Dynamics-Aware Unsupervised Discovery of Skills

Archit Sharma[*], **Shixiang Gu**, **Sergey Levine**, **Vikash Kumar**, **Karol Hausman**
Google Brain
{architsh, shanegu, slevine, vikashplus, karolhausman}@google.com

Abstract

Conventionally, model-based reinforcement learning (MBRL) aims to learn a global model for the dynamics of the environment. A good model can potentially enable planning algorithms to generate a large variety of behaviors and solve diverse tasks. However, learning an accurate model for complex dynamical systems is difficult, and even then, the model might not generalize well outside the distribution of states on which it was trained. In this work, we combine model-based learning with model-free learning of primitives that make model-based planning easy. To that end, we aim to answer the question: how can we discover skills whose outcomes are easy to predict? We propose an unsupervised learning algorithm, Dynamics-Aware Discovery of Skills (DADS), which simultaneously discovers *predictable* behaviors and learns their dynamics. Our method can leverage continuous skill spaces, theoretically, allowing us to learn infinitely many behaviors even for high-dimensional state-spaces. We demonstrate that *zero-shot planning* in the learned latent space significantly outperforms standard MBRL and model-free goal-conditioned RL, can handle sparse-reward tasks, and substantially improves over prior hierarchical RL methods for unsupervised skill discovery. Video demonstration of our results are available at: `https://sites.google.com/view/dads-skill`

Figure 1: A humanoid agent discovers diverse locomotion primitives *without any reward* using DADS. We show zero-shot generalization to downstream tasks by composing the learned primitives using model predictive control, enabling the agent to follow an online sequence of goals (green markers) without any additional training.

1 Introduction

Deep reinforcement learning (RL) enables autonomous learning of diverse and complex tasks with rich sensory inputs, temporally extended goals, and challenging dynamics, such as discrete game-playing domains [46, 62], and continuous control domains including locomotion [59, 31] and manipulation [56, 34, 25]. Most of the deep RL approaches learn a Q-function or a policy that are directly optimized for the training task, which limits their generalization to new scenarios. In contrast, MBRL methods [45, 15, 72] can acquire dynamics models that may be utilized to perform unseen tasks at test time. While this capability has been demonstrated in some of the recent works

[*]Work done as a member of Google AI Residency Program (g.co/airesidency).

Preprint. Under review.

[44, 49, 12, 41, 26], learning an accurate global model that works for all state-action pairs can be exceedingly challenging, especially for high-dimensional system with complex and discontinuous dynamics. The problem is further exacerbated as the learned global model has limited generalization outside of the state distribution it was trained on and exploring the whole state space is generally infeasible. Can we retain the flexibility of model-based RL, while using model-free RL to acquire proficient low-level behaviors under complex dynamics?

While learning a global dynamics model that captures all the different behaviors for the entire state-space can be extremely challenging, learning a model for a specific behavior that acts only in a small part of the state-space can be much easier. For example, consider learning a model for dynamics of all gaits of a quadruped versus a model which only works for a specific gait. If we can learn many such behaviors and their corresponding dynamics, we can leverage model-predictive control to plan in the *behavior space*, as opposed to planning in the action space. The question then becomes: how do we acquire such behaviors, considering that behaviors could be random and unpredictable? To this end, we propose *Dynamics-Aware Discovery of Skills* (DADS), an unsupervised RL framework for learning low-level skills using model-free RL with the explicit aim of making model-based control easy. Skills obtained using DADS are directly optimized for *predictability*, providing a better representation on top of which predictive models can be learned. Crucially, the skills do not require any supervision to learn, and are acquired entirely through autonomous exploration. This means that the repertoire of skills and their predictive model are learned before the agent has been tasked with any goal or reward function. When a task is provided at test-time, the agent utilizes the previously learned skills and model to immediately perform the task without any further training.

The key contribution of our work is an unsupervised reinforcement learning algorithm, DADS, grounded in mutual-information-based exploration. We demonstrate that our objective can embed learned primitives in continuous spaces, which allows us to learn a large, diverse set of skills. Crucially, our algorithm also learns to model the dynamics of the skills, which enables the use of model-based planning algorithms for downstream tasks. We adapt the conventional model predictive control algorithms to plan in the space of primitives, and demonstrate that we can compose the learned primitives to solve downstream tasks without any additional training.

2 Preliminaries

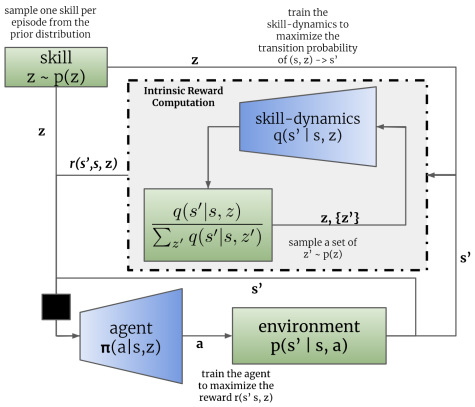
Mutual information has been used as an objective to encourage exploration in reinforcement learning [32, 47]. According to its definition, $\mathcal{I}(A; B) = \mathcal{H}(A) - \mathcal{H}(A \mid B)$, optimizing mutual information $\mathcal{I}$ with respect to B amounts to maximizing the entropy $\mathcal{H}$ of A while minimizing the conditional entropy $\mathcal{H}(A \mid B)$. If A is a function of the state and B represents actions, this objective encourages the state entropy to be high, causing the underlying policy to be exploratory. Recently, multiple works [17, 24, 2] apply this idea to learn diverse skills which maximally cover the state space.

To leverage planning-based control, MBRL estimates the true dynamics of the environment by learning a model $\hat{p}(s' \mid s, a)$. This allows it to predict a trajectory of states $\hat{\tau}_H = (s_t, \hat{s}_{t+1}, \dots \hat{s}_{t+H})$ resulting from a sequence of actions without any additional interaction with the environment. A similar simulation of the trajectory $\hat{\tau}_H$ can be carried out using a model parameterized as $q(s' \mid s, z)$, where z denotes the skill that is being executed. This modification to MBRL not only mandates the existence of a policy $\pi(\cdot \mid s, z)$ executing the actual actions in environment, but more importantly, the policy to execute these actions in a way that maintains *predictability under* q. In this setup, skills z are effectively an abstraction for the actions $a_1, a_2 \dots$ that are executed in the environment. This scheme forgoes a much harder task of learning a global model $\hat{p}$, in exchange of a collection of potentially simpler models of behavior-specific dynamics. In addition, the planning problem becomes easier as the planner is searching over a skill space z that can act on longer horizons than granular actions a.

These seemingly unrelated ideas can be combined into a single optimization scheme, where we first discover skills (and their models) without any extrinsic reward and then compose these skills to optimize for the task defined at test time using model-based planning. At train time, we assume a Markov Decision Process (MDP) $\mathcal{M}_1 \equiv (\mathcal{S}, \mathcal{A}, p)$. The state space $\mathcal{S}$ and action space $\mathcal{A}$ are assumed to be continuous, and the $\mathcal{A}$ bounded. We assume the transition dynamics p to be stochastic, such that $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, \infty)$. We learn a skill-conditioned policy $\pi(a \mid s, z)$, where the skills z belongs to the space $\mathcal{Z}$, detailed in Section 3. We assume that the skills are sampled from a prior $p(z)$ over $\mathcal{Z}$. We simultaneously learn a skill-conditioned transition function $q(s' \mid s, z)$, coined as

skill-dynamics, which predicts the transition to the next state s' from the current state s for the skill z under the given dynamics p. At test time, we assume an MDP $\mathcal{M}_2 \equiv (\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}, \mathcal{A}, p$ match those defined in $\mathcal{M}_1$, and the reward function $r : \mathcal{S} \times \mathcal{A} \mapsto (-\infty, \infty)$. We plan in $\mathcal{Z}$ using $q(s' \mid s, z)$ to compose the learned skills z for optimizing r in $\mathcal{M}_2$, which we detail in Section 4.

3 Dynamics-Aware Discovery of Skills (DADS)



Algorithm 1: Dynamics-Aware Discovery of Skills (DADS)

Initialize π, q_ϕ;
while *not converged* **do**
  Sample a skill $z \sim p(z)$ every episode;
  Collect new M on-policy samples;
  Update q_ϕ using K_1 steps of gradient descent on M transitions;
  Compute $r_z(s, a, s')$ for M transitions;
  Update π using any RL algorithm;
end

Figure 2: The agent π interacts with the environment to produce a transition $s \to s'$. Intrinsic reward is computed by computing the transition probability under q for the current skill z, compared to random samples from the prior $p(z)$. The agent maximizes the intrinsic reward computed for a batch of episodes, while q maximizes the log-probability of the actual transitions of $(s, z) \to s'$.

We now establish a connection between mutual-information-based exploration and model-based RL by deriving an intrinsic reward that reflects predictability under skill-dynamics. For an episodic setting with horizon T, we aim to maximize:

$$\mathcal{I}(s_1, \dots s_T; z) \quad s.t. \quad \sum_{t=1}^{T-1} \mathcal{I}(a_t; \{s_t, z\}) \leq I_c, \tag{1}$$

for an arbitrary constant upper bound I_c. The proposed objective encodes the intuition that every skill should be maximally informative about the resulting sequence of states $s_1, \dots s_T$ in the MDP $\mathcal{M}_1$, while being minimally informative about the sequence of actions used. For clarity of discussion, we defer a more rigorous justification for this information-bottleneck-style [68, 5] objective to the Appendix B. We simplify Eq. 1:

$$\mathcal{I}(s_1, \dots s_T; z) = \mathcal{I}(s_1; z) + \mathcal{I}(s_2; z \mid s_1) \dots \mathcal{I}(s_T; z \mid s_{T-1}, \dots s_1) \tag{2}$$

$$= \mathcal{I}(s_1; z) + \mathcal{I}(s_2; z \mid s_1) + \dots \mathcal{I}(s_T; z \mid s_{T-1}) \tag{3}$$

by using the chain rule of mutual information to obtain Eq. 2, and the Markovian assumption of the $\mathcal{M}_1$ to obtain Eq. 3. Returning to Eq. 1, we obtain our objective $R(\pi)$ as:

$$R(\pi) = \sum_{t=1}^{T-1} \mathcal{I}(s_{t+1}; z \mid s_t) - \beta \mathcal{I}(a_t; \{s_t, z\}) \tag{4}$$

where we formulate the dual objective using the Lagrangian multiplier β (and ignore the constant βI_c). Using the definition of mutual information, the resulting objective is given by:

$$R(\pi) = \sum_{t=1}^{T-1} \mathbb{E}_{\rho_z(s_t, a_t), z} \Big[\log \frac{p(s_{t+1} \mid s_t, z)}{p(s_{t+1} | s_t)} - \beta \log \frac{\pi(a_t \mid s_t, z)}{\pi(a_t)} \Big] \tag{5}$$

$$\geq \sum_{t=1}^{T} \mathbb{E}_{\rho_z(s_t, a_t), z} \Big[\log \frac{q_\phi(s_{t+1} \mid s_t, z)}{p(s_{t+1} \mid s_t)} - \beta \log \frac{\pi(a_t \mid s_t, z)}{p(a)} \Big] \tag{6}$$

$$R(\pi, q_\phi) \equiv \sum_{t=1}^{T} \mathbb{E}_{\rho_z(s_t, a_t), z} \big[\log q_\phi(s_{t+1} \mid s_t, z) - \log p(s_{t+1} \mid s_t) - \beta \log \pi(a_t \mid s_t, z) \big] \tag{7}$$

where $\rho_z(s_t, a_t)$ represents the stationary state-action distribution under the skill z. For Eq. 6, we use the non-negativity of KL-divergence, that is $\int \pi(a_t) \log \frac{\pi(a_t)}{p(a_t)} da_t \geq 0$, to replace the marginal over the policy $\pi(a_t)$ with the uniform prior $p(a)$ over the bounded action space $\mathcal{A}$. Similarly, we use $\int p(s_{t+1}|s_t, z) \log \frac{p(s_{t+1}|s_t,z)}{q_\phi(s_{t+1}|s_t,z)} ds_{t+1} \geq 0$ to introduce *skill-dynamics* as a parametric variational approximation $q_\phi(s'|s, z)$. Ignoring the constant $\beta \log p(a)$, we get our objective $R(\pi, q_\phi)$ in Eq. 7.

Maximizing $R(\pi, q_\phi)$ immediately suggests an alternating optimization scheme that is summarized in Figure 2. Note that the gradient for ϕ can be expressed as:

$$\nabla_\phi R(\pi, q_\phi) = \sum_{t=1}^{T-1} \mathbb{E}_{\rho_z(s_t,a_t),z}[\nabla_\phi \log q_\phi(s_{t+1}|s_t, z))], \tag{8}$$

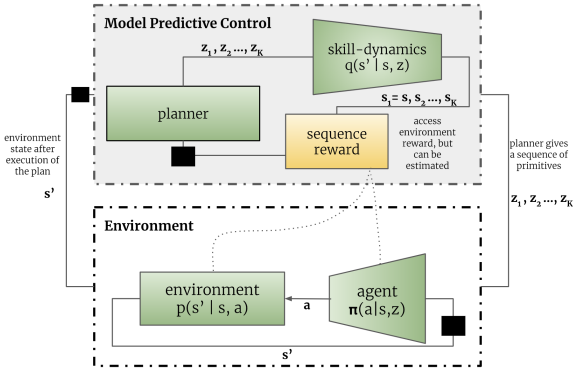
which is simply maximizing the likelihood of the transitions generated by the current policy.

The optimization of the policy π can be interpreted as entropy-regularized RL with a reward function $\log q_\phi(s_{t+1} \mid s_t, z) - \log p(s_{t+1} \mid s_t)$. Unfortunately, $\log p(s_{t+1} \mid s_t)$ is intractable to compute so we need to resort to approximations. We choose to re-use the skill dynamics model to approximate $p(s_{t+1} \mid s_t) = \int p(s_{t+1} \mid s_t, z)p(z)dz \approx \frac{1}{L}\sum_{i=1}^{L} p(s_{t+1} \mid s_t, z_i) \approx \frac{1}{L}\sum_{i=1}^{L} q_\phi(s_{t+1} \mid s_t, z_i)$, where z_i is sampled from the prior $p(z)$. The final reward function can be written as:

$$r_z(s_t, a_t, s_{t+1}) = \log \frac{q_\phi(s_{t+1} \mid s_t, z)}{\sum_{i=1}^{L} q_\phi(s_{t+1} \mid s_t, z_i)} + \log L, \quad z_i \sim p(z). \tag{9}$$

In practice, we often re-use the sample z in the denominator in Eq. 9 amongst the samples $\{z_i\}_{i=1}^{L}$, to obtain a softmax-like construction, providing a smoother optimization landscape. For the actual algorithm, we collect a large on-policy batch of data in every iteration, so that it contains experience collected from different skills. In order to take multiple gradient steps on the same batch of data, we use soft actor-critic [27, 28] as the optimization algorithm for the policy π (although our method is agnostic to the choice of the RL algorithm used to update the policy). The exact implementation details are discussed in the Appendix A.

4 Planning using Skill Dynamics



Algorithm 2: Latent Space Planner

$s \leftarrow s_0$;
Initialize parameters $\mu_1, \ldots \mu_{H_P}$;
for $i \leftarrow 1$ **to** H_E/H_Z **do**
  for $j \leftarrow 1$ **to** R **do**
    $\{z_i, \ldots z_{i+H_P-1}\}_{k=1}^{K} \sim \mathcal{N}_i, \ldots \mathcal{N}_{i+H_P-1}$;
    Compute r_{env} for $\{z_i, \ldots z_{i+H_P-1}\}_{k=1}^{K}$;
    Update $\mu_i, \ldots, \mu_{i+H_P-1}$;
  end
  Sample z_i from $\mathcal{N}_i$;
  Execute $\pi(a|s, z_i)$ for H_Z steps;
end

Figure 3: At test time, the planner executes simulates the transitions in environment using skill-dynamics q, and updates the distribution of plans according to the computed reward on the simulated trajectories. After a few updates to the plan, the first primitive is executed in the environment using the learned agent π.

Given the learned skills $\pi(a \mid s, z)$ and their respective skill-transition dynamics $q_\phi(s' \mid s, z)$, we can perform model-based planning in the latent space $\mathcal{Z}$ to optimize for a reward r that is given to an agent at test time. Note, that this essentially allows us to perform zero-shot planning given the unsupervised pre-training procedure described in Section 3.

In order to perform planning, we employ the model-predictive-control (MPC) paradigm [23], which in a standard setting generates a set of action plans $P_k = (a_{k,1}, \ldots a_{k,H}) \sim P$ for a planning horizon H.

The MPC plans can be generated due to the fact that the planner is able to simulate the trajectory $\hat{\tau}_k = (s_{k,1}, a_{k,1} \dots s_{k,H+1})$ assuming access to the transition dynamics $\hat{p}(s' \mid s, a)$. In addition, each plan computes the reward $r(\hat{\tau}_k)$ for its trajectory according to the reward function r that is provided for the test-time task. Following the MPC principle, the planner selects the best plan according to the reward function r and executes its first action a_1. The planning algorithm repeats this procedure for the next state iteratively until it achieves its goal.

We use a similar strategy to design an MPC planner to exploit previously-learned skill-transition dynamics $q_\phi(s' \mid s, z)$. Note that unlike conventional model-based RL, we generate a plan $P_k = (z_{k,1}, \dots z_{k,H_P})$ in the latent space $\mathcal{Z}$ as opposed to the action space $\mathcal{A}$ that would be used by a standard planner. Since the primitives are temporally meaningful, it is beneficial to hold a primitive for a horizon $H_Z > 1$, unlike actions which are usually held for a single step. Thus, effectively, the planning horizon for our latent space planner is $H = H_P \times H_Z$, enabling longer-horizon planning using fewer primtiives. Similar to the standard MPC setting, the latent space planner simulates the trajectory $\hat{\tau}_k = (s_{k,1}, z_{k,1}, a_{k,1}, s_{k,2}, z_{k,2}, a_{k,2}, \dots s_{k,H+1})$ and computes the reward $r(\hat{\tau}_k)$. After a small number of trajectory samples, the planner selects the first latent action z_1 of the best plan, executes it for H_Z steps in the environment, and the repeats the process until goal completion.

The latent planner P maintains a distribution of latent plans, each of length H_P. Each element in the sequence represents the distribution of the primitive to be executed at that time step. For continuous spaces, each element of the sequence can be modelled using a normal distribution, $\mathcal{N}(\mu_1, \Sigma), \dots \mathcal{N}(\mu_{H_P}, \Sigma)$. We refine the planning distributions for R steps, using K samples of latent plans P_k, and compute the r_k for the simulated trajectory $\hat{\tau}_k$. The update for the parameters follows that in Model Predictive Path Integral (MPPI) controller [73]:

$$\mu_i = \sum_{k=1}^{K} \frac{\exp(\gamma r_k)}{\sum_{p=1}^{K} \exp(\gamma r_p)} z_{k,i} \quad \forall i = 1, \dots H_P \tag{10}$$

While we keep the covariance matrix of the distributions fixed, it is possible to update that as well as shown in [73]. We show an overview of the planning algorithm in Figure 3, and provide more implementation details in Appendix A.

5 Related Work

Central to our method is the concept of skill discovery via mutual information maximization. This principle, proposed in prior work that utilized purely model-free unsupervised RL methods [14, 18, 17, 24, 71], aims to learn diverse skills via a discriminability objective: a good set of skills is one where it is easy to distinguish the skills from each other, which means they perform distinct tasks and cover the space of possible behaviors. Building on this prior work, we distinguish our skills based on how they modify the original uncontrolled dynamics of the system. This simultaneously encourages the skills to be both *diverse* and *predictable*. We also demonstrate that constraining the skills to be predictable makes them more amenable for hierarchical composition and thus, more useful on downstream tasks.

Another line of work that is conceptually close to our method copes with intrinsic motivation [51, 52, 58] which is used to drive the agent's exploration. Examples of such works include empowerment [37, 47], count-based exploration [8, 50, 67, 21], information gain about agent's dynamics [64] and forward-inverse dynamics models [53]. While our method uses an information-theoretic objective that is similar to these approaches, it is used to learn a variety of skills that can be directly used for model-based planning, which is in contrast to learning a better exploration policy for a single skill. We provide a discussion on the connection between empowerment and DADS in Appendix C.

The skills discovered using our approach can also provide extended actions and temporal abstraction, which enable more efficient exploration for the agent to solve various tasks, reminiscent of hierarchical RL (HRL) approaches. This ranges from the classic option-critic architecture [66, 65, 55] to some of the more recent work [7, 70, 48, 29]. However, in contrast to end-to-end HRL approaches [30, 54], we can leverage a stable, two-phase learning setup. The primitives learned through our method provide action and temporal abstraction, while planning with skill-dynamics enables hierarchical composition of these primitives, bypassing many problems of end-to-end HRL.

In the second phase of our approach, we use the learned skill-transition dynamics models to perform model-based planning - an idea that has been explored numerous times in the literature. Model-

based reinforcement learning has been traditionally approached with methods that are well-suited for low-data regimes such as Gaussian Processes [57] showing significant data-efficiency gains over model-free approaches [16, 35, 39, 38]. More recently, due to the challenges of applying these methods to high-dimensional state spaces, MBRL approaches employs Bayesian deep neural networks [49, 12, 22, 20, 42] to learn dynamics models. In our approach, we take advantage of the deep dynamics models that are conditioned on the skill being executed, simplifying the modelling problem. In addition, the skills themselves are being learned with the objective of being predictable, further assists with the learning of the dynamics model. There also have been multiple approaches addressing the planning component of MBRL including linear controllers for local models [44, 40, 10], uncertainty-aware [12, 22] or deterministic planners [49] and stochastic optimization methods [73]. The main contribution of our work lies in discovering model-based skill primitives that can be further combined by a standard model-based planner, therefore we take advantage of an existing planning approach - Model Predictive Path Integral [73] that can leverage our pre-trained setting.

6 Experiments

Through our experiments, we aim to demonstrate that: (a) DADS as a general purpose skill discovery algorithm can scale to high-dimensional problems; (b) discovered skills are amenable to hierarchical composition and; (c) not only is planning in the learned latent space feasible, but it is competitive to strong baselines. In Section 6.1, we provide visualizations and qualitative analysis of the skills learned using DADS. We demonstrate in Section 6.2 and Section 6.4 that optimizing the primitives for predictability renders skills more amenable to temporal composition that can be used for Hierarchical RL.We benchmark against state-of-the-art model-based RL baseline in Section 6.3, and against goal-conditioned RL in Section 6.5.

6.1 Qualitative Analysis

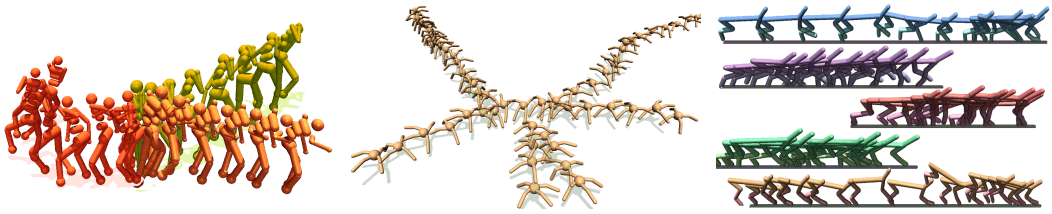


Figure 4: Skills learned on different MuJoCo environments in the OpenAI gym. DADS can discover diverse skills without any extrinsic rewards, even for problems with high-dimensional state and action spaces.

In this section, we provide a qualitative discussion of the unsupervised skills learned using DADS. We use the MuJoCo environments [69] from the OpenAI gym as our test-bed [9]. We find that our proposed algorithm can learn diverse skills without any reward, even in problems with high-dimensional state and actuation, as illustrated in Figure 4. DADS can discover primitives for Half-Cheetah to run forward and backward with multiple different gaits, for Ant to navigate the environment using diverse locomotion primitives and for Humanoid to walk using stable locomotion primitives with diverse gaits and direction. The videos of the discovered primitives are available at: `https://sites.google.com/view/dads-skill`

Qualitatively, we find the skills discovered by DADS to be predictable and stable, in line with implicit constraints of the proposed objective. While the Half-Cheetah will learn to run in both backward and forward directions, DADS will disincentivize skills which make Half-Cheetah flip owing to the reduced predictability on landing. Similarly, skills discovered for Ant rarely flip over, and tend to provide stable navigation primitives in the environment. This also incentivizes the Humanoid, which is characteristically prone to collapsing and extremely unstable by design, to discover gaits which are stable for sustainable locomotion.

One of the significant advantages of the proposed objective is that it is compatible with continuous skill spaces, which has not been shown in prior work on skill discovery [17]. Not only does this allow us to embed a large and diverse set of skills into a compact latent space, but also the smoothness of the learned space allows us to interpolate between behaviors generated in the environment. We

demonstrate this on the Ant environment (Figure 5), where we learn two-dimensional continuous skill space with a uniform prior over $(-1, 1)$ in each dimension, and compare it to a discrete skill space with a uniform prior over 20 skills. Similar to [17], we restrict the observation space of the skill-dynamics q to the cartesian coordinates (x, y). We hereby call this the *x-y prior*, and discuss its role in Section 6.2.

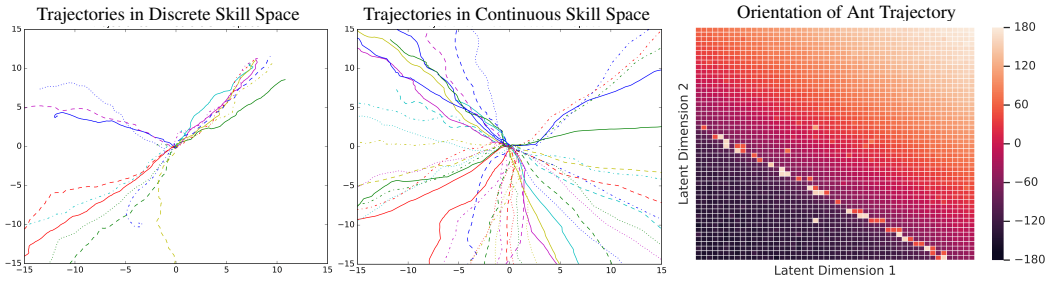


Figure 5: (Left, Centre) X-Y traces of Ant skills and (Right) Heatmap to visualize the learned continuous skill space. Traces demonstrate that the continuous space offers far greater diversity of skills, while the heatmap demonstrates that the learned space is smooth, as the orientation of the X-Y trace varies smoothly as a function of the skill.

In Figure 5, we project the trajectories of the learned Ant skills from both discrete and continuous spaces onto the Cartesian plane. From the traces of the skills, it is clear that the continuous latent space can generate more diverse trajectories. We demonstrate in Section 6.3, that continuous primitives are more amenable to hierarchical composition and generally perform better on downstream tasks. More importantly, we observe that the learned skill space is semantically meaningful. The heatmap in Figure 5 shows the orientation of the trajectory (with respect to the x-axis) as a function of the skill $z \in \mathcal{Z}$, which varies smoothly as z is varied, with explicit interpolations shown in Appendix D.

6.2 Skill Variance Analysis

In an unsupervised skill learning setup, it is important to optimize the primitives to be diverse. However, we argue that diversity is not sufficient for the learned primitives to be useful for downstream tasks. Primitives must exhibit low-variance behavior, which enables long-horizon composition of the learned skills in a hierarchical setup. We analyze the variance of the x-y trajectories in the environment, where we also benchmark the variance of the primitives learned by DIAYN [17]. For DIAYN, we use the x-y prior for the skill-discriminator, which biases the discovered skills to diversify in the x-y space. This step was necessary for that baseline to obtain a competitive set of navigation skills. Figure 6 (Left) demonstrates that DADS, which optimizes the primitives for predictability *and* diversity, yields *significantly* lower-variance primitives when compared to DIAYN, which only optimizes for diversity. This is starkly demonstrated in the plots of X-Y traces of skills learned in different setups. Skills learned by DADS show significant control over the trajectories generated in the environment, while skills from DIAYN exhibit high variance in the environment, which limits their utility for hierarchical control. This is further demonstrated quantitatively in Section 6.4.

While optimizing for predictability already significantly reduces the variance of the trajectories generated by a primitive, we find that using the x-y prior with DADS brings down the skill variance even further. For quantitative benchmarks in the next sections, we assume that the Ant skills are learned using an x-y prior on the observation space, for both DADS and DIAYN.

6.3 Model-Based Reinforcement Learning

The key utility of learning a parametric model $q_\phi(s'|s, z)$ is to be enable use of planning algorithms for downstream tasks, which can be extremely sample-efficient. In our setup, we can solve test-time tasks in zero-shot, that is *without any learning on the downstream task*. We compare with the state-of-the-art model-based RL method [11], which learns a dynamics model parameterized as $p(s'|s, a)$, on the task of the Ant navigating to a specified goal with a dense reward. Given a goal g, reward at any position u is given by $r(u) = -\|g - u\|_2$. We benchmark our method against the following variants:

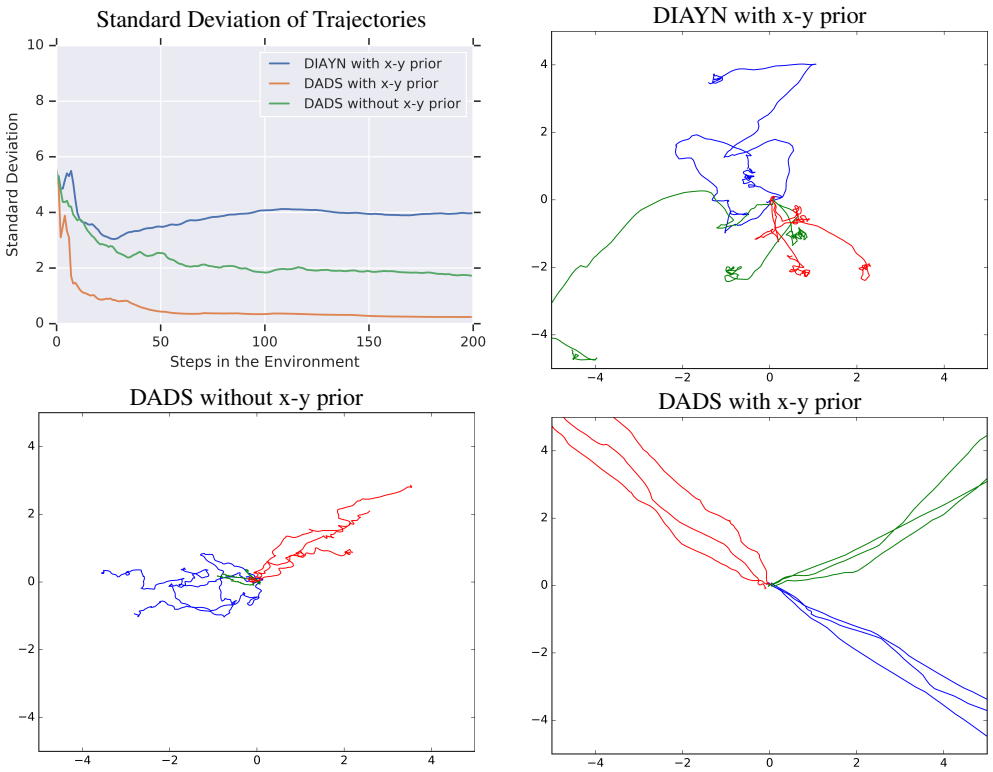


Figure 6: (Top-Left) Standard deviation of Ant's position as a function of steps in the environment, averaged over multiple skills and normalized by the norm of the position. (Top-Right to Bottom-Left Clockwise) X-Y traces of skills learned using DIAYN with x-y prior, DADS with x-y prior and DADS without x-y prior, where the same color represents trajectories resulting from the execution of the same skill z in the environment. High variance skills from DIAYN offer limited utility for hierarchical control.

- Random-MBRL (*rMBRL*): We train the model $p(s'|s,a)$ on randomly collected trajectories, and test the zero-shot generalization of the model on a distribution of goals.
- Weak-oracle MBRL (*WO-MBRL*): We train the model $p(s'|s,a)$ on trajectories generated by the planner to navigate to a goal, randomly sampled in every episode. The distribution of goals during training matches the distribution at test time.
- Strong-oracle MBRL (*SO-MBRL*): We train the model $p(s'|s,a)$ on a trajectories generated by the planner to navigate to a specific goal, which is fixed for both training and test time.

Amongst the variants, only the rMBRL matches our assumptions of having an unsupervised task-agnostic training. Both WO-MBRL and SO-MBRL benefit from goal-directed exploration during training, a significant advantage over DADS, which only uses mutual-information-based exploration.

We use $\Delta = \sum_{t=1}^{H} \frac{r(u)}{H\|g\|_2}$ as the metric, which represents the distance to the goal g averaged over the episode (with the same fixed horizon H for all models and experiments), normalized by the initial distance to the goal g. Therefore, lower Δ indicates better performance and $0 < \Delta \leq 1$ (assuming the agent goes closer to the goal). The test set of goals is fixed for all the methods, sampled from $[-15, 15]^2$.

Figure 7 demonstrates that the zero-shot planning significantly outperforms all model-based RL baselines, despite the advantage of the baselines being trained on the test goal(s). For the experiment depicted in Figure 7 (Right), DADS has an unsupervised pre-training phase, unlike SO-MBRL which is training directly for the task. A comparison with Random-MBRL shows the significance of mutual-information-based exploration, especially with the right parameterization and priors. This experiment also demonstrates the advantage of learning a continuous space of primitives, which outperforms planning on discrete primitives.

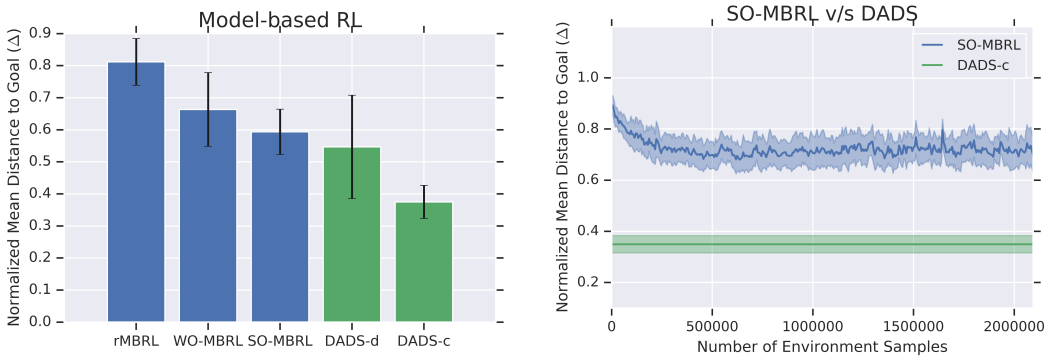


Figure 7: (Left) The results of the MPPI controller on skills learned using DADS-c (continuous primitives) and DADS-d (discrete primitives) significantly outperforms state-of-the-art model-based RL. (Right) Planning for a new task does not require any additional training and outperforms model-based RL being trained for the specific task.

6.4 Hierarchical Control with Unsupervised Primitives

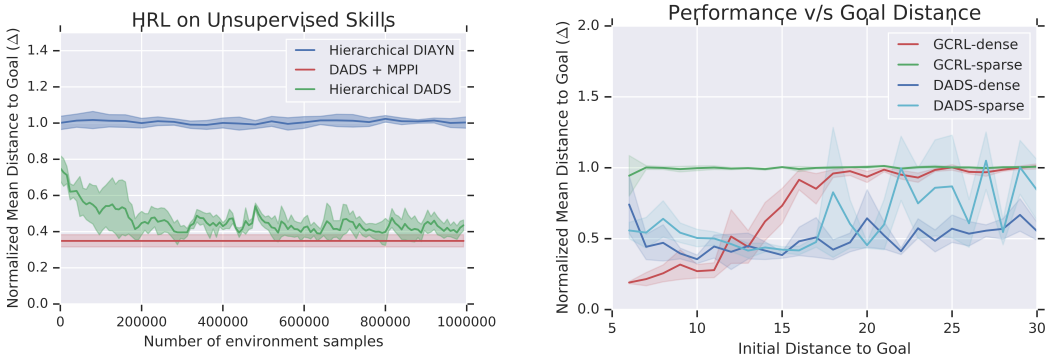


Figure 8: (Left) A RL-trained meta-controller is unable to compose primitive learned by DIAYN to navigate Ant to a goal, while it succeeds to do so using the primitives learned by DADS. (Right) Goal-Conditioned RL (GCRL-dense/sparse) does not generalize outside its training distribution, while MPPI controller on learned skills (DADS-dense/sparse) experiences significantly smaller degrade in performance.

We benchmark hierarchical control for primitives learned without supervision, against our proposed scheme using an MPPI based planner on top of DADS-learned skills. We persist with the task of Ant-navigation as described in 6.3. We benchmark against Hierarchical DIAYN [17], which learns the skills using the DIAYN objective, freezes the low-level policy and learns a meta-controller that outputs the skill to be executed for the next H_Z steps. We provide the x-y prior to the DIAYN's disciminator while learning the skills for the Ant agent. The performance of the meta-controller is constrained by the low-level policy, however, this hierarchical scheme is agnostic to the algorithm used to learn the low-level policy. To contrast the quality of primitives learned by the DADS and DIAYN, we also benchmark against Hierarchical DADS, which learns a meta-controller the same way as Hierarchical DIAYN, but learns the skills using DADS.

From Figure 8 (Left) We find that the meta-controller is unable to compose the skills learned by DIAYN, while the same meta-controller can learn to compose skills by DADS to navigate the Ant to different goals. This result seems to confirm our intuition described in Section 6.2, that the high variance of the DIAYN skills limits their temporal compositionality. Interestingly, learning a RL meta-controller reaches similar performance to the MPPI controller, taking an additional $200,000$ samples per goal.

6.5 Goal-conditioned RL

To demonstrate the benefits of our approach over model-free RL, we benchmark against goal-conditioned RL on two versions of Ant-navigation: (a) with a dense reward $r(u)$ and (b) with a sparse reward $r(u) = 1$ if $\|u - g\|_2 \leq \epsilon$, else 0. We train the goal-conditioned RL agent using soft actor-critic, where the state variable of the agent is augmented with $u - g$, the position delta to the goal. The agent gets a randomly sampled goal from $[-10, 10]^2$ at the beginning of the episode.

In Figure 8 (Right), we measure the average performance of the all the methods as a function of the initial distance of the goal, ranging from 5 to 30 metres. For dense reward navigation, we observe that while model-based planning on DADS-learned skills degrades smoothly as the initial distance to goal to increases, goal-conditioned RL experiences a sudden deterioration outside the goal distribution it was trained on. Even within the goal distribution observed during training of goal-conditioned RL model, skill-space planning performs competitively to it. With sparse reward navigation, goal-conditioned RL is unable to navigate, while MPPI demonstrates comparable performance to the dense reward up to about 20 metres. This highlights the utility of learning task-agnostic skills, which makes them more general while showing that latent space planning can be leveraged for tasks requiring long-horizon planning.

7 Conclusion

We have proposed a novel unsupervised skill learning algorithm that is amenable to model-based planning for hierarchical control on downstream tasks. We show that our skill learning method can scale to high-dimensional state-spaces, while discovering a diverse set of low-variance skills. In addition, we demonstrated that, without any training on the specified task, we can compose the learned skills to outperform competitive model-based baselines that were trained with the knowledge of the test tasks. We plan to extend the algorithm to work with off-policy data, potentially using relabelling tricks [6, 48] and explore more nuanced planning algorithms. We plan to apply the hereby-introduced method to different domains, such as manipulation and enable skill/model discovery directly from images, culminating into unsupervised skill discovery on robotic setups.

8 Acknowledgements

We would like to thank Evan Liu, Ben Eysenbach, Anusha Nagabandi for their help in reproducing the baselines for this work. We are thankful to Ben Eysenbach for their comments and discussion on the initial drafts. We would also like to acknowledge Ofir Nachum, Alex Alemi, Daniel Freeman, Yiding Jiang, Allan Zhou and other colleagues at Google Brain for their helpful feedback and discussions at various stages of this work. We are also thankful to Michael Ahn and others in Adept team for their support, especially with the infrastructure setup and scaling up the experiments.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

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

[3] D. B. F. Agakov. The im algorithm: a variational approach to information maximization. *Advances in Neural Information Processing Systems*, 16:201, 2004.

[4] A. A. Alemi and I. Fischer. Therml: Thermodynamics of machine learning. *arXiv preprint arXiv:1807.04162*, 2018.

[5] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

[6] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. *CoRR*, abs/1707.01495, 2017. URL `http://arxiv.org/abs/1707.01495`.

[7] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[8] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[9] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL `http://arxiv.org/abs/1606.01540`.

[10] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 703–711. JMLR. org, 2017.

[11] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *CoRR*, abs/1805.12114, 2018. URL `http://arxiv.org/abs/1805.12114`.

[12] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4759–4770, 2018.

[13] I. Csiszár and F. Matus. Information projections revisited. *IEEE Transactions on Information Theory*, 49(6):1474–1490, 2003.

[14] C. Daniel, G. Neumann, and J. Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pages 273–281, 2012.

[15] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[16] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2): 408–423, 2013.

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

[18] C. Florensa, Y. Duan, and P. Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

[19] N. Friedman, O. Mosenzon, N. Slonim, and N. Tishby. Multivariate information bottleneck. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 152–161. Morgan Kaufmann Publishers Inc., 2001.

[20] J. Fu, S. Levine, and P. Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4019–4026. IEEE, 2016.

[21] J. Fu, J. Co-Reyes, and S. Levine. Ex2: Exploration with exemplar models for deep reinforcement learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2577–2587. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/6851-ex2-exploration-with-exemplar-models-for-deep-reinforcement-learning.pdf`.

[22] Y. Gal, R. McAllister, and C. E. Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, volume 4, 2016.

[23] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

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

[25] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[26] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2455–2467, 2018.

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

[28] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018. URL `http://arxiv.org/abs/1812.05905`.

[29] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rk07ZXZRb`.

[30] N. Heess, G. Wayne, Y. Tassa, T. Lillicrap, M. Riedmiller, and D. Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

[31] N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, M. Riedmiller, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[32] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. D. Turck, and P. Abbeel. Curiosity-driven exploration in deep reinforcement learning via bayesian neural networks. *CoRR*, abs/1605.09674, 2016. URL `http://arxiv.org/abs/1605.09674`.

[33] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, et al. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[34] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[35] S. Kamthe and M. P. Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. *arXiv preprint arXiv:1706.06491*, 2017.

[36] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[37] A. S. Klyubin, D. Polani, and C. L. Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 128–135. IEEE, 2005.

[38] J. Ko, D. J. Klein, D. Fox, and D. Haehnel. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *Proceedings 2007 ieee international conference on robotics and automation*, pages 742–747. IEEE, 2007.

[39] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard. Gaussian process model based predictive control. In *Proceedings of the 2004 American Control Conference*, volume 3, pages 2214–2219. IEEE, 2004.

[40] V. Kumar, E. Todorov, and S. Levine. Optimal control with learned local models: Application to dexterous manipulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 378–383. IEEE, 2016.

[41] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

[42] I. Lenz, R. A. Knepper, and A. Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*. Rome, Italy, 2015.

[43] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[44] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[45] W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229, 2004.

[46] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[47] S. Mohamed and D. J. Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 2125–2133, 2015.

[48] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3307–3317, 2018.

[49] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[50] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*, pages 2863–2871, 2015.

[51] P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

[52] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

[53] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

[54] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4): 41, 2017.

[55] T. J. Perkins, D. Precup, et al. Using options for knowledge transfer in reinforcement learning. *University of Massachusetts, Amherst, MA, USA, Tech. Rep*, 1999.

[56] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[57] C. E. Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[58] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

[59] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *Icml*, volume 37, pages 1889–1897, 2015.

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

[61] Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Chris Harris, Vincent Vanhoucke, Eugene Brevdo. TF-Agents: A library for reinforcement learning in tensorflow. `https://github.com/tensorflow/agents`, 2018. URL `https://github.com/tensorflow/agents`. [Online; accessed 30-November-2018].

[62] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[63] N. Slonim, G. S. Atwal, G. Tkacik, and W. Bialek. Estimating mutual information and multi–information in large networks. *arXiv preprint cs/0502017*, 2005.

[64] B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *CoRR*, abs/1507.00814, 2015. URL `http://arxiv.org/abs/1507.00814`.

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

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

[67] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. X. Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762, 2017.

[68] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

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

[70] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3540–3549. JMLR. org, 2017.

[71] D. Warde-Farley, T. Van de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018.

[72] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.

[73] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[74] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

A Implementation Details

All of our models are written in the open source Tensorflow-Agents [61], based on Tensorflow [1].

A.1 Skill Spaces

When using discrete spaces, we parameterize $\mathcal{Z}$ as one-hot vectors. These one-hot vectors are randomly sampled from the uniform prior $p(z) = \frac{1}{D}$, where D is the number of skills, usually between 20 and 128. For continuous spaces, we sample $z \sim \text{Uniform}(-1, 1)^D$. We generally vary D from 2 (Ant learnt with x-y prior) to 5 (Humanoid on full observation spaces). The skills are sampled once in the beginning of the episode and fixed for the rest of the episode. However, it is possible to resample the skill from the prior within the episode, which allows for every skill to experience a different distribution than the initialization distribution and encourage skills which are temporally compositional. However, the re-sampling frequency should be such that it happens maximally once or twice per episode, so that every skill has sufficient time to act.

A.2 Agent

We use SAC as the optimizer for our agent $\pi(a \mid s, z)$, in particular, EC-SAC [28]. The s input to the policy generally excludes global co-ordinates (x, y) of the centre-of-mass, available for a lot of enviroments in OpenAI gym, which helps produce skills agnostic to the location of the agent. We restrict to two hidden layers for our policy and critic networks. However, to improve the expressivity of skills, it is beneficial to increase the capacity of the networks. The hidden layer sizes can vary from (128, 128) for Half-Cheetah to (1024, 1024) for Humanoid. The critic $Q(s, a, z)$ is similarly parameterized. The target function for critic Q is updated every iteration using a soft updates with co-efficient of 0.005. We use Adam [36] optimizer with a fixed learning rate of $3e{-}4$, and a fixed entropy co-efficient $\beta = 0.1$. While the policy is parameterized as a normal distribution $\mathcal{N}(\mu(s, z), \Sigma(s, z))$ where Σ is a diagonal covariance matrix, it undergoes through tanh transformation, to transform the output to the range $(-1, 1)$ and constrain to the action bounds.

A.3 Skill-Dynamics

Skill-dynamics, denoted by $q(s' \mid s, z)$, is parameterized by a deep neural network. A common trick in model-based RL is to predict the $\Delta s = s' - s$, rather than the full state s'. Hence, the prediction network is $q(\Delta s \mid s, z)$. Note, both parameterizations can represent the same set of functions. However, the latter will be easy to learn as Δs will be centred around 0. While the global co-ordinates are excluded from the input to q, it is useful to predict Δ_x, Δ_y, because reward functions for goal-based navigation generally rely on the position prediction from the model. The skill-dynamics has the same capacity as the agent/critic with the same hidden layer sizes. The output distribution is modelled as a Mixture-of-Experts [33], where expert is a diagonal state-dependent gaussian, and every expert has weight dependent on the input. The number of experts is 4. Batch-normalization was found to be useful for learning skill-dynamics. However, it is important to turn off the learnable parameters for the last layer for sanity of the learning process.

A.4 Other Hyperparameters

The episode horizon is generally kept shorter for stable agents like Ant (200 usually), while longer for unstable agents like Humanoid (500-1000). For Ant, longer episodes do not add value, but Humanoid can benefit from longer episodes as it helps it filter skills which are unstable. The optimization scheme is on-policy, and generally about 1000-4000 steps are collected in one iteration. The idea is to get about episodes of 5-10 skills in a batch. Re-sampling skills within episodes can be useful if working with longer episodes. Once a batch of episodes is collected, the skill-dynamics is updated using Adam optimizer with a fixed learning rate of 3e-4. The batch size is 128, and generally 20-50 steps of gradient descent are carried out. To compute the intrinsic reward, we need to resample the prior for computing the denominator. For continuous spaces, we set L between 50 to 500. For discrete spaces, we can marginalize over all skills. After the intrinsic reward is computed, the policy and critic networks are updated for 64-128 steps on batch size of 128. This is to ensure that every sample in the batch is seen about 3-4 times, in expectation.

A.5 Planning and Evaluation Setups

For evaluation, we fix the episode horizon to 200 for all models in all evaluation setups. Depending upon the size of the latent space and planning horizon, the number of samples from the planning distribution P is varied between 10-200. The co-efficient γ for MPPI is set to 10. We generally found that setting $H_P = 1$ and $H_Z = 10$ worked well, in which case set the number of refine steps $R = 10$. However, for sparse reward navigation it is important to have a longer horizon planning, in which case we set $H_P = 4, H_Z = 25$ with a higher number of samples from the planning distribution. Also, when using longer planning horizons, we found that smoothing the sampled plans help. Thus, if the sampled plan is $z_1, z_2, z_3, z_4 \ldots$, we smooth the plan to make $z_2 = \beta z_1 + (1 - \beta) z_2$ and so on. The β is generally kept high between 0.8-0.95.

For hierarchical controllers learning on top of low-level unsupervised primitives, we use PPO [60] for discrete action skills, while we use SAC for continuous skills. We keep the number of steps after which the meta-action is decided as 10 (that is $H_Z = 10$). The hidden layer sizes of the meta-controller are (128, 128). We use a learning rate of 1e-4 for PPO and 3e-4 for SAC.

B Graphical models, Information Bottleneck and Unsupervised Skill Learning

We now present a novel perspective on unsupervised skill learning, motivated from the literature on Information Bottleneck. This section takes inspiration from [4], which helps us provide a rigorous justification for our objective proposed earlier. To obtain our unsupervised RL objective, we setup a graphical model P as shown in Figure 9, which represents the distribution of trajectories generated by a given policy π. The joint distribution is given by:

$$p(s_1, a_1 \ldots a_{T-1}, s_T, z) = p(z)p(s_1) \prod_{t=1}^{T-1} \pi(a_t|s_t, z)p(s_{t+1}|s_t, a_t). \tag{11}$$

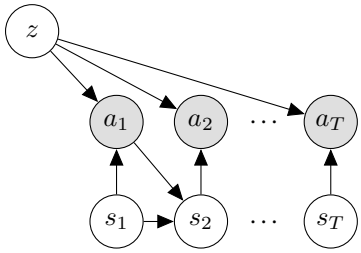


Figure 9: Graphical model for the world P in which the trajectories are generated while interacting with the environment. Shaded nodes represent the distributions we optimize.

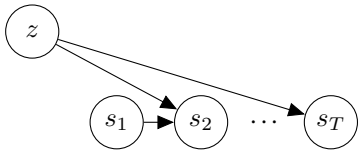


Figure 10: Graphical model for the world Q which is the desired representation of the world.

We setup another graphical model Q, which represents the desired model of the world. In particular, we are interested in approximating $p(s'|s, z)$, which represents the transition function for a particular primitive. This abstraction helps us get away from knowing the exact actions, enabling model-based planning in behavior space (as discussed in the main paper). The joint distribution for Q shown in Figure 10 is given by:

$$q(s_1, \ldots s_T, z) = p(z)p(s_1) \prod_{t=1}^{T-1} q(s_{t+1}|s_t, z). \tag{12}$$

The goal of our approach is to optimize the distribution $\pi(a|s, z)$ in the graphical model P to minimize the distance between the two distributions, when transforming to the representation of the graphical model Q. In particular, we are interested in minimizing the KL divergence between p and q - $\mathcal{D}_{KL}(p||q)$. However, since q is not known apriori, we setup the objective as $\min_{q \in Q} \mathcal{D}_{KL}(p||q)$, which is the reverse information projection [13]. An alternate way to understand the objective is to optimize the distribution P to optimally project onto the graphical model Q. Note, if Q had the same

structure as P, the information lost in projection would be 0 for any valid P. Interestingly, it was shown in [19] that:

$$\min_q \mathcal{D}_{KL}(p||q) = \mathcal{I}_P - \mathcal{I}_Q, \tag{13}$$

where $\mathcal{I}_P$ and $\mathcal{I}_Q$ represents the multi-information for distribution P on the respective graphical models. The multi-information [63] for a graphical model G with nodes g_i is defined as:

$$\mathcal{I}_G = \sum_i I(g_i; Pa(g_i)), \tag{14}$$

where $Pa(g_i)$ denotes the nodes upon which g_i has conditional dependence in G. Using this definition, we can compute the multi-information terms:

$$\mathcal{I}_P = \sum_{t=1}^{T} I(a_t; \{s_t, z\}) + \sum_{t=2}^{T} I(s_t; \{s_{t-1}, a_{t-1}\}) \quad \text{and} \quad \mathcal{I}_Q = \sum_{t=2}^{T} I(s_t; \{s_{t-1}, z\}). \tag{15}$$

Here, $I(s_t; \{s_{t-1}, a_{t-1}\})$ is constant as we assume the underlying dynamics to be fixed (and unknown), and we can safely ignore this term. The final objective to be maximized is given by:

$$R(\pi) = \sum_{t=1}^{T-1} I(s_{t+1}; \{s_t, z\}) - I(a_t; \{s_t, z\}) \tag{16}$$

$$= \sum_{t=1}^{T-1} \mathcal{H}(s_{t+1}) - \mathcal{H}(s_{t+1} \mid s_t, z) - I(a_t; \{s_t, z\}) \tag{17}$$

$$\geq \sum_{t=1}^{T-1} I(s_{t+1}; z \mid s) - I(a_t; \{s_t, z\}) \tag{18}$$

Here, we have used the non-negativity of mutual information, that is $I(s'; s) \geq 0 \implies \mathcal{H}(s') \geq \mathcal{H}(s' \mid s)$. This yields us the objective that we proposed to begin with. This results in an unsupervised skill learning objective that explicitly fits a model for transition behaviors, while providing a grounded connection with probabilistic graphical models. Note, unlike the setup of *control as inference* [43, 74] which casts policy learning as variational inference, the policy here is assumed to be part of the generative model itself (and thus the resulting difference in the direction of $\mathcal{D}_{KL}$).

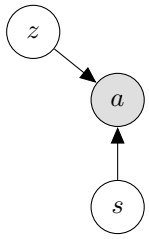


Figure 11: Graphical model for the world P representing the stationary state, action distribution. Shaded nodes represent the distributions we optimize.

Figure 12: Graphical model for the world Q using which we is the representation we are interested in.

We can carry out the exercise for the reward function in Diversity is All You Need (DIAYN) [17] to provide a graphical model interpretation of the objective used in the paper. To conform with objective in the paper, we assume to be sampling to be state-action pairs from skill-conditioned stationary distributions in the world P, rather than trajectories. Again, the objective to be maximized is given by

$$R(\pi) = -\mathcal{I}_P + \mathcal{I}_Q \tag{19}$$

$$= -I(a; \{s, z\}) + I(z; s) \tag{20}$$

$$= \mathbb{E}_\pi[\log \frac{p(z|s)}{p(z)} - \log \frac{\pi(a|s,z)}{\pi(a)}] \tag{21}$$

$$\geq \mathbb{E}_\pi[\log q_\phi(z|s) - \log p(z) - \log \pi(a|s,z)] = R(\pi, q_\phi) \tag{22}$$

where we have used the variational inequalities to replace $p(z|s)$ with $q_\phi(z|s)$ and $\pi(a)$ with a uniform prior over bounded actions $p(a)$ (which is ignored as a constant).

C Interpretation as Empowerment in the Latent Space

Recall, the empowerment objective [47] can be stated as

$$I(s';a|s) = \mathcal{H}(a|s) - \mathcal{H}(a|s',s) \geq \mathcal{H}(a|s) + \mathbb{E}_{p(s)\pi(a|s)p(s'|s,a)}[\log q_\phi(a|s',s))] \tag{23}$$

where the we are are learning a flat policy $\pi(a|s)$, and using the variational approximation $q(a|s',s)$ for the true action-posterior $p(a|s',s)$. We can connect our objective with empowerment if we assume a latent-conditioned policy $\pi(a|s,z)$ and optimize $I(s';z|s)$, which can be interpreted as empowerment in the latent space z. There are two ways to decompose this objective:

$$I(s';z|s) = \mathcal{H}(z|s) - \mathcal{H}(z|s,s') \tag{24}$$

$$= \mathcal{H}(s'|s) - \mathcal{H}(s'|s,z) \tag{25}$$

Using the first decomposition, we can construct a an objective using a variational lower bound which learns the network $q_\phi(z|s',s)$. This is an inference network, which learns to discriminate skills based on the transitions they generate in the environment and not the state-distribution induced by each skill. However, we are interested in learning the network $q_\phi(s'|s,z)$, which is why we work with the second decomposition. But, again we are stuck with marginal transition entropy, which is intractable to compute. We can handle it in a couple of ways:

$$I(s';z|s) \geq \mathbb{E}_s\mathbb{E}_z\mathbb{E}_{p(s'|s,z)}[\log \frac{q_\phi(s'|s,z)}{p(s'|s)}] \tag{26}$$

$$\approx \mathbb{E}_s\mathbb{E}_z\mathbb{E}_{p(s'|s,z)}[\log \frac{q_\phi(s'|s,z)}{\sum_{i=1}^{L} q_\phi(s'|s,z_i)} + \log L] \tag{27}$$

where $p(s'|s)$ represents the distribution of transitions from the state s. Note, we are using the approximation $p(s'|s) = \int p(s'|s,z)p(z)dz \approx \frac{1}{L}\sum_{i=1}^{L} q_\phi(s'|s,z_i)$. Our use of $q(s'|s,z)$ encodes the intuition that the q should represent the distribution of transitions from s under different primitives, and thus the marginal of q over z should approximately represent $p(s'|s)$. However, this procedure does not yield entropy-regularized RL by itself, but arguments similar to those provided for Information Maximization algorithm by [47] can be made here to justify it in this empowerment perspective.

Note, this procedure makes an assumption $p(z|s) = p(z)$ when approximating $p(s'|s)$. While every skill is expected to induce a different state-distribution in principle, this is not a bad assumption to make as we often times expect skills to be almost state-independent (consider locomotion primitives, which can essentially be activated from the state-distribution of any other locomotion primitive). The impact of this assumption can be further attenuated if skills are randomly re-sampled from the prior $p(z)$ within an episode of interaction with the environment. Irrespective, we can avoid making this assumption if we use the variational lower bounds from [3], which is the second way to learn for $I(s';z|s)$. We use the following inequality, used in [29]:

$$\mathcal{H}(x) \geq \int p(x,z)\log\frac{q(z|x)}{p(x,z)}dxdz \tag{28}$$

where q is a variational approximation to the posterior $p(z|x)$.

$$I(s';z|s) = -\mathcal{H}(s'|s,z) + \mathcal{H}(s'|s) \tag{29}$$

$$\geq \mathbb{E}_s\mathbb{E}_{p(s',z|s)}[\log q_\phi(s'|s,z)] + \mathbb{E}_s\mathbb{E}_{p(s',z|s)}[\log q_\alpha(z|s',s)] + \mathcal{H}(s',z|s) \tag{30}$$

$$= \mathbb{E}_s\mathbb{E}_{p(s',z|s)}[\log q_\phi(s'|s,z) + \log q_\alpha(z|s',s)] + \mathcal{H}(s',z|s) \tag{31}$$

where we have used the inequality for $\mathcal{H}(s'|s)$ to introduce the variational posterior for skill inference $q_\alpha(z \mid s',s)$ besides the conventional variational lower bound to introduce $q(s' \mid s,z)$. Further decomposing the leftover entropy:

$$\mathcal{H}(s',z|s) = \mathcal{H}(z|s) + \mathcal{H}(s'|s,z)$$

Reusing the variational lower bound for marginal entropy from [3], we get:

$$\mathcal{H}(s'|s,z) \geq \mathbb{E}_{s,z}\Big[\int p(s',a|s,z)\log\frac{q(a|s',s,z)}{p(s',a|s,z)}ds'da\Big] \tag{32}$$

$$= -\log c + \mathcal{H}(s',a|s,z) \tag{33}$$

$$= -\log c + \mathcal{H}(s'|s,a,z) + \mathcal{H}(a|s,z) \tag{34}$$

Since, the choice of posterior is upon us, we can choose $q(a|s', s, z) = 1/c$ to induce a uniform distribution for the bounded action space. For $\mathcal{H}(s'|s, a, z)$, notice that the underlying dynamics $p(s'|s, a)$ are independent of z, but the actions do depend upon z. Therefore, this corresponds to entropy-regularized RL when the dynamics of the system are deterministic. Even for stochastic dynamics, the analogy might be a good approximation , assuming the underlying dynamics are not very entropic. The final objective (making this low-entropy dynamics assumption) can be written as:

$$I(s'; z|s) \geq \mathbb{E}_s \mathbb{E}_{p(s',z|s)}[\log q_\phi(s'|s, z) + \log q_\alpha(z|s', s) - \log p(z|s)] + \mathcal{H}(a|s, z) \tag{35}$$

We defer experimentation with this objective to future work.

D Interpolation in Continuous Latent Space

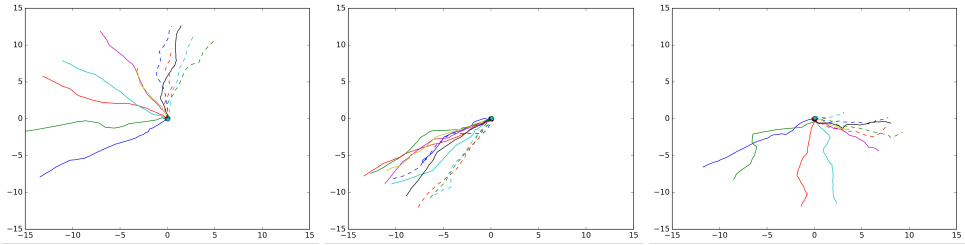


Figure 13: Interpolation in the continuous primitive space learned using DADS on the Ant environment corresponds to interpolation in the trajectory space. (Left) Interpolation from $z = [1.0, 1.0]$ (solid blue) to $z = [-1.0, 1.0]$ (dotted cyan); (Middle) Interpolation from $z = [1.0, 1.0]$ (solid blue) to $z = [-1.0, -1.0]$ (dotted cyan); (Right) Interpolation from $z = [1.0, 1.0]$ (solid blue) to $z = [1.0, -1.0]$ (dotted cyan).

E Model Prediction

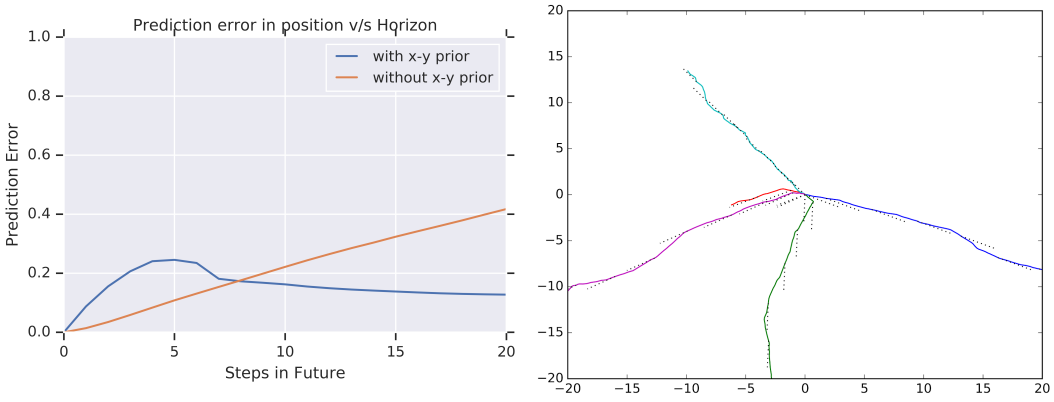


Figure 14: (Left) Prediction error in the Ant's co-ordinates (normalized by the norm of the actual position) for Skill-Dynamics. (Right) X-Y traces of actual trajectories (colored) compared to trajectories predicted by Skill-Dynamics (dotted-black) for different skills.

From Figure 14, we observe that skill-dynamics can provide robust state-predictions over long planning horizons. When learning skill-dynamics with x-y prior, we observe that the error in prediction rises slower with horizon as compared to the norm of the actual position. This provides strong evidence of cooperation between the primitives and skill-dynamics learned using DADS with x-y prior. As the error-growth for skill-dynamics learned on full-observation space is sub-exponential, similar argument can be made for DADS without x-y prior as well (albeit to a weaker extent).