EXPLOITING HIERARCHY FOR LEARNING AND TRANSFER IN KL-REGULARIZED RL

Dhruva Tirumala, Hyeonwoo Noh [*], Alexandre Galashov, Leonard Hasenclever, Arun Ahuja, Greg Wayne, Razvan Pascanu, Yee Whye Teh & Nicolas Heess
DeepMind
6 Pancras Square
Kings Cross, London, UK
dhruvat@google.com; shgusdngogo@postech.ac.kr

ABSTRACT

As reinforcement learning agents are tasked with solving more challenging and diverse tasks, the ability to incorporate prior knowledge into the learning system and the ability to exploit reusable structure in solution space is likely to become increasingly important. The KL-regularized expected reward objective constitutes a convenient tool to this end. It introduces an additional component, a default or prior behavior, which can be learned alongside the policy and as such partially transforms the reinforcement learning problem into one of behavior modelling. In this work we consider the implications of this framework in case where both the policy and default behavior are augmented with latent variables. We discuss how the resulting hierarchical structures can be exploited to implement different inductive biases and how the resulting modular structures can be exploited for transfer. Empirically we find that they lead to faster learning and transfer on a range of continuous control tasks.

1 INTRODUCTION

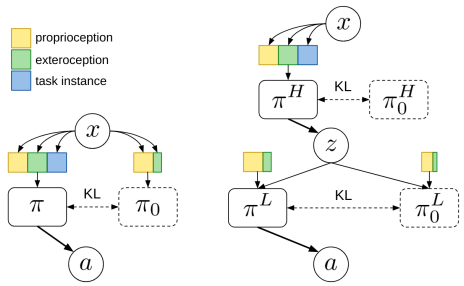
Reinforcement learning approaches, coupled with neural networks as function approximators, have solved an impressive range of tasks, from complex control tasks (Lillicrap et al., 2016; Heess et al., 2017; Riedmiller et al., 2018; Levine et al., 2016; OpenAI et al., 2018) to computer games (Mnih et al., 2015; OpenAI, 2018) and Go (Silver et al., 2016). Recent advances have greatly improved data efficiency, scalability, and stability of algorithms in a variety of domains (Rennie et al., 2017; Zoph & Le, 2017; Espeholt et al., 2018; Ganin et al., 2018; Zhu et al., 2018).

Nevertheless, many tasks remain challenging to solve and require large numbers of interactions with the environment. While the reasons can be hard to pin down they frequently have to do with the fact that solutions are unlikely to be found efficiently by chance when no prior knowledge is available, or that the solution space is dominated by sub-optimal local minima in terms of returns or other desired properties of the behaviour that are somehow not captured by rewards.

The KL-regularized objective (Todorov, 2007; Kappen et al., 2012; Rawlik et al., 2012; Schulman et al., 2017) creates a connection between RL and probabilistic models. It introduces a second component, a prior or default behaviour, and the policy is then encouraged to remain close to it in terms of the Kullback-Leibler (KL) divergence – which can be used to influence the learned policy. Recently, within this framework, (Teh et al., 2017; Czarnecki et al., 2018; Galashov et al., 2019) have proposed to learn a parameterized *default policy* in the role of a more informative prior.

These works suggest an elegant solution for enforcing complex biases that can be also learned or transferred from different tasks. And the objective provides much flexibility in terms of model and algorithm choice. In this work we extend this line of thought, considering the scenario when both the default policy and the agent are hierarchically structured and augmented with latent variables. This provides new mechanisms for restricting the information flow and introducing inductive biases. In addition, we also explore how the resulting modular policies can be used in transfer learning scenarios.

[*]Equal contribution. Work done during internship with co-affiliation to POSTECH, Korea



(a) unstructured policy (b) hierarchical policy

Figure 1: Diagram of the generic structure of the regularized KL-objective considered. (a) shows an unstructured policy, where information asymmetry (e.g. hiding task instance information) is exploited to induce a meaningful default policy π_0 (Galashov et al., 2019); (b) shows the scenario when we use structured policies composed from high-level π^H and low-level π^L policies that communicate through the latent action z. Note that now different forms of information asymmetry can be employed. See text for more details.

We provide empirical results on several tasks with physically simulated bodies and continuous action spaces, highlighting the role of the structured policies.

2 RL AS PROBABILISTIC MODELLING

In this section, we will provide an overview of how the KL-regularized objective connects RL as probabilistic model learning, before developing our approach in the next section. We start by introducing some standard notation. We will denote states and actions at time t respectively with s_t and a_t. $r(s, a)$ is the instantaneous reward received in state s when taking action a. We will refer to the history up to time t as $x_t = (s_1, a_1, .., s_t)$ and the whole trajectory as $\tau = (s_1, a_1, s_2, a_2, \ldots)$. The agent policy $\pi(a_t|x_t)$ denotes a distribution over next actions given history x_t, while $\pi_0(a_t|x_t)$ denotes a default or habitual policy. The KL-regularized RL objective (Todorov, 2007; Kappen et al., 2012; Rawlik et al., 2012; Schulman et al., 2017) takes the form:

$$\mathcal{L}(\pi, \pi_0) = \mathbb{E}_\tau \left[\sum_{t \geq 1} \gamma^t r(s_t, a_t) - \alpha \gamma^t D_{\mathrm{KL}}(a_t|x_t) \right] \tag{1}$$

where we use a convenient notation[1] for the KL divergence: $D_{\mathrm{KL}}(a_t|x_t) = \mathbb{E}_{\pi(a_t|x_t)}[\log \frac{\pi(a_t|x_t)}{\pi_0(a_t|x_t)}]$, γ is the discount factor and α is a hyperparameter controlling the relative contributions of both terms. $\mathbb{E}_\tau[\cdot]$ is taken with respect to the distribution over trajectories defined by the agent policy and system dynamics: $p(s_1) \prod_{t \geq 1} \pi(a_t|x_t) p(s_{t+1}|s_t, a_t)$.

π_0 can be used to inject detailed prior knowledge into the learning problem. In a transfer scenario π_0 can be a *learned* object, and the KL term plays effectively the role of a shaping reward. π and π_0 can also be co-optimized. In this case the relative parametric forms of π_0 and π are of importance. The optimal π_0 in eq. equation 1 is

$$\pi_0^*(a_t|x_t) = \arg\max_{\pi_0} \mathbb{E}_{\pi(a|x_t)}[\log \pi_0(a|x_t)]. \tag{2}$$

(Galashov et al., 2019) details further the interplay between π and π_0.

3 HIERARCHICALLY STRUCTURED POLICIES

Our approach decomposes policies into high-level and low-level components which interact via auxiliary latent variables. Let z_t be a (continuous) latent variable for each time step t. The agent policy is extended as $\pi(a_t, z_t|x_t) = \pi^H(z_t|x_t)\pi^L(a_t|z_t, x_t)$ and likewise for the default policy π_0. z_t can be interpreted as a high-level or abstract action, taken according to the high-level (HL) controller π^H, and which is translated into low-level or motor action a_t by the low-level (LL) controller π^L. We extend the histories x_t and trajectories τ to appropriately include z_t's. Structuring a policy into HL and LL controllers has been studied (e.g. Heess et al., 2016; Hausman et al., 2018; Haarnoja et al., 2018a; Merel et al., 2019), but the concept of default policy has not been widely explored in this context.

In case z_t's can take on many values or are continuous, the objective equation 1 becomes intractable as the marginal distributions $\pi(a_t|x_t)$ and $\pi_0(a_t|x_t)$ in the KL divergence cannot be computed in

[1] In the following, $D_{\mathrm{KL}}(Y|X)$ always denotes $\mathbb{E}_{\pi(Y|X)}[\log \frac{\pi(Y|X)}{\pi_0(Y|X)}]$ for arbitrary variables X and Y.

closed form. This problem can be addressed in different ways. For simplicity and concreteness we here assume that the latent variables in π and π_0 have the same dimension and semantics. We can then construct a lower bound for the objective by using the following upper bound for the KL:

$$D_{\mathrm{KL}}(a_t|x_t) \leq D_{\mathrm{KL}}(z_t|x_t) + \mathbb{E}_{\pi(z_t|x_t)}[D_{\mathrm{KL}}(a_t|z_t, x_t)], \tag{3}$$

which is tractably approximated using Monte Carlo sampling. Note that:

$$D_{\mathrm{KL}}(z_t|x_t) = D_{\mathrm{KL}}\big(\pi^H(z_t|x_t)\|\pi_0^H(z_t|x_t)\big)$$
$$D_{\mathrm{KL}}(a_t|z_t, x_t) = D_{\mathrm{KL}}\big(\pi^L(a_t|z_t, x_t)\|\pi_0^L(a_t|z_t, x_t)\big).$$

The resulting lower bound for the objective is:

$$\mathcal{L}(\pi, \pi_0) \geq \mathbb{E}_\tau\Big[\textstyle\sum_{t\geq 1} \gamma^t r(s_t, a_t) - \alpha\gamma^t D_{\mathrm{KL}}(z_t|x_t) - \alpha\gamma^t D_{\mathrm{KL}}(a_t|z_t, x_t)\Big]. \tag{4}$$

Additionally, we consider sharing low-level controllers in both the agent and the default policy, i.e. $\pi^L(a_t|z_t, x_t) = \pi_0^L(a_t|z_t, x_t)$, for faster learning. This results in a new lower bound:

$$\mathcal{L}(\pi, \pi_0) \geq \mathbb{E}_\tau\left[\textstyle\sum_{t\geq 1} \gamma^t r(s_t, a_t) - \alpha\gamma^t D_{\mathrm{KL}}(z_t|x_t)\right] \tag{5}$$

Note that this objective function is similar in spirit to current KL-regularized RL approaches discussed in Section 2, except that the KL divergence is between policies defined on abstract actions z_t as opposed to concrete actions a_t. The effect of this KL divergence is that it regularizes both the HL policies as well as the space of behaviours parameterised by the abstract actions.

3.1 Information asymmetries and parametrization

We rely on information asymmetry to impose the separation between HL and LL controllers (see Figure 1). Specifically we introduce a separation of concerns between π^L and π^H by providing full information only to π^H while information provided to π^L is limited. In our experiments we vary the information provided π^L; it receives body-specific (proprioceptive) information as well as different amounts of environment-related (exteroceptive) information. The task is only known to π^H. Hiding task specific information from the LL controller makes it easier to transfer across tasks. It forces π^L to focus on learning task agnostic behaviour, and to rely on the abstract actions selected by π^H to solve the task.

Similarly, we hide task specific information from π_0^L, regardless of parameter sharing strategy for LL controllers. Since we also limit the information available to π_0^H , this setup implements a similar default behaviour policy $\pi_0(a_t|x_t)$ as in (Galashov et al., 2019), which can be derived by marginalizing the latents $\int_{z_t} \pi_0^H(z_t|x_t)\pi_0^L(a_t|z_t, x_t)dz_t$.

In the experiments we further consider transferring the HL controller across bodies, in situations where the abstract task is the same but the body changes. Here we additionally hide body-specific information from π^H, so that the HL controller is forced to learn body-agnostic behaviour.

For LL default policy, we use identical parametric forms to implement π^L and π_0^L, regardless of parameter sharing strategy. The specific form of LL controller depends on the experiments. The remaining freedom lies in the choice of the default HL controller $\pi_0^H(z_t|x_t)$. Here, we consider the following choices:

Independent isotropic Gaussian We define the default HL policy as $\pi_0^H(z_t|x_t) = \mathcal{N}(z_t|0, 1)$.

AR(1) process $\pi_0^H(z_t|x_t) = \mathcal{N}(z_t|\alpha z_{t-1}, \sqrt{1-\alpha^2})$, i.e. the default HL policy is a first-order auto-regressive process with a fixed parameter $0 \leq \alpha < 1$ chosen to ensure a marginal distribution $\mathcal{N}(0, 1)$. This allows for more structured temporal dependence among the abstract actions.

Learned AR prior Similar to the AR(1) process this default HL policy allows z_t to depend on z_{t-1} but now the high-level default policy is a Gaussian distribution with mean and variance that are learned functions of z_{t-1} with parameters ϕ: $\pi_0^H(z_t|x_t) = \mathcal{N}(z_t|\mu_\phi(z_{t-1}), \sigma_\phi^2(z_{t-1}))$.

4 Transfer Learning

The hierarchical structure introduces a modularity of the policy and default policy, which can be utilized for transfer learning. We consider two transfer scenarios: 1) task transfer where we reuse the

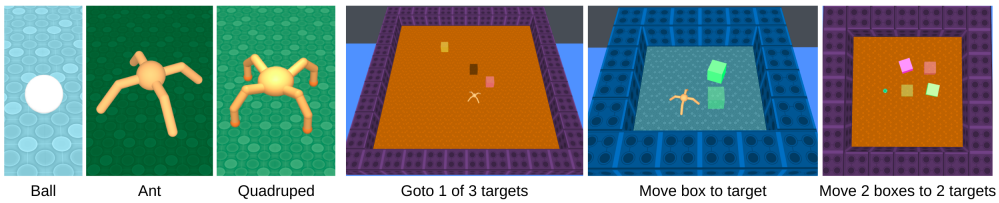


Figure 2: **Bodies and tasks for experiments. Left**: All considered bodies. **Right**: Example tasks.

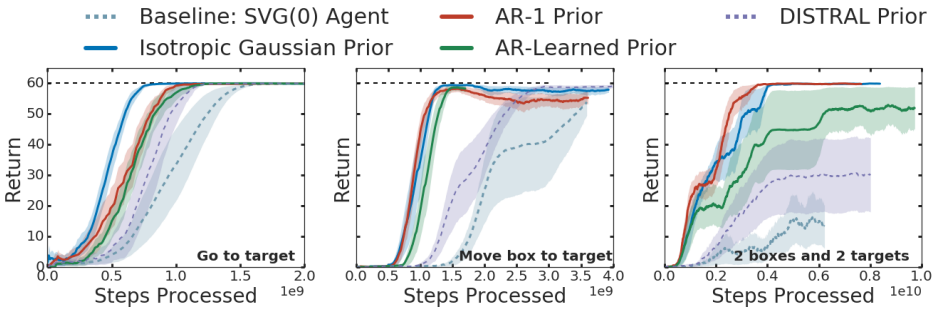


Figure 3: **Speed up for learning complex tasks from scratch. Left**: Go to 1 of 3 targets with the Ant. **Center**: Move a single box to a single target with the Ant. **Right**: Move 2 boxes to 2 targets with the Ball. The proposed model is denoted with the type of HL default policy: Isotropic Gaussian, AR-1, AR-Learned.

learned default policy to solve novel tasks with different goals, and 2) body transfer, where transfering the body agnostic HL controller and default policy to another body.

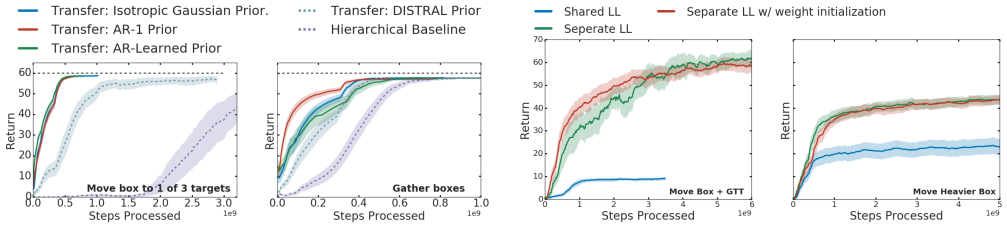
Task transfer For task transfer, we consider task distributions whose solutions exhibit significant shared structure, e.g. because solution trajectories can be produced by a common set of skills or repetitive behaviour. If the default policy can capture and transfer this reusable structure it will facilitate learning similar tasks. Transfer then involves specializing the default behavior to the needs of the target task (e.g. by directing locomotion towards a goal). We reuse pretrained goal agnostic components, including the HL default policy π_0^H and the LL default policy π_0^L, and learn a new HL controller π^H for a target task. In general, we set the LL controller π^L identical to the LL default policy, but for some tasks we allow π^L to diverge from π_0^L. Similarly e.g. to Heess et al. (2016); Hausman et al. (2018), the new HL controller π^H learns to manipulate the LL controller π^L by modulating and interpolating the latent space, while being regularized by π_0^H.

Body transfer Our formulation can also be used for transfer between different bodies which share common behaviour structures for the same task distribution. To transfer the HL structure of a goal-specific behaviour, we reuse the pretrained body-agnostic components, HL controller $\pi^H(z_t|x_t)$ and the default policy $\pi_0^H(z_t|x_t)$, and learn a new body-specific LL controller $\pi^L(a_t|z_t, x_t)$. The transferred HL components provide goal-specific behaviour actuated on the latent space, which can then be instantiated by learning a new LL controller.

5 Experiments

We evaluate our method in several environments with continuous action space and states. We consider a set of structured, sparse reward tasks that can be executed by multiple bodies with different degrees of freedom. The tasks and bodies are illustrated in Figure 2. Details of tasks and bodies used in experiments are described in Appendix C

Learning from scratch We first study whether KL regularization with the proposed structure and parameterization benefits end-to-end learning. As baselines, we use a policy with entropy



(a) Task transfer summary (b) Analysis on task transfer results.

Figure 4: **Task transfer results.** (a) **Left**: From move box to target to move box to 1 of 3 targets with Ant. Box position and proprioception are given to LL controller. **Right**: From move 2 boxes to 2 targets to congregate boxes with Ball. (b) **Left**: From move box to target task to move box and go to target with Ant, AR-1. **Right**: From move box to target to move heavier box with Ant, AR-1. In all cases, we use shared LL prior during pretraining.

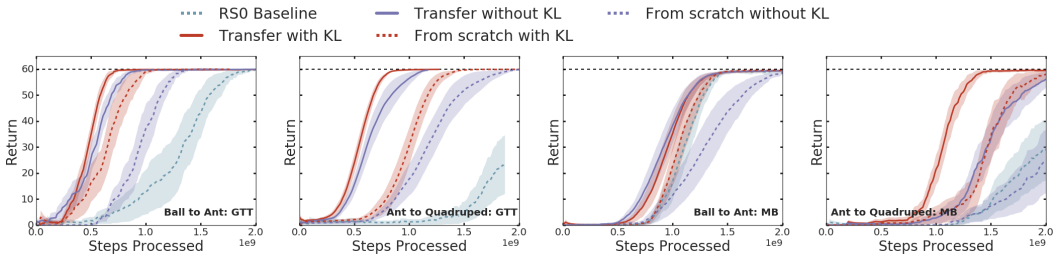


Figure 5: **Body transfer with the AR-1 Prior. Column 1**: Ball to Ant, Go to 1 of 3 targets. **Column 2**: Ball to Ant, Move box to target. **Column 3**: Ant to Quadruped, Go to 1 of 3 targets. **Column 4**: Ant to Quadruped, Move box to target.

regularization (SVG-0) and a KL regularized policy with unstructured default policy similar to Galashov et al. (2019); Teh et al. (2017) (DISTRAL prior). As described in Section 3 we employ a hierarchical structure with shared LL components as a default setting. The HL controller receives full information while the LL controller (and hence the default policy) receives proprioceptive information plus the positions of the box(es) as indicated. The same information asymmetry is applied to the DISTRAL prior i.e. the default policy receives proprioception plus box positions as inputs. We explore multiple HL default policies including Isotropic Gaussian, AR(1) process, and learned AR process. Figure 3 illustrates the results of the experiment. Our main finding is that the KL regularized objective significantly speeds up learning of complex tasks, and that the hierarchical approach provides an advantage over the flat, DISTRAL formulation. The gap increases for more challenging tasks (*e.g.* move 2 boxes to 2 targets).

Task transfer In the experiments we introduce two baselines. The first baseline, the identical model learned from scratch (Hierarchical Agent), allows us to assess the benefit of transfer. The second baseline is transfer with a DISTRAL-style prior, which provides an indication whether the hierarchical policy structure is beneficial for transfer. Additionally, we compare different types of HL default policies. Figure 4a illustrates the result of task transfer. Overall, transferring the pretrained default policy brings significant benefits when learning related tasks. Furthermore the hierarchical architecture which facilitates parameter reuse performs better than the DISTRAL prior regardless of type of HL default policy. While sharing the LL is effective for transfer between tasks with significant overlap, allowing the LL *policy* to diverge from the LL *default policy* as in eq. (4) is useful in some cases. Figure 4b illustrates the result on task transfer scenarios requiring adaptation of skills. Here the LL policy is only initialized and soft-constrained to the behavior of the LL default policy (via the KL term in eq. (4)) which allows adapting the LL skills as required for target task.

Body transfer We explore this body transfer setup in continuous environments. We compare performance to learning the hierarchical policy from scratch and analyze the effects of the KL regularization. The experimental setup in the continuous case is the same as before, and Figure 5

provides results for different types of bodies and tasks. Generally transferring the HL component and relying on both the task reward and the KL term as a dense shaping reward signal for LL controller works best in these settings.

REFERENCES

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Wojciech Czarnecki, Siddhant Jayakumar, Max Jaderberg, Leonard Hasenclever, Yee Whye Teh, Nicolas Heess, Simon Osindero, and Razvan Pascanu. Mix match agent curricula for reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pp. 271–278, 1993.

Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2169–2176. IEEE, 2017.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2017.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 202–211. AUAI Press, 2016.

Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-level discovery of deep options. *CoRR*, abs/1703.08294, 2017. URL `http://arxiv.org/abs/1703.08294`.

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. In *International Conference on Learning Representations*, 2018.

Alexandre Galashov, Siddhant Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojtek M. Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in KL-regularized RL. In *International Conference on Learning Representations*, 2019.

Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, S. M. Ali Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1666–1675, 2018.

Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. In *International Conference on Learning Representations*, 2018.

Anirudh Goyal, Riashat Islam, DJ Strouse, Zafarali Ahmed, Hugo Larochelle, Matthew Botvinick, Sergey Levine, and Yoshua Bengio. Transfer and exploration via the information bottleneck. In *International Conference on Learning Representations*, 2019.

Jordi Grau-Moya, Felix Leibfried, and Peter Vrancx. Soft q-learning with mutual-information regularization. In *International Conference on Learning Representations*, 2019.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. In *International Conference on Learning Representations*, 2017.

Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. In *International Conference on Learning Representations*, 2017.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pp. 1352–1361, 2017.

Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1851–1860, 2018a.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870, 2018b.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, 2015.

Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Nicolas Heess, Dhruva Tirumala, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. DDCO: discovery of deep continuous options forrobot learning from demonstrations. *CoRR*, abs/1710.05421, 2017. URL `http://arxiv.org/abs/1710.05421`.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control. In *International Conference on Learning Representations*, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, 2016.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2775–2785, 2017.

Ofir Nachum, Shixiang (Shane) Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems 31*, pp. 3307–3317. 2018.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2019.

OpenAI. Openai five. `https://blog.openai.com/openai-five/`, 2018.

OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

Pedro A Ortega and Daniel A Braun. Thermodynamics as a theory of decision-making with information-processing costs. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 469(2153):20120683, 2013.

Ronald Parr and Stuart J Russell. Reinforcement learning with hierarchies of machines. In *Advances in neural information processing systems*, pp. 1043–1049, 1998.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *Robotics: science and systems*, volume 13, pp. 3052–3056, 2012.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7008–7024, 2017.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom van de Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4344–4353, 2018.

Jonathan Rubin, Ohad Shamir, and Naftali Tishby. Trading value and information in mdps. *Decision Making with Imperfect Decision Makers*, pp. 57–74, 2012.

Simon Schmitt, Jonathan J Hudson, Augustin Zidek, Simon Osindero, Carl Doersch, Wojciech M Czarnecki, Joel Z Leibo, Heinrich Kuttler, Andrew Zisserman, Karen Simonyan, et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

Pierre Sermanet, Corey Lynch, Jasmine Hsu, and Sergey Levine. Time-contrastive networks: Self-supervised learning from multi-view observation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 14–15, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Susanne Still and Doina Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148, 2012.

Daniel Strouse, Max Kleiman-Weiner, Josh Tenenbaum, Matt Botvinick, and David J Schwab. Learning to share and hide intentions using information regularization. In *Advances in Neural Information Processing Systems*, pp. 10270–10281, 2018.

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

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4496–4506, 2017.

Stas Tiomkin and Naftali Tishby. A unified bellman equation for causal information and value in markov decision processes. *arXiv preprint arXiv:1703.01585*, 2017.

Naftali Tishby and Daniel Polani. Information theory of decisions and actions. *Perception-Action Cycle*, pp. 601–636, 2011.

Emanuel Todorov. Linearly-solvable markov decision problems. In *Advances in Neural Information Processing Systems*, 2007.

Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1049–1056. ACM, 2009.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 3540–3549, 2017.

Saining Xie, Alexandre Galashov, Siqi Liu, Shaobo Hou, Razvan Pascanu, Nicolas Heess, and Yee Whye Teh. Transferring task goals via hierarchical reinforcement learning, 2018. URL `https://openreview.net/forum?id=S1Y6TtJvG`.

Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. *arXiv preprint arXiv:1810.06045*, 2018.

Brian D Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, 2010.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

A ALGORITHM

This section provides more details about the learning algorithm we use to optimize Equation equation 4 in the main text. We employ SVG(0) Heess et al. (2015) with experience replay as a base algorithm and adapt it to support learning hierarchical policy and prior. Unless otherwise mentioned, we follow notations from the main paper.

To optimize the hierarchical policy, we follow a strategy similar to Heess et al. (2016) and reparameterize $z_t \sim \pi^H(z_t|x_t)$ as $z_t = f^H(x_t, \epsilon_t)$, where $\epsilon_t \sim \rho(\epsilon_t)$ is a fixed distribution. The $f^H(\cdot)$ is a deterministic function that outputs distribution parameters. In practice this means that the hierarchical policy can be treated as a flat policy $\pi(a_t|\epsilon_t, x_t) = \pi^L(a_t|f^H(x_t, \epsilon_t), x_t)$. We exploit the reparameterized flat policy to employ existing distributed learning algorithm with minimal modification.

We employ distributed version of SVG(0) Heess et al. (2015) augmented with experience replay and off-policy correction algorithm called Retrace Munos et al. (2016). The SVG(0) reparameterize a policy $p(a|s)$ and optimize it by backpropagating gradient from a learned action value function $Q(a, s)$ through a sampled action a.

To employ this algorithm, we reparameterize action from flat policy $a_t \sim \pi_\theta(a_t|\epsilon_t, x_t)$ with parameter θ as $a_t = h_\theta(\epsilon_t, x_t, \xi_t)$, where $\xi_t \sim \rho(\xi_t)$ is a fixed distribution, and $h_\theta(\epsilon_t, x_t)$ is a deterministic function outputting the parameters of distribution $\pi_\theta(a_t|\epsilon_t, x_t)$. We also introduce the action value function $Q(a_t, z_t, x_t)$. Unlike policies without hierarchy, we estimate the action value depending on the sampled action z_t as well, so that it could capture the future returns depending on z_t. Given the flat policy and the action value function, SVG(0) Heess et al. (2015) suggests to use following gradient estimate

$$\begin{aligned}
&\nabla_\theta \mathbb{E}_{\pi_\theta(a|\epsilon_t, x_t)} Q(a, z_t, x_t) \\
&= \nabla_\theta \mathbb{E}_{\rho(\xi)} Q(h_\theta(\epsilon_t, x_t, \xi), z_t, x_t) \\
&= \mathbb{E}_{\rho(\xi)} \frac{\partial Q}{\partial h} \frac{\partial h}{\partial \theta} \approx \frac{1}{M} \sum_{i=1}^{M} \frac{\partial Q}{\partial h} \frac{\partial h}{\partial \theta}\bigg|_{\xi=\xi_i},
\end{aligned} \tag{6}$$

which facilitates using backpropagation. Note that policy parameter θ could be learned through z_t as well, but we decide not to because it tends to make learning unstable.

To learn action value function $Q(a_t, z_t, x_t)$ and learn policy, we use off-policy trajectories from experience replay. We use Retrace Munos et al. (2016) to estimate the action values from off-policy trajectories. The main idea behind Retrace is to use importance weighting to correct for the difference between the behavior policy μ and the online policy π, while cutting the importance weights to reduce variance. Specifically, we estimate corrected action value with

$$\hat{Q}_t^R = Q_t + \sum_{s \geq t} \gamma^{s-t'} \left(\prod_{i=s}^{t} c_i \right) \delta_s Q, \tag{7}$$

where $\delta_s Q = r_s + \gamma(\hat{V}_{s+1} - \alpha D_{\mathrm{KL}\,s+1}) - Q_s$ and $Q_t = Q(a_t, z_t, x_t)$. $\hat{V}_s = \mathbb{E}_{\pi(a|\epsilon_t, x_t)}[Q(a, z_t, x_t)]$ is estimated bootstrap value, $D_{\mathrm{KL}s} = D_{\mathrm{KL}}\left[\pi^H(z|x_s) \| \pi_0^H(z|x_s)\right]$ and γ is discount. $c_i = \lambda \min\left(\frac{\pi(a_i|\epsilon_i, x_i)}{\mu(a_i|x_i)}\right)$ is truncated importance weight called *traces*.

There are, however, a few notable details that we adapt for our method. Firstly, we do not use the latent z_t sampled from the trajectory. This is possible because the latent does not affect the environment directly. Instead, we consider the behavior policy as $\mu(a|x)$, which does not depend on latents. This approach is useful since we do not need to consider the importance weight with respect to the HL policy, which might introduce additional variance in the estimator. Another detail is that the KL term at step s is not considered in $\delta_s Q$ because the KL at step s is not the result of action a_s. Instead, we introduce close form KL at step s as a loss for the high level policy $\pi^H(z_t|x_t)$ to compensate for this. The pseudocode for the resulting algorithm is illustrated in Algorithm 1.

B Related Work

Entropy regularized reinforcement learning (RL), also known as maximum entropy RL (Ziebart, 2010; Kappen et al., 2012; Toussaint, 2009) is a special case of KL regularized RL. This framework connects probabilistic inference and sequential decision making problems. Recently, this idea has been adapted to deep reinforcement learning (Fox et al., 2016; Schulman et al., 2017; Nachum et al., 2017; Haarnoja et al., 2017; Hausman et al., 2018; Haarnoja et al., 2018b).

Introducing a parameterized default policy provides a convenient way to transfer knowledge or regularize the policy. Schmitt et al. (2018) use a pretrained policy as the default policy; other works jointly learn the policy and default policy to capture reusable behaviour from experience (Teh et al., 2017; Czarnecki et al., 2018; Galashov et al., 2019; Grau-Moya et al., 2019). To retain the role of default policy as a regularizer, it has been explored to restrict its input (Galashov et al., 2019; Grau-Moya et al., 2019), parameteric form (Czarnecki et al., 2018) or to share it across different contexts (Teh et al., 2017; Ghosh et al., 2018).

Another closely related regularization for RL is using information bottleneck (Tishby & Polani, 2011; Still & Precup, 2012; Rubin et al., 2012; Ortega & Braun, 2013; Tiomkin & Tishby, 2017). Galashov et al. (2019) discussed the relation between information bottleneck and KL regularized RL. Strouse et al. (2018) learn to hide or reveal information for future use in multi-agent cooperation or competition. Goyal et al. (2019) consider identifying bottleneck states based on objective similar to eq. equation 5, which is a special case of our framework, and using it for transfer. Their transfer scenario is differently motivated with ours, and different objective based on KL divergence between pretrained HL controllers is used to provide intrinsic reward for learning a new policy without latent variables.

The hierarchical RL literature (Dayan & Hinton, 1993; Parr & Russell, 1998; Sutton et al., 1999) has studied hierarchy extensively as a means to introduce inductive bias. Among various ways (Sutton et al., 1999; Bacon et al., 2017; Vezhnevets et al., 2017; Nachum et al., 2018; 2019; Xie et al., 2018), our approach resembles Heess et al. (2016); Hausman et al. (2018); Haarnoja et al. (2018a); Merel et al. (2019), in that a HL controller modulates a LL controller through a continuous channel. For learning the LL controller, imitation learning (Fox et al., 2017; Krishnan et al., 2017; Merel et al., 2019), unsupervised learning (Gregor et al., 2017; Eysenbach et al., 2019) and meta learning (Frans et al., 2018) have been employed. Similar to our approach, (Heess et al., 2016; Florensa et al., 2017; Hausman et al., 2018) use a pretraining task to learn a reusable LL controller. However, the concept of a default policy has not been widely explored in this context.

Works that transfer knowledge across different bodies include (Devin et al., 2017; Gupta et al., 2017; Sermanet et al., 2017; Xie et al., 2018). Devin et al. (2017) mixes and matches modular task and

Flat policy: $\pi_\theta(a_t|\epsilon_t, x_t)$ with parameter θ
HL policy: $\pi^H_\theta(z_t|x_t)$, where latent is sampled by reparameterization $z_t = f^H_\theta(x_t, \epsilon_t)$
Default policies: $\pi^H_{0,\phi}(z_t|x_t)$ and $\pi^L_{0,\phi}(a_t|z_t, x_t)$ with parameter ϕ
Q-function: $Q_\psi(a_t, z_t, x_t)$ with parameter ψ
Initialize target parameters $\theta' \leftarrow \theta, \quad \phi' \leftarrow \phi, \quad \psi' \leftarrow \psi$.
Target update counter: $c \leftarrow 0$
Target update period: P
Replay buffer: $\mathcal{B}$
repeat
  for $t = 0, K, 2K, ...T$ **do**
    Sample partial trajectory $\tau_{t:t+K}$ with action log likelihood $l_{t:t+K}$ from replay buffer $\mathcal{B}$:
    $\tau_{t:t+K} = (s_t, a_t, r_t, ..., r_{t+K}), \quad l_{t:t+K} = (l_t, ..., l_{t+K}) =$
    $(\log \mu(a_t|x_t), ..., \log \mu(a_{t+K}|x_{t+K}))$
    Sample latent: $\epsilon_{t'} \sim \rho(\epsilon)$, $z_{t'} = f^H_\theta(x_{t'}, \epsilon_{t'})$
    Compute KL:
    $\widehat{D_{\mathrm{KL}t'}} = D_{\mathrm{KL}}\left[\pi^H_\theta(z|x_{t'}) \| \pi^H_{0,\phi'}(z|x_{t'})\right] + D_{\mathrm{KL}}\left[\pi^L_\theta(a|z_{t'}, x_{t'}) \| \pi^L_{0,\phi'}(a|z_{t'}, x_{t'})\right]$
    Compute KL for Distillation:$\widehat{D_{\mathrm{KL}t'}}^{\mathcal{D}} =$
    $D_{\mathrm{KL}}\left[\pi^H_\theta(z|x_{t'}) \| \pi^H_{0,\phi}(z|x_{t'})\right] + D_{\mathrm{KL}}\left[\pi^L_\theta(a|z_{t'}, x_{t'}) \| \pi^L_{0,\phi}(a|z_{t'}, x_{t'})\right]$
    Compute action entropy: $\widehat{\mathrm{H}}_{t'} = \mathbb{E}_{\pi_\theta(a|\epsilon_{t'}, x_{t'})}[\log \pi_\theta(a|\epsilon_{t'}, x_{t'})]$
    Estimate bootstrap value: $\hat{V}_{t'} = \mathbb{E}_{\pi_\theta(a|\epsilon_{t'}, x_{t'})}\left[Q_{\psi'}(a, z_{t+K}, x_{t+K})\right] - \alpha\widehat{D_{\mathrm{KL}}}_{t+K}$
    Estimate traces Munos et al. (2016): $\hat{c}_{t'} = \lambda \min\left(\frac{\pi_\theta(a_{t'}|\epsilon_{t'}, x_{t'})}{l_{t'}}\right)$
    Apply Retrace to estimate Q targets Munos et al. (2016):
    $\hat{Q}^R_{t'} = Q_{\psi'}(a_{t'}, z_{t'}, x_{t'}) +$
    $\sum_{s \geq t'} \gamma^{s-t'} \left(\prod_{i=s}^{t'} \hat{c}_i\right)\left(r_s + \gamma\left(\hat{V}_{s+1} - \alpha\widehat{D_{\mathrm{KL}}}_{s+1}\right) - Q_{\psi'}(a_s, z_s, x_s)\right)$
    Policy loss: $\hat{L}_\pi = \sum_{i=t}^{t+K-1} \mathbb{E}_{\pi_\theta(a|\epsilon_i, x_i)} Q_{\psi'}(a, z_i, x_i) - \alpha\widehat{D_{\mathrm{KL}}}_i + \alpha_{\mathrm{H}}\widehat{\mathrm{H}}_i$
    Q-value loss: $\hat{L}_Q = \sum_{i=t}^{t+K-1} \|\hat{Q}^R_i - Q_\psi(a, z_i, x_i)\|^2$
    Default policy loss: $\hat{L}_{\pi^H_0} = \sum_{i=t}^{t+K-1} \widehat{KL_i}^{\mathcal{D}}$
    $\theta \leftarrow \theta + \beta_\pi \nabla_\theta \hat{L}_\pi \quad \phi \leftarrow \phi + \beta_{\pi^H_0} \nabla_\phi \hat{L}_{\pi^H_0}$
    $\psi \leftarrow \psi - \beta_Q \nabla_\psi \hat{L}_Q$
    Increment counter $c \leftarrow c + 1$
    if $c > P$ **then**
      Update target parameters $\theta' \leftarrow \theta, \quad \phi' \leftarrow \phi, \quad \psi' \leftarrow \psi$
      $c \leftarrow 0$
    end if
  end for
until

Algorithm 1: SVG(0) Heess et al. (2015) with experience replay for hierarchical policy

body policies for zero-shot generalization to unseen combination. Gupta et al. (2017); Sermanet et al. (2017) learn a common representation space to align poses from different bodies. Xie et al. (2018) transfer the HL controller in a hierarchical agent, where the LL controller is learned with an intrinsic reward based on goals in state space. This approach, however, requires careful design of the goal space.

C Experimental Settings

C.1 Tasks and Bodies

We consider task distributions that are designed such that their solutions exhibit significant overlap in trajectory space so that transfer can reasonably be expected. They are further designed to contain instances of variable difficulty and hence provide a natural curriculum.

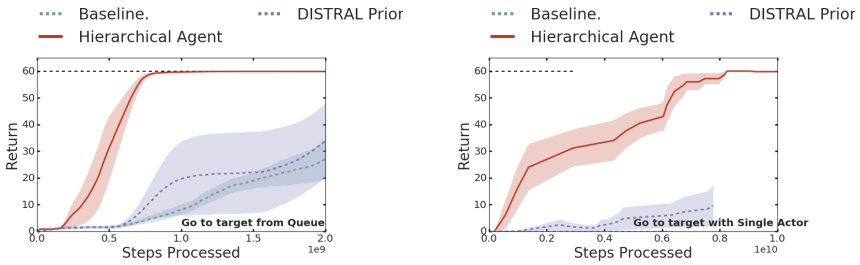


Figure 6: **Results with alternative training regimes.** in Go to 1 of 3 targets, Ant. AR-1 process. **Left:** Learning curves with quasi on-policy training regime. **Right:** Learning curves with a single actor. Go to 1 of 3 targets, Ant. AR-1 process.

Go to 1 of K targets: In this task the agent receives a sparse reward on reaching a specific target among K locations. The egocentric locations of each of the targets and the goal target index are provided as observations. **Move K boxes to K targets**: the goal is to move one of K boxes to one of K targets (the positions of which are randomized) as indicated by the environment. **Move heavier box**: variants of move K boxes to K targets with heavier boxes. **Gather boxes**: the agent needs to move two boxes such that they are in contact with each other. We also consider combinations of these tasks, such as **Move box or go to target**, where in each episode the agent is required solve either the Go to targets task or the Move box to targets task, and a combined task, **Move box and go to target**, in which the agent is required to move the box to one target and then go to a different target in a single episode. Here two small sparse rewards are provided for each component of the task that is completed and an extra bonus reward is awarded for solving both tasks.

We use three different bodies: Ball, Ant, and Quadruped. Ball and Ant have been used in several previous works (Heess et al., 2017; Xie et al., 2018; Galashov et al., 2019), and we introduce the Quadruped as an alternative to the Ant. The **Ball** is a body with 2 actuators for moving forward or backward, turning left, and turning right. The **Ant** is a body with 4 legs and 8 actuators, which moves its legs to walk and to interact with objects. The **Quadruped** is similar to the Ant, but with 12 actuators. Each body is characterized by a different set of proprioceptive (proprio) features.

C.2 Details of Experimental Setup

Throughout the experiments, we use 32 actors to collect trajectories and a single learner to optimize the model. We plot average episode return with respect to the number of steps processed by the learner. Note that the number of steps is different from the number of agent's interaction with environment, because the collected trajectories are processed multiple times by a centralized learner to update model parameters. When learning from scratch we report results as the mean and standard deviation of average returns for 5 random seeds. For the transfer learning experiments, we use 5 seed for the initial training, and then transfer all pretrained models to a new task, and train two new HL or LL controllers (two random seeds) per model on the transfer task. Thus, in total, 10 different runs are used to estimate mean and standard deviations of the average returns. Hyperparameters, including KL cost and action entropy regularization cost, are optimized on a per-task basis.

D Additional Experimental Results

In the main paper, we present results based on learning speed with respect to the number of time steps processed by learner in distributed learning setup. Note that the number of time steps processed by the learner does not necessarily correspond to the number of collected trajectory time steps because of the use of experience replay, which allows to learning to proceed regardless of the amount of collected trajectories. We also experimented with two alternative training regimes to ensure that the speedup results reported are consistent. In Figure 6 left, we compare the learning curves for our method against the SVG-0 and DISTRAL baselines in a quasi on-policy training regime similar to that of Espeholt et al. (2018). In Figure 6 right, we perform a similar comparison in the original replay based off policy training regime but with a single actor generating the data. In both cases, our method learns faster than both baselines.