.77 (-25%)	
	HiPPO rand p	0.92	**0.73** (-21%)	**0.87** (-5%)	**0.81** (-12%)	**0.91** (-0.8%)	
Ant	Flat PPO	1.06	0.70 (-34%)	0.76 (-28%)	0.77 (-27%)	0.70 (-35%)	-0.04 (-104%)
	HiPPO	1.03	1.09 (6%)	1.06 (3%)	1.17 (14%)	0.87 (-16%)	0.04 (-96%)
	HiPPO rand p	**1.68**	**1.33** (-21%)	**1.57** (-7%)	**1.40** (-17%)	**1.25** (-26%)	**0.19** (-89%)

Table 1. Zero-shot transfer performance of flat PPO, HiPPO, and HiPPO with randomized period.

5.2. Adapting SwimmerGather Skills

For this task, we take 6 pretrained subpolicies encoded by a Stochastic Neural Network (Tang & Salakhutdinov, 2013) that were trained in a diversity-promoting environment (Florensa et al., 2017a). We fine-tune them with HiPPO on the SwimmerGatherHField environment, which has a rough terrain shown in Fig. 2c. This requires adjustment of the subpolicies, which were trained on a flat, smooth terrain, as they may not perform as well due to the changed locomotion dynamics. Fig. 3 shows the difference between fixing the subpolicies 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.

5.3. Learning from Scratch and Zero-Shot Transfer

In these experiments, we explore a modification to HiPPO and test its effects on learning from scratch and zero-shot transfer. We sample a new time-commitment period p every time the manager network outputs a latent, instead of keeping p fixed. Interestingly, HiPPO with randomized period actually performs quite well when learning from scratch. As shown in Fig. 4a and Fig. 4b, both HiPPO and HiPPO with randomized period are able to perform at least on par with flat PPO. The reason remains unclear, but perhaps being forced to commit to a skill for p timesteps enhances exploration within these sparse reward tasks.

We try several different modifications to the base Swimmer Gather and Ant Gather environments. One at a time, we change the body mass, degree of freedom dampening, body inertia, and friction characteristics of both robots. In this paper, we choose the first half of these characteristics to multiply by 1.5. We do this so that there's an asymmetric change in the dynamics that cannot be easily solved by multiplying the forces at each joint by the same factor. Finally, we can cripple the Ant robot's front left leg by shrinking and freezing it.

Surprisingly, HiPPO with randomized period not only learns faster initially on the original task, but it also is able to better handle these dynamics changes. It significantly outperforms flat PPO on these zero-shot transfer experiments, as shown in Table 1. It is able to consistently achieve the highest performance on these related tasks without even taking any gradient steps. One 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.

6. 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.

Still, we have many different possible avenues of future work. We only used SNNs to encode our subpolicies for both experiments. How would we perform if we were provided n separate neural networks to finetune? We still need to explore this setting, as finetuning for SNNs has the problem where training one subpolicy will change the behavior of the other. While HiPPO does learn good emergent skills, their transferability is not guaranteed. As HiPPO is simply a policy architecture and gradient expression, we could explore using meta-learning on top in order to learn better skills that are more useful on a distribution of different tasks.

Finally, while the agent-task split discussed in Section 4.3 is intuitive for hierarchical environments such as Gather, it is unclear how to use it in a general setting. Replacing this 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.

References

Andreas, J., Klein, D., and Levine, S. Modular Multitask Reinforcement Learning with Policy Sketches. *International Conference in Machine Learning*, 2017. URL http://github.com/.

Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., and Ray, A. Learning Dexterous In-Hand Manipulation. pp. 1–27.

Bacon, P.-L., Harb, J., and Precup, D. The Option-Critic Architecture. *AAAI*, pp. 1726–1734, 2017. URL http://arxiv.org/abs/1609.05140.

Daniel, C., van Hoof, H., Peters, J., Neumann, G., Gärtner, T., Nanni, M., Passerini, A., and Robardet Christian Daniel ChristianDaniel, C. B. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104(104), 2016. doi: 10.1007/s10994-016-5580-x.

Duan, Y., Chen, X., Schulman, J., and Abbeel, P. Benchmarking Deep Reinforcement Learning for Continuous Control. *International Conference in Machine Learning*, 2016. URL http://arxiv.org/abs/1604.06778.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is All You Need: Learning Skills without a Reward Function. *International Conference in Learning Representations*, 2019. URL http://arxiv.org/abs/1802.06070.

Florensa, C., Duan, Y., and Abbeel, P. Stochastic Neural Networks for Hierarchical Reinforcement Learning. *International Conference in Learning Representations*, pp. 1–17, 2017a. ISSN 14779129. doi: 10.1002/rcm.765. URL http://arxiv.org/abs/1704.03012.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P. Reverse Curriculum Generation for Reinforcement Learning. *Conference on Robot Learning*, pp. 1–16, 2017b. ISSN 1938-7228. doi: 10.1080/00908319208908727. URL http://arxiv.org/abs/1707.05300.

Florensa, C., Degrave, J., Heess, N., Springenberg, J. T., and Riedmiller, M. Self-supervised Learning of Image Embedding for Continuous Control. In *Workshop on Inference to Control at NeurIPS*, 2018a. URL http://arxiv.org/abs/1901.00943.

Florensa, C., Held, D., Geng, X., and Abbeel, P. Automatic Goal Generation for Reinforcement Learning Agents. *International Conference in Machine Learning*, 2018b. URL http://arxiv.org/abs/1705.06366.

Frans, K., Ho, J., Chen, X., Abbeel, P., and Schulman, J. Meta Learning Shared Hierarchies. *International Conference in Learning Representations*, pp. 1–11, 2018. ISSN 14639076. doi: 10.1039/b203755f. URL http://arxiv.org/abs/1710.09767.

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

Haarnoja, T., Hartikainen, K., Abbeel, P., and Levine, S. Latent Space Policies for Hierarchical Reinforcement Learning. *Internation Conference in Machine Learning*, 2018. URL http://arxiv.org/abs/1804.02808.

Harb, J., Bacon, P.-L., Klissarov, M., and Precup, D. When Waiting is not an Option : Learning Options with a Deliberation Cost. *AAAI*, 9 2017. URL http://arxiv.org/abs/1709.04571.

Hausman, K., Springenberg, J. T., Wang, Z., Heess, N., and Riedmiller, M. Learning an Embedding Space for Transferable Robot Skills. *International Conference in Learning Representations*, pp. 1–16, 2018.

Heess, N., Wayne, G., Tassa, Y., Lillicrap, T., Riedmiller, M., Silver, D., and Deepmind, G. Learning and Transfer of Modulated Locomotor Controllers. 2016. URL https://arxiv.org/abs/1610.05182.

Heess, N., TB, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S. M. A., Riedmiller, M., and Silver, D. Emergence of Locomotion Behaviours in Rich Environments. 7 2017. URL http://arxiv.org/abs/1707.02286.

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

Kulkarni, T. D., Narasimhan, K. R., Saeedi CSAIL, A., and Tenenbaum BCS, J. B. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. *Advances in Neural Information Processing Systems*, pp. 1–13, 2016.

Le, H. M., Jiang, N., Agarwal, A., Dud, M., and Hal, Y. Hierarchical Imitation and Reinforcement Learning. *International Conference in Machine Learning*, 2018.

Levy, A., Platt, R., and Saenko, K. Hierarchical Reinforcement Learning with Hindsight. *International Conference on Learning Representations*, 5 2019. URL http://arxiv.org/abs/1805.08180.

Merel, J., Ahuja, A., Pham, V., Tunyasuvunakool, S., Liu, S., Tirumala, D., Heess, N., and Wayne, G. Hierarchical visuomotor control of humanoids. *International Conference in Learning Representations*, 2019. URL http://arxiv.org/abs/1811.09656.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. a., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Nachum, O., Lee, H., Gu, S., and Levine, S. Data-Efficient Hierarchical Reinforcement Learning. *Advances in Neural Information Processing Systems*, 2018.

Nair, A., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual Reinforcement Learning with Imagined Goals. *Adavances in Neural Information Processing Systems*, 2018.

Precup, D. Temporal abstraction in reinforcement learning, 1 2000. URL https://scholarworks.umass.edu/dissertations/AAI9978540.

Ranchod, P., Rosman, B., and Konidaris, G. Nonparametric Bayesian Reward Segmentation for Skill Discovery Using Inverse Reinforcement Learning. 2015. ISSN 21530866. doi: 10.1109/IROS.2015.7353414.

Schulman, J., Moritz, P., Jordan, M., and Abbeel, P. Trust Region Policy Optimization. *International Conference in Machine Learning*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., 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.

Sharma, A., Sharma, M., Rhinehart, N., and Kitani, K. M. 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.

Shu, T., Xiong, C., and Socher, R. 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.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Van Den Driessche, G., Graepel, T., and Hassabis, D. 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.

Sohn, S., Oh, J., and Lee, H. Multitask Reinforcement Learning for Zero-shot Generalization with Subtask Dependencies. *Advances in Neural Information Processing Systems*, 2018.

Sutton, R. S., Precup, D., and Singh, S. 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.

Tang, Y. and Salakhutdinov, R. Learning Stochastic Feedforward Neural Networks. *Advances in Neural Information Processing Systems*, 2:530–538, 2013. doi: 10.1.1.63.1777.

Vezhnevets, A., Mnih, V., Agapiou, J., Osindero, S., Graves, A., Vinyals, O., and Kavukcuoglu Google DeepMind, K. Strategic Attentive Writer for Learning Macro-Actions. *Advances in Neural Information Processing Systems*, 2016.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal Networks for Hierarchical Reinforcement Learning. *International Conference in Machine Learning*, 2017. URL https://arxiv.org/pdf/1703.01161.pdf.

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

A. 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 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 (Sutton et al., 1999; Precup, 2000), but only recently has it been applied to high-dimensional continuous domains as we do in this work (Kulkarni et al., 2016; Daniel et al., 2016).

To obtain the lower level policies, or skills, most methods assume some additional assumptions, like the access to demonstrations (Le et al., 2018; Merel et al., 2019; Ranchod et al., 2015; Sharma et al., 2018), policy sketches (Andreas et al., 2017), or task decomposition into sub-tasks (Ghavamzadeh & Mahadevan, 2003; Sohn et al., 2018). 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 (Nachum et al., 2018; Levy et al., 2019; Vezhnevets et al., 2017). These methods are very promising for state-reaching tasks, but might require access to a 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 (Hausman et al., 2018; Heess et al., 2016). Other works allows to build on previous skills by constantly supplementing the set of skills with new ones (Shu et al., 2018), 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 (Haarnoja et al., 2018), 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 (Sutton et al., 1999), and it has recently been applied to continuous state spaces (Bacon et al., 2017). One of the most delicate parts of this formulation is the termination policy, and it requires several regularizers to avoid the skill collapse (Harb et al., 2017; Vezhnevets et al., 2016). 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 the 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. (2018). Their method can be seen as a special case of ours, and it also 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 information bottleneck to obtain skills that generalize better.

B. Detailed Algorithm

C. Hyperparameters and Architectures

For all experiments, both PPO and HiPPO used a maximum path length of 5000, learning rate 3×10^{-3}, clipping parameter $\epsilon = 0.1$, and 10 gradient updates per iteration. HiPPO used 6 skills and a time-commitment period $p = 10$. All runs used three random seeds. For the experiments in Fig. 3 and Fig. **??**, we used a batch size of 500,000. For Fig. 4a and Fig. 4b, we used a batch size of 100,000. 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 $p \sim \text{Uniform}\{1, 10\}$ 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.

D. 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 the policy distributions $\pi_{\theta_h}(z|s)$ and $\pi_{\theta_l}(a|s, z)$ are

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, bottleneck function $o = f(s)$
Reset environment $s_0 \sim \rho_0$, $t = 0$.
while $t < H$ **do**
  Sample time-commitment $p \sim \texttt{Categorical}(P_{\min}, P_{\max})$
  Sample skill $z_t \sim \pi_{\theta_h}(\cdot|s_t)$
  for $t' = t$ **to** $t + p$ **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 +=p'
end while
Output: $(s_0, z_0, a_0, o_1, a_1, \ldots, s_t, z_t, a_t, o_{t+1}, \ldots)$

Algorithm 2 HiPPO

Input: skills $\pi_{\theta_l}(a|s,z)$, manager $\pi_{\theta_h}(z|s)$, horizon H
for i=1 **to** Iter **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
  $\theta \leftarrow$ Optimize surrogate $L^{CLIP}_{HiPPO}(\theta)$ wrt θ
end for

Lipschitz in their parameters, and $\pi_{\theta_l}(a_t|s_t, z_{kp}) > 1 - \epsilon$, then

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

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,

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

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 $z_j\neq z_{kp}$,

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

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.

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

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 function b 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}:

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

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.

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