Acquiring Diverse Skills using Curriculum Reinforcement Learning with Mixture of Experts

Onur Celik [*] [†] [§] **Aleksandar Taranovic**[†] **Gerhard Neumann**[†] [§]
[†] Autonomous Learning Robots, Karlsruhe Institute of Technology
[§] FZI Research Center for Information Technology

Abstract

Reinforcement learning (RL) is a powerful approach for acquiring a good-performing policy. However, learning diverse skills is challenging in RL due to the commonly used Gaussian policy parameterization. We propose **Di**verse **Skil**l **L**earning (Di-SkilL), an RL method for learning diverse skills using Mixture of Experts, where each expert formalizes a skill as a contextual motion primitive. Di-SkilL optimizes each expert and its associate context distribution to a maximum entropy objective that incentivizes learning diverse skills in similar contexts. The per-expert context distribution enables automatic curricula learning, allowing each expert to focus on its best-performing sub-region of the context space. To overcome hard discontinuities and multi-modalities without any prior knowledge of the environment's unknown context probability space, we leverage energy-based models to represent the per-expert context distributions and demonstrate how we can efficiently train them using the standard policy gradient objective. Di-SkilL can learn diverse and performant skills on challenging robot simulation tasks.

1 Introduction

Solving tasks in diverse manners enables agents to better adapt to unknown and challenging situations. This diverse skill set is beneficial in many scenarios, such as playing table tennis, where applying different strikes (e.g. backhand, forehand, or smashing) to similar incoming balls is advantageous because the strike is less predictable for the opponent. Similarly, in scenarios with environmental changes where learned skills might be infeasible over time (e.g. grasping an object while avoiding obstacles), diverse skills provide additional adaptivity by discarding these invalid skills and relying on alternatives. This property makes them superior because complete relearning of skills is avoided.

Acquiring these diverse skill sets requires learning a policy that can represent multi-modality in the behavior space. Recent advances in supervised policy learning have demonstrated the potential of training high-capacity policies capable of capturing multi-modal behaviors [1, 2, 3, 4]. These policies exhibit remarkably diverse skills and outperform state-of-the-art methods. However, Reinforcement Learning (RL) is essential to acquire skills in cases where no expert data is available, or data collection is expensive. Discovering multi-modal behaviors using RL is challenging since the policies usually rely on Gaussian parameterization and thus can only discover a single behavior.

We consider training agents that possess diverse skills, from which they can select to tackle a specific task differently. For capturing these multi-modalities in the agent's behavior space, we employ highly non-linear Mixture of Experts policies. Furthermore, we use automatic curriculum learning for efficient learning, enabling each expert to focus on a specific sub-region of the context space it favors. We introduce this curriculum shaping by optimizing for an additional per-expert context distribution

[*] `celik@kit.edu`. Project webpage: `https://alrhub.github.io/di-skill-website/`.
The full version of this work has been accepted at ICML 2024.

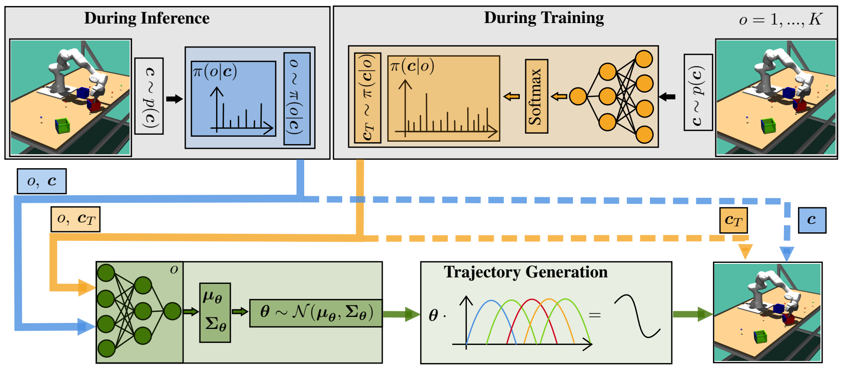

During Inference
During Training
$o = 1, ..., K$
$\mathbf{c} \sim p(\mathbf{c})$
$\pi(o|\mathbf{c})$
$o \sim \pi(o|\mathbf{c})$
$\mathbf{c}_T \sim \pi(\mathbf{c}|o)$
$\pi(\mathbf{c}|o)$
Softmax
$o, \mathbf{c}$
$o, \mathbf{c}_T$
$\mathbf{c}_T$
$\mathbf{c}$
o
$\boldsymbol{\mu}_{\boldsymbol{\theta}}$
$\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$
$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$
Trajectory Generation
$\boldsymbol{\theta} \cdot$
$=$

Figure 1: **The Sampling Procedure for Di-SkilL.** During **Inference** the agent observes contexts $\mathbf{c}$ from the environment's unknown context distribution $p(\mathbf{c})$. The agent calculates the gating probabilities $\pi(o|\mathbf{c})$ for each context and samples an expert o resulting in $(o, \mathbf{c})$ samples marked in blue. During **Training** we first sample a batch of contexts $\mathbf{c}$ from $p(\mathbf{c})$, which is used to calculate the per-expert context distribution $\pi(\mathbf{c}|o)$ for each expert $o = 1, ..., K$. The $\pi(\mathbf{c}|o)$ provides a higher probability for contexts preferred by the expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$. To enable curriculum learning, we provide each expert the contexts sampled from its corresponding $\pi(\mathbf{c}|o)$, resulting in the samples $(o, \mathbf{c}_T)$ marked in orange. In both cases, the chosen $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ samples motion primitive parameters $\boldsymbol{\theta}$ for each context, resulting in a trajectory τ that is subsequently executed on the environment. Before execution, the corresponding context, e.g., the goal position of a box, needs to be set in the environment. This is illustrated by the dashed arrows, with the context in blue for inference and orange for training.

that is used to sample contexts from the preferred regions to train the corresponding expert. Automatic curriculum learning has proven to increase performance by improving the exploration of agents, particularly in sparse-rewarded environments [5].

We explore Contextual Reinforcement Learning in which a continuous-valued context describes the task [6]. In the example of robot table tennis (see Fig. 3a), a context includes the desired ball landing positions on the opponent's tableside as well as physical aspects, such as the incoming ball's velocity or friction properties. In continuous context spaces, the curriculum shaping per-expert context distributions are often parameterized as Gaussian [7, 8]. However, the agent is usually unaware of the context bounds, which makes additional techniques necessary to constrain the distribution updates to stay within the context region [8]. Instead, we employ energy-based per-expert context distributions, which can be evaluated for any context and effectively represent multi-modality in the context space. Importantly, our model is trained solely using context samples from the environment that are inherently valid. Our approach eliminates the need for additional regularization of the context distribution and does not require prior knowledge about the environment. Due to the overlapping probability distributions of different per-expert contexts, our resulting mixture policy offers diverse solutions for similar contexts with a high probability.

Recent research in RL has explored Mixture of Experts policies, but often these methods either train the mixture in unsupervised RL settings and then select the best-performing expert in the downstream task [9, 10] or train linear experts, limiting their performance [11, 8]. Our inspiration draws from recent advancements that have achieved diverse skill learning with a similar objective. However, their approach involves linear expert models with Gaussian context distributions. It requires prior knowledge of the environment to design a penalty term when the algorithm samples contexts outside the environment's bounds. These factors restrict the algorithm's performance and applicability when defining the context bounds requires knowledge, such as forward kinematics in robotics.

To summarize, we introduce a novel RL method for learning a Mixture of Experts policy that we refer to as **Di-SkilL** – **Di**verse **Skil**l **L**earning (see Fig. 1). Our method can generalize to the continuous range of contexts defined by the (unknown) environment's context distribution while learning diverse, and non-linear skills for solving a task defined by a specific context. Importantly, our approach operates without any assumptions about the environment. We show how we can learn multi-modal context distributions by training an energy-based model solely on context samples obtained from the environment. On multiple sophisticated simulated robot tasks, we demonstrate that we can learn diverse skills while performing on par or better than baselines.

2 Preliminaries

Contextual Episode-based Policy Search (CEPS). We consider learning diverse skills in the CEPS framework in which the continuous-valued context $\mathbf{c} \in \mathcal{C}$ defines the task, e.g. a goal location to reach. The context $\mathbf{c} \sim p(\mathbf{c})$ is observed from the agent and is drawn from the environment's unknown context distribution $p(\mathbf{c})$ at the beginning of each episode. The agent's search distribution $\pi(\boldsymbol{\theta}|\mathbf{c})$ maps the context $\mathbf{c}$ to continuous-valued controller parameters $\boldsymbol{\theta} \in \Theta$, which we represent as motion primitives (MP) [12, 13, 14] (see Appendix C). We denote $\pi(\boldsymbol{\theta}|\mathbf{c})$ as the agent's policy as common in the literature and optimize it by maximizing the objective

$$\max_{\pi(\boldsymbol{\theta}|\mathbf{c})} \mathbb{E}_{p(\mathbf{c})}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c})}[\mathrm{R}(\mathbf{c},\boldsymbol{\theta})]\right], \tag{1}$$

where $\mathrm{R}(\mathbf{c},\boldsymbol{\theta})$ denotes the return of a whole episode after executing the MP parameter $\boldsymbol{\theta}$ in context $\mathbf{c}$. Due to the direct return optimization, CEPS does not require the Markov assumption as in common MDPs and is therefore specifically suitable for tasks where the formulation of a Markovian reward function is difficult.

Mixture of Experts (MoE) Policy for Curriculum Learning. Due to their ability to represent multi-modality, MoE policies are a favorable choice in diverse skill learning. The common MoE policy $\pi(\boldsymbol{\theta}|\mathbf{c}) = \sum_o \pi(o|\mathbf{c})\pi(\boldsymbol{\theta}|\mathbf{c},o)$ [15] contains the gating distribution $\pi(o|\mathbf{c})$ that is assigning probabilities to each expert o given context $\mathbf{c}$ during inference. However, to enable automatic curriculum learning during training, a learnable distribution $\pi(\mathbf{c}) = \sum_o \pi(\mathbf{c}|o)\pi(o)$ is required that can explicitly choose and set context samples in the environment, so each expert o can decide on which contexts it favors training [8]. Using Bayes' rule $\pi(o|\mathbf{c}) = \pi(\mathbf{c}|o)\pi(o)/\pi(\mathbf{c})$ the MoE is rewritten as

$$\pi(\boldsymbol{\theta}|\mathbf{c}) = \sum_o \frac{\pi(\mathbf{c}|o)\pi(o)}{\pi(\mathbf{c})}\pi(\boldsymbol{\theta}|\mathbf{c},o). \tag{2}$$

The per-expert context distribution $\pi(\mathbf{c}|o)$ can now be optimized and allows the expert o to choose contexts $\mathbf{c}$ it favors. We model each $\pi(\mathbf{c}|o)$ as an energy-based model and each $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ as a neural network returning a Gaussian distribution for a context $\mathbf{c}$ (see Fig. 1 and Appendix C). The prior $\pi(o)$ is set to a uniform distribution throughout this work.

Self-Paced Diverse Skill Learning with MoE. Due to its ability to represent multi-modality and automatic curriculum learning, the MoE model in Eq. 2 is a suitable policy representation for discovering diverse skills in the same context-defined task. For explicit optimization of this policy, we are using the KL-regularized Maximum Entropy RL objective in CEPS [8]

$$\max_{\pi(\boldsymbol{\theta}|\mathbf{c}),\pi(\mathbf{c})} \mathbb{E}_{\pi(\mathbf{c})}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c})}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta})\right] + \alpha \mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c})\right]\right] - \beta \mathrm{KL}\left(\pi(\mathbf{c}) \parallel p(\mathbf{c})\right). \tag{3}$$

The KL-term incentivizes the context distribution $\pi(\mathbf{c})$ to match the environment's distribution $p(\mathbf{c})$ and can be prioritized during optimization by choosing the scaling parameter β appropriately. The entropy of the mixture model incentivizes learning diverse solutions [8] and can be prioritized with a high scaling parameter α. It is well-known that this objective is difficult to optimize for MoE policies and requires further steps to obtain a tractable lower-bound [8]

$$\max_{\pi(\boldsymbol{\theta}|\mathbf{c},o)} \mathbb{E}_{\pi(\mathbf{c}|o),\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha \log \tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right] + \alpha \mathbb{E}_{\pi(\mathbf{c}|o)}\left[\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c},o)\right]\right] \tag{4}$$

for the expert $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ updates and a lower-bound for the per-expert context $\pi(\mathbf{c}|o)$ updates

$$\max_{\pi(\mathbf{c}|o)} \mathbb{E}_{\pi(\mathbf{c}|o)}\left[L_c(o,\mathbf{c}) + (\beta-\alpha)\log\tilde{\pi}(o|\mathbf{c})\right] + \beta \mathrm{H}\left(\pi(\mathbf{c}|o)\right), \tag{5}$$

where $L_c(o,\mathbf{c}) = \mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha\log\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right] + \alpha\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c},o)\right]$. The variational distributions $\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta}) = \pi_{old}(o|\mathbf{c},\boldsymbol{\theta})$ and $\tilde{\pi}(o|\mathbf{c}) = \pi_{old}(o|\mathbf{c})$ arise through the decomposition and are responsible for learning diverse solutions and concentrating on context regions with small, or no support by $\pi(\mathbf{c})$ [8]. In every iteration, the variational distributions are updated in closed form to tighten the bounds. Details of the equations are in the Appendix A.

3 Related Work

Contextual Episode-based Policy Search (CEPS). CEPS is a black-box approach to reinforcement learning (RL), in which the search distribution is the agent's policy that maps the contexts to controller parameters, typically represented as motion primitives [12, 13, 14]. One of the noteworthy advantages of CEPS lies in the independence of assumptions such as the Markovian property in common MDPs. This characteristic renders it a versatile methodology, particularly well-suited for addressing a diverse array of intricate tasks where the formulation of a Markovian reward function is difficult [16]. CEPS has been explored by applying various optimization techniques, including Policy Gradients [17], Natural Gradients [18], stochastic search strategies [19, 20, 21], and trust-region optimization techniques [22, 11, 23], particularly in the non-contextual setting. Researchers extended the setting by incorporating linear [23, 21] and non-linear contextual adaptation [16, 24], leveraging the recently introduced trust-region layers for neural networks [25]. The work by [24] additionally introduces step-wise updates to improve sample-efficiency. All previously mentioned methods learn single-mode policies and do not address acquiring diverse skills leveraging automatic curriculum learning.

Curriculum Reinforcement Learning. Curriculum reinforcement learning can potentially increase the performance of RL agents, especially in sparse-rewarded environments [26] in which exploration is fundamentally difficult. Adapting the environment based on the agent's learning process has been proposed by several works already, e.g. automatically generating sets of tasks or goals to increase the learning speed of the agent [27, 28, 29, 30, 31, 32], or generating a curriculum by interpolating an auxiliary and known distribution of target tasks [5, 7, 33, 34]. Works propose sampling a training level from a prespecified set of environments [35], or unsupervised environment design [36, 37] based on the agent's learning process. The work by [38] proposes improving the approximation of the state-action value function by representing it as a sum of residuals acquired in previous curriculum tasks. None of the above methods apply automatic curriculum learning on an RL problem with an MoE policy, except for the work in [8]. However, they parameterize the curriculum distribution as Gaussian, suffering from low representation capacity and requiring knowledge about the environment's context distribution. Instead, we leverage energy-based models to avoid these shortcomings.

RL with Mixture of Experts (MoE). In [39] using an MoE policy representation and a novel gradient estimator to calculate the gradients w.r.t. the MoE parameters is proposed. In [40] a model-based RL approach to train latent variable models is presented. The work presents a novel lower bound for training the multi-modal policy parameterization. Recently, in [41] an MoEs for learning a shared representation in multi-task reinforcement learning is presented, whereas works proposed training interpretable MoEs in continuous RL [42]. These methods differ from our work in that they are not categorized in the CEPS framework, or are model-based variants and do not use automatic curriculum learning techniques. In the CEPS framework, RL with MoE policies has also been explored [11, 43], in which an MoE model with linear experts without automatic curriculum learning is learned. Additional constraints need to be added to enforce diversity in the experts. In [44] a mixture model is used to perform RL, however, pre-recorded demonstration data is required to train the mixture model and no curriculum learning is considered. Related method to MoEs, Product of Experts was used in [45, 46] for motion generation. The work in [8] also uses MoE policies and relies on the maximum entropy objective as we do, however, their method only considers linear experts with Gaussian per-expert distributions which limits the performance and consequently requires many experts to solve a task. Moreover, it requires environment knowledge to hand-tune a punishment term to keep the optimization of the per-expert context distributions within the context bounds.

We discuss related works in **Quality-Diversity Optimization** and **Unsupervised RL** in the App. B.

4 Diverse Skill Learning

The common Contextual Episodic Policy Search (CEPS) loop [6] with a Mixture of Experts (MoE) policy representation learning observes a context $\mathbf{c}$, and then selects an expert o that subsequently adjusts the controller parameters $\boldsymbol{\theta}$ given ($\mathbf{c}$, o). We consider the same process during testing time, as shown in blue color in Fig. 1 (see also Fig. 7a). However, the procedure changes during training for Di-SkilL as automatic curriculum learning requires that the agent can determine which context regions it prefers to focus on. In this case, we observe a batch of context samples from the environment's context distribution $p(\mathbf{c})$. For each of these samples, every per-expert context distribution $\pi(\mathbf{c}|o)$ calculates a probability, which results in a categorical distribution over the contexts $\mathbf{c}$. We use these probabilities to sample contexts $\mathbf{c}_T$ for the corresponding expert o resulting in $(\mathbf{c}_T, o)$ sample pairs (see orange parts in Fig. 1 and Fig. 7b). Each chosen expert o provides Gaussian distributions over the motion primitive parameters $\boldsymbol{\theta}$ by mapping the contexts $\mathbf{c}_T$ to mean vectors $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and covariance matrices $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ using a parameterized neural network. We can now sample motion primitive parameters $\boldsymbol{\theta}$ from these Gaussian distributions to generate trajectories τ using a motion primitive generator. These trajectories are subsequently executed on the environment (green color in Fig. 1) and an episode return $R(\boldsymbol{\theta}, \mathbf{c}_T)$ is observed and used for updating the MoE (see Section 4.2). Yet, there exist several issues for a stable overall training of the MoE model, which requires special treatment for each $\pi(\mathbf{c}|o)$ and $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$. Details (algorithmic and model specific) can be found in the Appendix C.

4.1 Energy-Based Model For Automatic Curriculum Learning

To illustrate these issues, we consider a bounded, uniformly distributed two-dimensional environment context distribution $p(\mathbf{c})$ (see example in the Appendix C in Fig. 7c). It is challenging for a Reinforcement Learning (RL) agent to automatically learn its curriculum $\pi(\mathbf{c}|o)$ within the valid context space [8]. Hard discontinuities such as steps often naturally arise in $p\left(\mathbf{c}\right)$ due to the environment's finite support in real-world environments. For instance, in an environment where the agent's task is to place an object in specific positions on a table, the probability of observing a goal position outside the table's surface is zero. This implies that a large subset of the context space has no probability mass. Therefore, exploration in these regions might be difficult if there is no guidance encoded in the reward. Even if it is guaranteed that $\pi(\mathbf{c}|o)$ only samples valid contexts, it still needs to be able to represent multi-modal distributions, such as illustrated in Fig. 7d. This multi-modality can be present because of environmental circumstances or simply if experts $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ prefer contexts in spatially apart regions. For the object placing example, this could correspond to regions on the table where the object cannot be placed due to obstacles or holes. We therefore require $\pi(\mathbf{c}|o)$ being able to represent **i)** complex distributions, **ii)** multi-modality and **iii)** only explore within the valid context bounds of $p\left(\mathbf{c}\right)$. We propose parameterizing each per-expert distribution $\pi(\mathbf{c}|o)$ as an energy-based model

$$\pi(\mathbf{c}|o) = \exp(\boldsymbol{\phi}_o(\mathbf{c}))/Z \tag{6}$$

to address the issues i) and ii), where the energy function ϕ_o is a per-expert learnable neural network. Energy-based models (EBMs) have shown to be capable of representing sharp discontinued functions and multi-modal distributions [47]. Yet, they are hard to train and sample from due to the intractable normalizing constant $Z = \int_{\mathbf{c}} \exp(\boldsymbol{\phi}_o(\mathbf{c}))d\mathbf{c}$. We can circumvent and address these issues iii) by approximating the normalizing constant with contexts $\mathbf{c} \sim p\left(\mathbf{c}\right)$ as $Z \approx \sum_{i=1}^{N} \exp(\phi_o(\mathbf{c}_i))$. This approximation is justified as we can sample from $p\left(\mathbf{c}\right)$ by simply resetting the environment without execution. Additionally, the EBM will encounter important parts of the context space during the training by resampling a large enough batch of contexts $\mathbf{c} \sim p\left(\mathbf{c}\right)$ in each iteration. Each expert can therefore sample preferred contexts from the current batch of valid contexts by calculating the probability for each of the contexts using $\pi(\mathbf{c}|o)$ as parameterized in Eq. 6. Updating the parameters of the EBM can readily be addressed by the standard RL objective for diverse skill learning, as described in the next section. It should be noted that explicit models such as Gaussians, or Normalizing Flows [48] can also be used to parameterize $\pi(\mathbf{c}|o)$, but their support cannot be easily restricted to a bounded space with hard discontinuities defined by the environment. Therefore, sampling from an explicit $\pi(\mathbf{c}|o)$ can easily generate invalid contexts, especially if the valid distribution has hard non-linearities.

4.2 Updating the Mixture of Experts Model

We update each expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ and its corresponding per-expert context distribution $\pi(\mathbf{c}|o)$ by maximizing the objectives in Eq. 4 and in Eq. 5, respectively. These decomposed objectives allow us to independently update both distributions and to retain the properties of diverse skill learning from the objective in Eq. 9. However, updating the distributions is not straightforward due to the bi-level optimization that leads to a dependency on both terms. This is particularly challenging for the expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ as the sampled contexts $\mathbf{c}$ can drastically change from one iteration to another if $\pi(\mathbf{c}|o)$ changes too aggressively. The same applies for updating $\pi(\mathbf{c}|o)$ as calculating the objective requires calculating an integral over $\boldsymbol{\theta}$ under the expectation of $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$. For a stable update for both distributions, we employ trust-region updates to restrict the change of both distributions from one iteration to another. These updates have been shown to improve the learning process [49, 50, 51, 25].

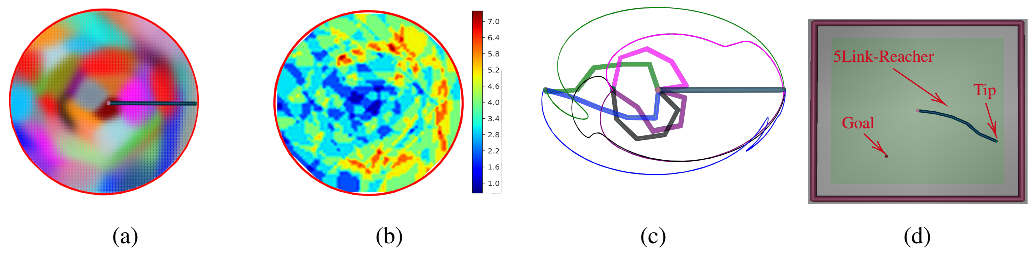


Figure 2: **a)** High-probability regions of the individual per-expert context distributions $\pi(\mathbf{c}|o)$, where a color represents an expert o. The red circle marks the context space of goal-reaching positions for the 5-Link Reacher's tip. The specialization of $\pi(\mathbf{c}|o)$ is induced by $\tilde{\pi}(o|\mathbf{c})$. **b)** Different $\pi(\mathbf{c}|o)$ need to overlap for learning diverse skills. This overlapping is induced by the entropy bonus $\mathrm{H}\left[\pi(\mathbf{c}|o)\right]$. **c)** Different tip trajectories sampled in the same contexts. The trajectories and the end joint constellation are in the same color. The diversity in the parameter space is induced by $\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})$. **d)** Visualization of the 5-Link Reacher task (**5LR**).

Expert Update. We parameterize each expert $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ with a single neural network and update them by a trust-region constrained optimization

$$\max_{\pi(\boldsymbol{\theta}|\mathbf{c},o)} \mathbb{E}_{\pi(\mathbf{c}|o),\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta})+\alpha\log\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right]+\alpha\mathbb{E}_{\pi(\mathbf{c}|o)}\left[\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c},o)\right]\right] \tag{7}$$
$$\text{s.t.}\quad \mathrm{KL}\left(\pi(\boldsymbol{\theta}|\mathbf{c},o)\,\|\,\pi_{\text{old}}(\boldsymbol{\theta}|\mathbf{c},o)\right)\leq\epsilon\qquad\forall\,\mathbf{c}\in\mathcal{C},$$

where the KL-bound ensures that the expert $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ does not differ too much from the expert $\pi_{\text{old}}(\boldsymbol{\theta}|\mathbf{c},o)$ from the iteration before for each context $\mathbf{c}$. The entropy bonus $\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c},o)\right]$ incentivizes $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ to fully cover the parameter space while avoiding $(\boldsymbol{\theta},\mathbf{c})$ regions that are covered by other experts o. The latter is guaranteed by $\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})$ which rewards $(\boldsymbol{\theta},\mathbf{c})$ regions that can be assigned to expert o with high probability. We efficiently update the experts using trust region layers [25, 16].

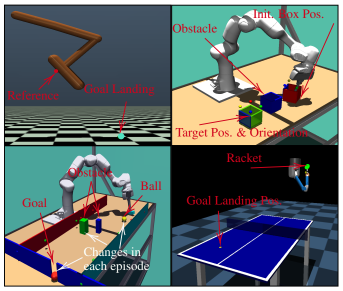
Per-Expert Context Distribution Update. We consider the objective with the augmented rewards as shown in Eq. 5 for updating each context distribution $\pi(\mathbf{c}|o)$. We can not apply the trust region layers [25] in this case, as $\pi(\mathbf{c}|o)$ is a discrete distribution over the context samples $\mathbf{c}_i$ parameterized by the EBM. Yet, we can still use PPO [51] for updating $\pi(\mathbf{c}|o)$ and simplifying our objective, as we can now calculate many terms in closed form. For this, we rewrite the objective as

$$\max_{\pi(\mathbf{c}|o)}\sum_{\mathbf{c}_i\sim p(\mathbf{c})}\pi(\mathbf{c}_i|o)L_c(o,\mathbf{c}_i)+\sum_{\mathbf{c}_i\sim p(\mathbf{c})}\pi(\mathbf{c}_i|o)\big((\beta-\alpha)\log\tilde{\pi}(o|\mathbf{c}_i)-\beta\log\pi(\mathbf{c}_i|o)\big) \tag{8}$$

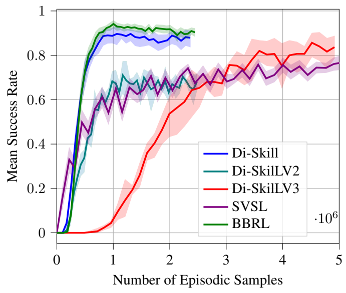
and observe that all terms in the second sum can be calculated in closed form. The first term is approximated by resampling the contexts using $\pi(\mathbf{c}|o)$ since computing $L_c(o,\mathbf{c})$ requires calculating the integral over $\boldsymbol{\theta}$ under the expectation of $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ as $L_c(o,\mathbf{c})=\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta})+\alpha\log\tilde{\pi}(o|\mathbf{c})\right]+\alpha\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c},o)\right]$. This expectation can only be estimated for context vectors that are actually chosen by the component. The entropy bonus in Eq. (8) incentivizes covering of the context space, while focusing on context regions that are not, or only partly covered by other options. The latter is guaranteed by $\tilde{\pi}(o|\mathbf{c})$ which assigns a high probability if expert o can be assigned to the context $\mathbf{c}$.

4.3 How does Diversity Emerge?

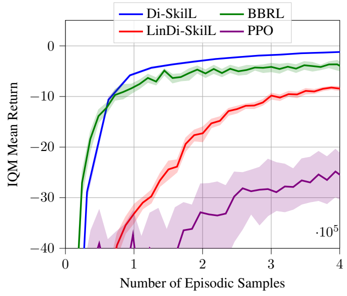
From the Eq. 7 and Eq. 8 it is clear that diverse behaviors, represented by the experts, emerge from the interplay of those terms in Eq. 7 and Eq. 8. We visually demonstrate the meaning of the individual terms on the 5-Link Reacher task (see Fig. 2d). The Reacher needs to reach a goal position in the two-dimensional space with its tip. In this task, a context represents the goal position within the context space, visualized as a red circle around the reacher's fixed first joint (Fig. 2a). We trained Di-SkilL with 50 experts. In Fig. 2a we show the high-probability regions of the individual per-expert context distributions $\pi(\mathbf{c}|o)$, by setting the color intensity proportional to this probability. Each color represents an individual expert o. Each $\pi(\mathbf{c}|o)$ concentrates on a sub-region of the context space such that the corresponding $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ becomes an expert there. This specialization is incentivized by the term $\tilde{\pi}(o|\mathbf{c})$ in Eq. 8. However, for learning diverse behaviors for the same context regions, it is necessary that the per-expert context distributions $\pi(\mathbf{c}|o)$ overlap, which is motivated by the entropy



(a) Visualization of Environments



(b) Mean Success Rate (**TT**)



(c) IQM Mean Return (**5LR**)

Figure 3: **a)** (left top) Hopper Jump (**HJ**). (Top right) Box Pushing with Obstacle (**BPO**). (Bottom Left) Robot Mini Golf (**MG**). (Bottom right) Robot table tennis (**TT**). **b)** Ablation studies, showcasing the need for automatic curriculum learning for Di-SkilL. BBRL and Di-SkilL can solve TT environment decently. Di-SkilL's variants without curriculum learning struggle to achieve a good performance. SVSL needs more samples to achieve around 80% success rate, suffering under the linear experts. **c)** Performance of Di-SkilL, BBRL, LinDi-SkilL, and PPO on 5LR with sparse in-time rewards.

term $\mathrm{H}\left[\pi(\mathbf{c}|o)\right]$ in Eq. 8. These overlapping context regions are visualized in Fig. 2b, where we count how many experts o are active for each context. The figure shows that more experts prefer regions close to the initial position of the reacher, indicating that these contexts are easier to solve. Despite the closeness to the reacher's initial position, the agent does not have to exert much energy to reach these points. Indeed, both aspects are present in the task's reward function (see App. D for details), explaining why the left half plane of the context space has fewer overlapping. However, the learned MoE has two or more experts active in most parts of the context region. These experts differ in their behavior (see Fig. 2), which is motivated by the terms $\tilde{\pi}(o|\mathbf{c}, \boldsymbol{\theta})$ and $\mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c}, o)\right]$ in Eq. 7.

5 Experiments

In our evaluations, we compare Di-SkilL against the baselines **BBRL** [16] and **SVSL** [8]. Whenever the environment satisfies the Markov properties, we additionally compare against **PPO** [51]. BBRL and SVSL are suitable baselines as they are state-of-the-art CEPS algorithms that can learn complex skills. BBRL can learn highly non-linear policies leveraging trust region updates. SVSL learns a linear Mixture of Experts (MoE) model and can capture multi-modality in the behavior space. We consider challenging robotic environments with continuous context and parameter spaces. The considered environments either have a non-Markovian reward function, i.e. require retrospective data for calculation, or temporally sparse reward functions increasing the learning complexity.

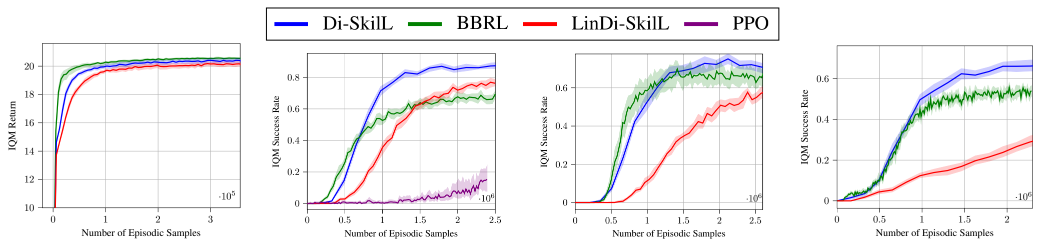
Environments. Environment visualizations are in Fig. 3a. Further information is in Appendix C.

Table Tennis (TT). A 7-degree of freedom (DoF) robot has to learn fast and precise motions to smash the ball to a desired position on the opponent's side. The 4-dim. context consists of the incoming ball's landing position and the desired ball's landing position on the opponent's side. The TT environment requires good exploratory behavior and has a non-markovian reward structure making step-based approaches infeasible to learn useful skills [16].

Table Tennis Hard (TT-H). We extend the TT environment to a more challenging version by varying the ball's initial velocity. This additionally increases the learning complexity, as the agent now needs to reason about the physical effects of changed velocity ranges and requires improved adaptability.

5-Link Reacher (5LR). The 5-Link reacher has to reach a goal position within all quadrants in the context space (see Fig. 2a) as opposed to the version in [16], where the multi-modality in the behavior space (see Fig. 2c) was avoided by constraining the context space to the upper half of the context space. Additionally, the time-sparse reward makes this task a challenging benchmark.

Hopper Jump (HJ). Presented in [16] in which the Hopper [52] is tasked to jump as high as possible while landing in a goal position. The HJ has a non-markovian reward, making step-based RL methods unfeasible to learn useful policies [16].



(a) Mean Return (**HJ**) (b) Succ. Rate (**BPO**) (c) Succ. Rate (**TT-Hard**) (d) Succ. Rate (**MG**)

Figure 4: **Performance on the a) HJ b) BPO**, **c) TT-H**, and **d) MG** tasks. **a)** Di-SkilL performs on par with BBRL on the HJ task. **b)** The multi-modality introduced by the obstacle in the box pushing task leads to worse performance for BBRL than for Di-SkilL and LinDi-SkilL. PPO suffers under the time-sparse reward setting. **c)** BBRL converges faster, Di-SkilL achieves a higher success rate eventually. **d)** Di-SkilL outperforms the baselines on the **MG** task. LinDi-SkilL performs poorly on the non-Markovian rewarded tasks TT-H and MG, indicating that non-linear policies are beneficial.

Box Pushing with Obstacle (BPO). A 7DoF robot is tasked to push a box to a target position and rotation while avoiding an obstacle. In addition to the time-spare reward [16], our version includes the obstacle and considers a larger range of the box's target position.

Mini Golf (MG). The 7DoF robot has to hit the ball such that it passes the tight goal while avoiding the obstacles (static, varying). The context is the obstacle's, the goal's, and the ball's position. The env. has a non-markov. reward, making step-based RL methods unfeasible to learn useful policies.

5.1 ACL Benefits

Automatic Curriculum learning (ACL) enables Di-SkilL's experts to shape their curriculum by explicitly sampling from preferred context regions. We analyze the importance of this feature by comparing the performance of variants of Di-SkilL on the table tennis (TT) task. For both variants *Di-SkilLV2* and *Di-SkilLV3* we disable ACL by fixing the term induced by the variational distribution to $\log \tilde{\pi}(o|\mathbf{c}) = 0$ in Eq. 8 and by setting the entropy scaling parameter $\beta = 2000$. Ignoring the variational distribution $\tilde{\pi}(o|\mathbf{c})$ during training eliminates the intrinsic motivation of the per-expert context distribution $\pi(\mathbf{c}|o)$ to focus on sub-regions in the context space that are not, or only partially, covered by any other $\pi(\mathbf{c}|o)$ (Section 4.3). Setting $\beta = 2000$ incentivizes each $\pi(\mathbf{c}|o)$ to maximize its entropy, resulting in a uniform distribution in the environment's bounded context space. For Di-SkilL we keep the ACL and set $\beta = 4$. We provide the same number of 50 context-parameter samples per expert for *Di-SkilLV2* and Di-SkilL, whereas *Di-SkilLV3* receives 260 samples per expert in each iteration. All variants possess 5 experts. In Fig. 3b we report the mean success rates and the 95% confidence interval for each method on at least 4 seeds. *Di-SkilLV2* converges to a much smaller success rate, and *Di-SkilLV3* needs more samples to reach the level of Di-SkilL. BBRL and Di-SkilL achieve high success rates, while BBRL performs slightly better. SVSL shows worse performance, even though the model has 20 experts. The results indicate that ACL is an essential feature of Di-SkilL ensuring that Di-SkilL can learn high-perfroming skills with fewer samples. Moreover, SVSL's poor performance shows that Gaussian parameterized per-expert context distributions that require additionally tuned punishment terms for guided updates are together with linear experts incapable of achieving a satisfying performance.

5.2 Analyzing the Performance and Diversity

For a detailed analysis, we have evaluated all methods on *24 seeds* for each environment and algorithm and report the *interquantile mean* (IQM) with a 95% stratified bootstrap confidence interval as suggested by [53]. Please note that SVSL requires designing a punishment function to guide the context samples in the environment's valid context region, which makes its application difficult, especially if the context influences the objects' physics. We therefore propose comparing against LinDi-SkilL instead of SVSL. LinDi-SkilL also has linear experts but benefits from Di-SkilL's energy-based per-expert context distribution $\pi(\mathbf{c}|o)$ eliminating the need for punishment functions.

The performance curve of the **HJ** task in Fig. 4a shows that Di-SkilL performs on par with BBRL, while BBRL converges slightly faster. Both methods can solve the task, indicating that the task doesn't

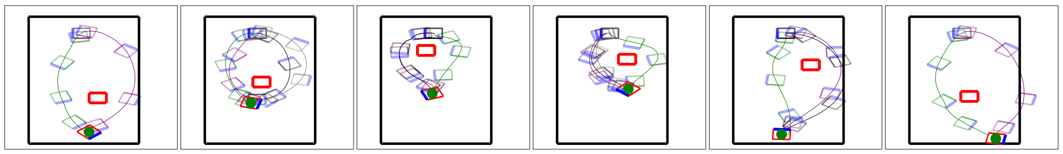
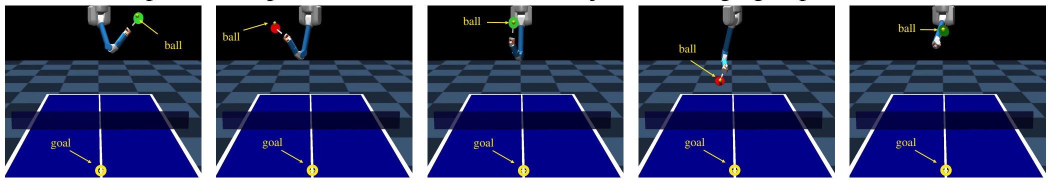


Figure 5: Di-SkilL's **Diverse Skills** for the Box Pushing with Obstacle **BPO** Task. The figures visualize diverse solutions to the same contexts **c** on a table (black rectangle). The red, thick rectangle represents the obstacle. The 7DoF robot is tasked to push the box (shown in different colors for each solution found) to the goal box position (red rectangle with a green dot) and align the blue edges to match the orientation. We visualized successful box trajectories for each sampled skill. The diversity learned in the parameter space results in different box trajectories ranging in position and orientation.



(a) Backhand Drive (b) Forehand Drive (c) Backhand Block (d) Forehand Push (e) Backhand Smash

Figure 6: Di-SkilL's **Diverse Skills** for the **TT-H** task. We fixed the ball's desired landing position and varied the serving landing position and the ball's initial velocity.

require diversity. LinDi-SkilL achieves a similar performance as BBRL and Di-SkilL but needs more samples to converge. We provide additional analysis of this task in Appendix E. Fig. 4b shows the performance curves on the **BPO** task. The obstacle introduces multi-modality in the behavior space. This multi-modality explains why DiSkilL and LinDi-SkilL outperform BBRL, while Di-SkilL achieves the highest success rate. PPO's poor performance indicates that time-correlated exploration as used with motion primitives is effective in sparse rewarded tasks. A similar performance behavior can be observed in the **5LR** task. In Fig. 3c we report the achieved returns and observe that Di-SkilL outperforms BBRL due to the ability to capture multi-modal behaviors (e.g. reaching from different sides) while PPO suffers from the sparse rewarded setting. LinDi-SkilL's linear experts cause slow convergence, indicating that more experts are needed to effectively cover the whole context space. For both tasks, Di-SkilL's diverse skills in the parameter space $\boldsymbol{\theta}$ induce different behaviors. Fig. 5 shows diverse box trajectories to several fixed goal and obstacle positions in the BPO task, whereas Fig. 2c shows different tip trajectories to several fixed goal positions in the 5LR task.

The non-Markovian rewarded tasks (**TT-H** and **MG**) show that non-linear policies as learned by BBRL and Di-SkilL are beneficial. Di-SkilL and BBRL perform similarly well on the TT-H task (see Fig. 4c), where Di-SkilL achieves a slightly higher end success rate compared to BBRL. However, there is a clear performance gap between Di-SkilL and BBRL on the MG task (see Fig. 4d) with Di-SkilL outperforming BBRL. In both tasks, LinDi-SkilL performs worse than Di-SkilL and BBRL indicating that linear experts are insufficient for solving these tasks. Di-SkilL can discover diverse striking styles in the table tennis task (TT-H). Fig. 6 shows some of these learned skills. Additional strike visualizations are in Appendix E.

6 Conclusion and Future Work

We proposed a novel method for learning diverse skills using a contextual Mixture of Experts. Each expert learns its curriculum by optimizing for a per-expert context distribution $\pi(\mathbf{c}|o)$. We have demonstrated challenges that arise through enabling automatic curriculum learning (ACR) and proposed parameterizing $\pi(\mathbf{c}|o)$ as energy-based models (EBMs) to address these challenges. Additionally, we provided a methodology to efficiently optimize these EBMs. We also proposed using trust-region updates for the deep experts to stabilize our bi-level optimization problem. In an ablation, we have shown that ACR is necessary for efficient and performant learning. Moreover, in sophisticated robot simulation environments, we have shown that our method can learn diverse skills while performing on par or better than the baselines. Currently, the major drawback of our approach is that it is not able to replan, causing failures in the tasks if the robot even has small collisions with objects. We intend to address this issue in future research. To improve the sample complexity of our approach, we additionally plan to use off-policy RL techniques.

References

[1] Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

[2] Denis Blessing, Onur Celik, Xiaogang Jia, Moritz Reuss, Maximilian Xiling Li, Rudolf Lioutikov, and Gerhard Neumann. Information maximizing curriculum: A curriculum-based approach for learning versatile skills. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[3] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[4] Xiaogang Jia, Denis Blessing, Xinkai Jiang, Moritz Reuss, Atalay Donat, Rudolf Lioutikov, and Gerhard Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. In *The Twelfth International Conference on Learning Representations*, 2024.

[5] Pascal Klink, Haoyi Yang, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Curriculum reinforcement learning via constrained optimal transport. In *International Conference on Machine Learning*, pages 11341–11358. PMLR, 2022.

[6] Andras Kupcsik, Marc Deisenroth, Jan Peters, and Gerhard Neumann. Data-efficient generalization of robot skills with contextual policy search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 27, pages 1401–1407, 2013.

[7] Pascal Klink, Hany Abdulsamad, Boris Belousov, and Jan Peters. Self-paced contextual reinforcement learning. In *Conference on Robot Learning*, pages 513–529. PMLR, 2020.

[8] Onur Celik, Dongzhuoran Zhou, Ge Li, Philipp Becker, and Gerhard Neumann. Specializing versatile skill libraries using local mixture of experts. In *Conference on Robot Learning*, pages 1423–1433. PMLR, 2022.

[9] Misha Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. Urlb: Unsupervised reinforcement learning benchmark. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021.

[10] 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.

[11] Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pages 273–281. PMLR, 2012.

[12] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer, 2006.

[13] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. *Advances in neural information processing systems*, 26, 2013.

[14] Ge Li, Zeqi Jin, Michael Volpp, Fabian Otto, Rudolf Lioutikov, and Gerhard Neumann. Prodmp: A unified perspective on dynamic and probabilistic movement primitives. *IEEE Robotics and Automation Letters*, 8(4):2325–2332, 2023.

[15] Christopher Bishop. Pattern recognition and machine learning. *Springer google schola*, 2:531–537, 2006.

[16] Fabian Otto, Onur Celik, Hongyi Zhou, Hanna Ziesche, Vien Anh Ngo, and Gerhard Neumann. Deep black-box reinforcement learning with movement primitives. In *Conference on Robot Learning*, pages 1244–1265. PMLR, 2023.

[17] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

[18] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

[19] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[20] Shie Mannor, Reuven Y Rubinstein, and Yohai Gat. The cross entropy method for fast policy search. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 512–519, 2003.

[21] Abbas Abdolmaleki, David Simoes, Nuno Lau, Luís Paulo Reis, and Gerhard Neumann. Contextual direct policy search: With regularized covariance matrix estimation. *Journal of Intelligent & Robotic Systems*, 96:141–157, 2019.

[22] Abbas Abdolmaleki, Rudolf Lioutikov, Jan R Peters, Nuno Lau, Luis Pualo Reis, and Gerhard Neumann. Model-based relative entropy stochastic search. *Advances in Neural Information Processing Systems*, 28, 2015.

[23] Voot Tangkaratt, Herke Van Hoof, Simone Parisi, Gerhard Neumann, Jan Peters, and Masashi Sugiyama. Policy search with high-dimensional context variables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[24] Ge Li, Hongyi Zhou, Dominik Roth, Serge Thilges, Fabian Otto, Rudolf Lioutikov, and Gerhard Neumann. Open the black box: Step-based policy updates for temporally-correlated episodic reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2023.

[25] Fabian Otto, Philipp Becker, Ngo Anh Vien, Hanna Carolin Ziesche, and Gerhard Neumann. Differentiable trust region layers for deep reinforcement learning. *arXiv preprint arXiv:2101.09207*, 2021.

[26] Stone Tao, Arth Shukla, Tse-kai Chan, and Hao Su. Reverse forward curriculum learning for extreme sample and demonstration efficiency in rl. 2024.

[27] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.

[28] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

[29] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations*, 2018.

[30] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33:7648–7659, 2020.

[31] Jan Wöhlke, Felix Schmitt, and Herke van Hoof. A performance-based start state curriculum framework for reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1503–1511, 2020.

[32] Sebastien Racaniere, Andrew Lampinen, Adam Santoro, David Reichert, Vlad Firoiu, and Timothy Lillicrap. Automated curriculum generation through setter-solver interactions. In *International Conference on Learning Representations*, 2020.

[33] Pascal Klink, Carlo D'Eramo, Jan R Peters, and Joni Pajarinen. Self-paced deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:9216–9227, 2020.

[34] Pascal Klink, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. On the benefit of optimal transport for curriculum reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–15, 2024.

[35] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pages 4940–4950. PMLR, 2021.

[36] Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34:1884–1897, 2021.

[37] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

[38] Pascal Klink, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Boosted curriculum reinforcement learning. In *International Conference on Learning Representations*, 2022.

[39] Jie Ren, Yewen Li, Zihan Ding, Wei Pan, and Hao Dong. Probabilistic mixture-of-experts for efficient deep reinforcement learning. *arXiv preprint arXiv:2104.09122*, 2021.

[40] Zhiao Huang, Litian Liang, Zhan Ling, Xuanlin Li, Chuang Gan, and Hao Su. Reparameterized policy learning for multimodal trajectory optimization. *ICML*, 2023.

[41] Ahmed Hendawy, Jan Peters, and Carlo D'Eramo. Multi-task reinforcement learning with mixture of orthogonal experts. In *The Twelfth International Conference on Learning Representations*, 2024.

[42] Riad Akrour, Davide Tateo, and Jan Peters. Continuous action reinforcement learning from a mixture of interpretable experts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6795–6806, 2021.

[43] Felix End, Riad Akrour, Jan Peters, and Gerhard Neumann. Layered direct policy search for learning hierarchical skills. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6442–6448. IEEE, 2017.

[44] Samuele Tosatto, Georgia Chalvatzaki, and Jan Peters. Contextual latent-movements off-policy optimization for robotic manipulation skills. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10815–10821. IEEE, 2021.

[45] Kay Hansel, Julen Urain, Jan Peters, and Georgia Chalvatzaki. Hierarchical policy blending as inference for reactive robot control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10181–10188. IEEE, 2023.

[46] An Thai Le, Kay Hansel, Jan Peters, and Georgia Chalvatzaki. Hierarchical policy blending as optimal transport. In *Learning for Dynamics and Control Conference*, pages 797–812. PMLR, 2023.

[47] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[48] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1):2617–2680, 2021.

[49] Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1607–1612, 2010.

[50] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[51] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[52] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[53] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

[54] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[55] Olle Nilsson and Antoine Cully. Policy gradient assisted map-elites. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 866–875, 2021.

[56] Maxence Faldor, Félix Chalumeau, Manon Flageat, and Antoine Cully. Map-elites with descriptor-conditioned gradients and archive distillation into a single policy. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 138–146, 2023.

[57] Maxence Faldor, Félix Chalumeau, Manon Flageat, and Antoine Cully. Synergizing quality-diversity with descriptor-conditioned reinforcement learning. *arXiv preprint arXiv:2401.08632*, 2023.

[58] Leon Keller, Daniel Tanneberg, Svenja Stark, and Jan Peters. Model-based quality-diversity search for efficient robot learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9675–9680. IEEE, 2020.

[59] Victor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giró i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1317–1327. PMLR, 2020.

[60] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[61] Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6736–6747. PMLR, 18–24 Jul 2021.

[62] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020.

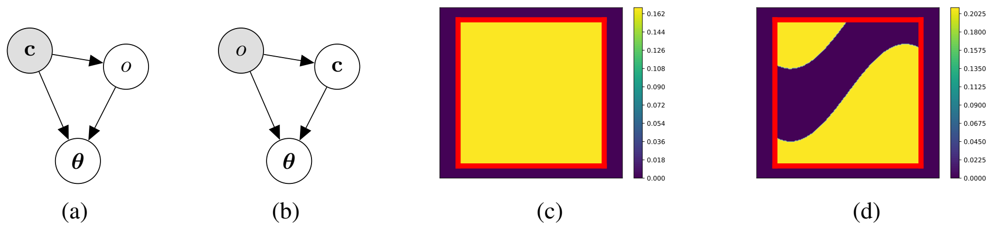


Figure 7: Probabilistic Graphical Models (PGMs) during **inference a)** and **training b)**. During **a))** the model observes the contexts $\mathbf{c}$ from the environment. An expert o is sampled from $\pi(o|\mathbf{c})$, which leads to an adjustment of the motion primitive parameters $\boldsymbol{\theta}$ by $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$. We iterate over each expert during **(b)**, sample the contexts $\mathbf{c}$ and $\boldsymbol{\theta}$ from the per-expert distribution $\pi(\mathbf{c}|o)$ and $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ respectively. Sampling from $\pi(\mathbf{c}|o)$ allows shaping the expert's curriculum. **c)** illustrates the environment's context distribution $p(\mathbf{c})$ and a possibly optimal $\pi(\mathbf{c}|o)$ **(d))** in two-dim. space. Yellow areas indicate high and purple zero probability. The illustrations show that optimizing $\pi(\mathbf{c}|o)$ requires dealing with i) step-like non-linearities, ii) multi-modality, iii) bounded within the red rectangle support of $p(\mathbf{c})$, complicating exploration.

A Additional Information to Self-Paced Diverse Skill Learning with MoE

The general self-paced diverse skill learning objective

$$\max_{\pi(\boldsymbol{\theta}|\mathbf{c}), \pi(\mathbf{c})} \mathbb{E}_{\pi(\mathbf{c})} \left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c})} \left[\mathrm{R}(\mathbf{c}, \boldsymbol{\theta}) \right] + \alpha \mathrm{H} \left[\pi(\boldsymbol{\theta}|\mathbf{c}) \right] \right] - \beta \mathrm{KL} \left(\pi(\mathbf{c}) \parallel p(\mathbf{c}) \right)$$

can be reformulated to

$$\max_{\pi(\mathbf{c}, \boldsymbol{\theta})} \mathbb{E}_{\pi(o), \pi(\mathbf{c}|o)} \left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c}, o)} \left[\mathrm{R}(\mathbf{c}, \boldsymbol{\theta}) + \alpha \log \pi(o|\mathbf{c}, \boldsymbol{\theta}) \right] + \beta \log p(\mathbf{c}) + (\beta - \alpha) \log \pi(o|\mathbf{c}) \right] \quad (9)$$

$$+ \alpha \mathbb{E}_{\pi(o), \pi(\mathbf{c}|o)} \left[\mathrm{H} \left[\pi(\boldsymbol{\theta}|\mathbf{c}, o) \right] \right] + \beta \mathbb{E}_{\pi(o)} \left[\mathrm{H} \left[\pi(\mathbf{c}|o) \right] \right] + \beta \mathrm{H} \left[\pi(o) \right], \quad (10)$$

by inserting $\pi(\boldsymbol{\theta}|\mathbf{c})$, $\pi(\mathbf{c})$ from Eq. (2) into Eq. (9) and applying Bayes theorem. This objective is not straightforward to optimize for Mixture of Experts MoE models and requires further steps to introduce a lower bound (see Section 2) that can be efficiently optimized. Please note that the variational distributions in Eq. 4 and Eq. 5 can be calculated in closed form by the identities

$$\tilde{\pi}(o|\mathbf{c}, \boldsymbol{\theta}) = \pi_{old}(o|\mathbf{c}, \boldsymbol{\theta}) = \frac{\pi_{old}(\boldsymbol{\theta}|\mathbf{c}, o)\pi_{old}(o|\mathbf{c})}{\pi_{old}(\boldsymbol{\theta}|\mathbf{c})}$$

$$\tilde{\pi}(o|\mathbf{c}) = \pi_{old}(o|\mathbf{c}) = \frac{\pi_{old}(\mathbf{c}|o)\pi(o)}{\pi_{old}(\mathbf{c})}$$

We refer the interested reader to [8] for a detailed derivation.

B Additional Related Work

Quality-Diversity Optimization (QDO). Learning diverse skills has also been explored in the evolutionary strategy community, most notably with the MAP-Elites algorithm [54], where behavioral descriptors are defined to distinguish the different learned motions. Extensions [55, 56, 57] have been proposed to improve the performance of these methods. However, these methods can not easily be applied to the contextual setting where different controller parameters should be chosen in different situations such that post hoc adaptations [58, 57] are required. In contrast to QDO methods, in our work diversity measurement naturally arises through the considered objective and does not need defining behavioral descriptors. Moreover, Di-SkilL indirectly learns a gating distribution that selects the expert after observing a context.

Unsupervised Reinforcement Learning. Another field of research that considers learning diverse policies is unsupervised reinforcement learning (URL). In URL the agent is first trained solely with an intrinsic reward to acquire a diverse set of skills from which the most appropriate is picked to solve a downstream task. More related to our work is a group of algorithms that obtain their intrinsic reward based on information-theoretic formulations [9, 10, 59, 60, 61]. However, their resulting objective is based on the mutual-information and differs from the objective we maximize. The learned skills in the pre-training aim to cover distinct parts of the state-space during pre-training in the absence of an extrinsic task reward which implies that skills are not explicitly trained to solve the same task in different ways. Those methods operate within the step-based RL setting which differs from CEPS.

C Additional Information to Diverse Skill Learning

C.1 The Parameterization of the Mixture of Experts (MoE) Model

In the following, we provide details on the parameterization of the MoE model.

Parametrization of the expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$. We parameterize each expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ as a Gaussian policy $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\gamma}}(\mathbf{c}), \boldsymbol{\Sigma}_{\boldsymbol{\gamma}}(\mathbf{c})$, where the mean $\boldsymbol{\mu}_{\boldsymbol{\gamma}}(\mathbf{c})$ and the covariance $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}}(\mathbf{c})$ are functions of the context $\mathbf{c}$ and parameterized by a neural network with parameters $\boldsymbol{\gamma}$. Although the covariance $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}}(\mathbf{c})$ is formalized as a function of the context $\mathbf{c}$, we have not observed any advantages in doing so. In our experiments, we therefore parameterize the covariance as a lower-triangular matrix $\mathbf{L}$ and form the covariance matrix $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$.

Parameterization of the per-expert context distribution $\pi(\mathbf{c}|o)$. The reader is referred to Section 4 for details on the parameterization of $\pi(\mathbf{c}|o)$

Parameterization of the prior $\pi(o)$. We fix the prior $\pi(o)$ to a uniform distribution over the number *K* of available components and do not further optimize this distribution. This is a useful definition to increase the entropy of the mixture model.

Parameterization of the context distribution $\pi(\mathbf{c})$**.** Due to the relation $\pi(\mathbf{c}) = \sum_o \pi(\mathbf{c}|o)\pi(o)$, $\pi(\mathbf{c})$ is defined by $\pi(\mathbf{c}|o)$ and does not need explicit modelling.

Parameterization of the gating distribution $\pi(o|\mathbf{c})$. Due to the relation $\pi(o|\mathbf{c}) = \frac{\pi(\mathbf{c}|o)\pi(o)}{\pi(\mathbf{c})}$ we do not need an explicit parameterization of $\pi(o|\mathbf{c})$ and can easily calculate the probabilities for choosing the expert o given a context $\mathbf{c}$.

C.2 Using Motion Primitives in the Context of Reinforcement Learning

Motion Primitives (MPs) are a low-dimensional representation of a trajectory. For instance, instead of parameterizing a desired joint-level trajectory as the single state in each time step, MPs introduce a low-dimensional parameter vector $\boldsymbol{\theta}$ which concisely defines the trajectory to follow. The generation of the trajectory depends on the method that is used. Probabilistic Movement Primitives (ProMPs) [13] for example define the desired trajectory as a simple linear function $\boldsymbol{\tau} = \boldsymbol{\Phi}^T\boldsymbol{\theta}$, where $\boldsymbol{\Phi}$ are time-dependent basis functions (e.g. normalized radial basis functions). Dynamic Movement Primitives (DMPs) [12] rely on a second-order dynamic system that provides smooth trajectories in the position and velocity space. Recently Probabilistic Dynamic Movement Primitives (ProDMPs) were introduced in [14] and combines the advantages of both methods, that is the easy generation of trajectories and smooth trajectories. We therefore rely on ProDMPs throughout this work.

In the context of reinforcement learning, the policy $\pi(\boldsymbol{\theta}|\mathbf{c})$, or in our case an expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ defines a distribution over the parameters $\boldsymbol{\theta}$ of the MP depending on the observed context $\mathbf{c}$. This allows the policy to quickly adapt to new tasks defined by $\mathbf{c}$.

C.3 Algorithm Details

Detailed descriptions of the algorithm during training and during inference are provided in the algorithm boxes Alg. 1 and Alg. 1, respectively. In each iteration during training, we sample a batch of contexts $\mathbf{c}$ from the environment by resetting it. We then iterate over each expert and evaluate the probabilities of these contexts $\mathbf{c}$ on each per-expert context distribution $\pi(\mathbf{c}|o)$ and sample then training contexts $\mathbf{c}_T$ from them. From the corresponding expert $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ we sample motion primitive parameters $\boldsymbol{\theta}$ and evaluate the samples $(\mathbf{c}_T, \boldsymbol{\theta})$ on the environment and observe a return $\mathrm{R}(\mathbf{c}, \boldsymbol{\theta})$ which we use to update the experts $\pi(\boldsymbol{\theta}|\mathbf{c}, o)$ and the per-expert context distributions $\pi(\mathbf{c}|o)$ by maximizing Obj. 7 and Obj. 8 respectively. During inference, we observe contexts $\mathbf{c}$ from the environment, calculate the gating distributions $\pi(o|\mathbf{c}) = \frac{\pi(\mathbf{c}|o)\pi(o)}{\pi(\mathbf{c})}$ from which we sample the expert o. We then either take the mean or sample an $\boldsymbol{\theta}$ from this expert and execute it on the environment.

Algorithm 1 Di-SkilL Training

```
Input: $\alpha$, $\beta$, N(max. iterations), K(num. experts),T(num. samples per expert)
Output: $\pi(\boldsymbol{\theta}|\mathbf{c})$
 1: for $k = 1$ to N do
 2:     $\mathbf{c} \sim p(\mathbf{c})$ (context batch by environment resetting)
 3:     for $o = 1$ to $K$ do
 4:         $\mathbf{c}_T \sim \pi(\mathbf{c}|o)$ (context batch from EBM)
 5:         $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta}|\mathbf{c}_T, o)$
 6:         $\mathrm{R}(\mathbf{c}, \boldsymbol{\theta}) \leftarrow$ eval($\boldsymbol{\theta}, \mathbf{c}_T$)
 7:         $\pi(\boldsymbol{\theta}|\mathbf{c}, o) \leftarrow$ Obj. 7
 8:         $\pi(\mathbf{c}|o) \leftarrow$ Obj. 8
 9:     end for
10: end for
```

Algorithm 2 Di-SkilL Inference

```
Input: $\pi(\boldsymbol{\theta}|\mathbf{c})$
 1: $\mathbf{c} \sim p(\mathbf{c})$ (observe contexts from environment)
 2: $o \sim \pi(o|\mathbf{c})$, where $\pi(o|\mathbf{c}) = \frac{\pi(\mathbf{c}|o)\pi(o)}{\pi(\mathbf{c})}$
 3: $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta}|\mathbf{c}, o)$
 4: $\mathrm{R}(\mathbf{c}, \boldsymbol{\theta}) \leftarrow$ eval($\boldsymbol{\theta}, \mathbf{c}$)
```

D Experimental Details

D.1 Environment Details

D.1.1 Table Tennis Easy

Environment. We use the same table tennis environment as presented in [16], in which a 7 Degree of Freedom (DoF) robot has to return a ball to a desired ball landing position. The **context** is the four-dimensional space of the ball's initial landing position ($x \in [-1, -0.2]$, $y \in [-0.65, 0.65]$) on the robot's table side and the desired ball landing position ($x \in [-1.0, -0.2]$, $y \in [-0.6, 0.6]$) on the opponent's table side. The robot is controlled with torques on the joint level in each time step. The torques are generated by the tracking controller (PD-controller) that tracks the desired trajectory generated by the motion primitive. We consider three basis functions per joint resulting in a 21-dimensional parameter ($\boldsymbol{\theta}$) space. We additionally allow the agent to learn the trajectory length and the starting time step of the trajectory. Note that the starting point allows the agent to define when after the episode's start the generated desired trajectory should be tracked. Induced by the varying contexts, this is helpful to react to the varying time the served ball needs to reach a positional space that is convenient to hit the ball with the robot's racket. Overall the **parameter space** is 23 dimensional. The task is considered successful if the returned ball lands on the opponent's side of the table and within ≤ 0.2m to the goal location.

The **reward function** is unchanged from [16] and is defined as

$$R_{task} = \begin{cases} 0, & \text{if cond. 1,} \\ f_2(\mathbf{p}_r, \mathbf{p}_b) & \text{if cond. 2,} \\ f_3(\mathbf{p}_r, \mathbf{p}_b, \mathbf{p}_l, \mathbf{p}_{goal}) & \text{if cond. 3,} \\ f_4(\mathbf{p}_r, \mathbf{p}_b, \mathbf{p}_l, \mathbf{p}_{goal}) & \text{if cond. 4,} \\ f_5(\mathbf{p}_r, \mathbf{p}_b, \mathbf{p}_l, \mathbf{p}_{goal}) & \text{if cond. 5,} \end{cases}$$

where $\mathbf{p}_r$ is the executed trajectory position of the racket center, $\mathbf{p}_b$ is the executed position trajectory of the ball, $\mathbf{p}_l$ is the ball landing position, $\mathbf{p}_{goal}$ is the target position. The individual functions are defined as

$$\begin{aligned} f_2(\mathbf{p}_r, \mathbf{p}_b) &= 0.2 - 0.2g(\mathbf{p}_r, \mathbf{p}_b) \\ f_3(\mathbf{p}_r, \mathbf{p}_b, \mathbf{p}_l, \mathbf{p}_{goal}) &= 3 - 2g(\mathbf{p}_r, \mathbf{p}_b) - h(\mathbf{p}_l, \mathbf{p}_{goal}) \\ f_4(\mathbf{p}_r, \mathbf{p}_b, \mathbf{p}_l, \mathbf{p}_{goal}) &= 6 - 2g(\mathbf{p}_r, \mathbf{p}_b) - 4h(\mathbf{p}_l, \mathbf{p}_{goal}) \\ f_5(\mathbf{p}_r, \mathbf{p}_b, \mathbf{p}_l, \mathbf{p}_{goal}) &= 7 - 2g(\mathbf{p}_r, \mathbf{p}_b) - 4h(\mathbf{p}_l, \mathbf{p}_{goal}), \end{aligned}$$

where $g(\mathbf{x}, \mathbf{y}) = \tanh(\min ||\mathbf{x} - \mathbf{y}||^2)$ and $h(\mathbf{x}, \mathbf{y}) = \tanh(||\mathbf{x} - \mathbf{y}||^2)$. The different conditions are

- cond. 1: the end of the episode is not reached,
- cond. 2: the end of the episode is reached,
- cond. 3: cond.2 is satisfied and the robot did hit the ball,
- cond. 4: cond.3 is satisfied and the returned ball lands on the table,
- cond. 5: cond.4 is satisfied and the landing position is at the opponent's side.

The episode ends when any of the following conditions are met

- the maximum horizon length is reached
- ball did land on the floor without hitting
- ball did land on the floor or table after hitting

The whole desired trajectory is obtained ahead of environment interaction, making use of this property we can collect some samples without physical simulation. The reward function based on this desired trajectory is defined as

$$r_{traj} = -\sum_{(i,j)} |\tau_{ij}^d| - |q_j^b|, \quad (i,j) \in \{(i,j) \mid |\tau_{ij}^d| > |q_j^b|\}$$

where τ^d is the desired trajectory, i is the time index, j is the joint index, q^b is the joint position upper bound. The desired trajectory is considered as invalid if $r_{traj} < 0$, an invalid trajectory will not be executed on the robot. The overall reward is defined as:

$$r = \begin{cases} r_{traj}, & r_{traj} < 0 \\ r_{task}, & \text{otherwise} \end{cases}$$

SVSL. SVSL requires designing a guiding punishment term for context samples that are not in a valid region. For the four-dimensional context space in table tennis, this can be done using quadratic functions (as proposed in the original work [8]):

$$R_c(\mathbf{c}) = -20 \cdot d_c^2,$$

where d_c^2 is the distance of the current context $\mathbf{c}$ to the valid context region.

SVSL Hyperparameters All hyperparameters are summarized in the Table 1.

Hyperparameters are listed in the Table 2.

D.1.2 Table Tennis Task Hard

Environment. We extend the table tennis environment described in Appendix D.1.1 by additionally including the ball's initial velocity in the context space making the task harder as the agent has to react to ranging velocities now. We define the initial velocity $v_x \in [1.5\frac{m}{s}, 4\frac{m}{s}]$. Note that every single constellation within the resulting context space is a valid context. However, there exist ball landing positions that can not be set along with a subset of the initial velocity range. This makes designing a guiding punishment term for SVSL especially difficult. We adopt the **parameter space** and the **reward function** as defined in the standard table tennis environment as described in Appendix D.1.1.

Hyperparameters are listed in the Table 4.

D.2 Hopper Jump

Environment. We use the same hopper jump environment as presented in [16], in which the hopper [52] has to jump as high as possible and land at a specified position. The **context** is the four-dimensional space of the last three joints of the hopper and the goal landing position $[j_3, j_4, j_5, g]$, where the ranges are from $[-0.5, -0.2, 0, 0.3]$ to $[0.0, 0.0, 0.785, 1.35]$. The hopper is controlled the same as in [52]. Here, we consider three basis functions per joint and a goal basis resulting in a **parameter space** ($\boldsymbol{\theta}$) of 12 dimensions. The reward is non-markovian and is unchanged from [16].

In each time-step t the action cost

$$\tau_t = 10^{-3} \sum_i^K (a_t^i)^2,$$

is provided. The variable $K = 3$ corresponds to the number of degrees of freedom. At the end of the episode, a reward containing retrospective information about the maximum height in the z-direction of the center of mass achieved h_{max}, the goal landing position of the heel p_{goal}, the foot's heel position when having contact with the ground after jumping the first time $p_{\text{foot, contact}}$ is given. Additionally, per-time information such as $p_{\text{foot, t}}$ describing the position of the foot's heel in world-coordinates is given. The resulting reward function is

$$R_{tot} = -\sum_{t=0}^{T} \tau_t + R_{height} + R_{gdist} + R_{cdist} + R_{healthy},$$

where

$$\begin{aligned} R_{height} &= 10 h_{max}, \\ R_{gdist} &= ||p_{foot,T} - p_{goal}||_2, \\ R_{cdist} &= ||p_{foot,contact} - p_{goal}||_2, \\ R_{healthy} &= \begin{cases} 2 & \text{if } z_T \in [0.5, \infty] \text{and } \theta, \gamma, \phi \in [-\infty, \infty] \\ 0 & \text{else.} \end{cases} \end{aligned}$$

The healthy reward is the same as provided by [52].

Hyperparameters are listed in the Table 5.

D.2.1 Box Pushing with Obstacle Task

Environment. We increase the difficulty of the box pushing environment as presented in [16], by changing major parts of the context space. The goal of the box pushing task is to move a box to a specified goal location and orientation using the seven DoF Franka Emika Panda. The newly **context space** (compared to the original version in [16]) are described in the following. We increase the box' goal position range to $x_g \in [0.3, 0.6], y_g \in [-0.7, 0.45]$, and keep the goal orientation angle $\phi \in [0rad, 2\pi rad]$. Additionally, we include an obstacle between the initial box and the box's goal. The range of the obstacle position is $x_o \in [0.3, 0.6], y_o \in [-0.3, 0.15]$. Note that we guarantee a distance of at least 0.15m between the obstacle's position and the initial position as well as at least 0.15m between the obstacle's position and the box's goal position.

The robot is controlled via torques on the joint level. We use four basis functions per DoF, resulting in a **parameter space** of 28 dimensions. We consider an episode successful if the box's orientation around the z-axis error is smaller than 0.5 rad and the position error is smaller than 0.05m.

The **sparse-in-time reward function** is up to a scaling parameter the same as presented in [16]. We describe the whole reward function in the following.

The box's distance to the goal position is

$$R_{\text{goal}} = ||\mathbf{p} - \mathbf{p}_{goal}||,$$

where $\mathbf{p}$ is the box position and $\mathbf{p}_{goal}$ is the goal position. The rotation distance is defined as

$$R_{\text{rotation}} = \frac{1}{\pi} \arccos |\mathbf{r} \cdot \mathbf{r}_{goal}|,$$

where $\mathbf{r}$ and $\mathbf{r}_{goal}$ are the box orientation and goal orientation quaternion respectively. The incentive to keep the rod within the box is defined as

$$R_{\text{rod}} = \text{clip}(||\mathbf{p} - \mathbf{h}_{pos}||, 0.05, 10),$$

where $\mathbf{h}_{pos}$ is the position of the rod tip. Similarly, to incentivize to maintain the rod in a desired rotation, the reward

$$R_{\text{rod\_rotation}} = \text{clip}(\frac{2}{\pi} \arccos |\mathbf{h}_{rot} \cdot \mathbf{h}_0|, 0.25, 2)$$

is defined, where $\mathbf{h}_{rot}$ and $\mathbf{h}_0 = (0, 1, 0, 0)$ are the current and desired rod orientation in quaternion respectively. To incentivize the robot to stay within the joint and velocity bounds, the error

$$\begin{aligned} \text{err}(\mathbf{q}, \dot{\mathbf{q}}) = & \sum_{i \in \{i || q_i | > |q_i^b|\}} (|q_i| - |q_i^b|) \\ & + \sum_{j \in \{j || \dot{q}_j | > |\dot{q}_j^b|\}} (|\dot{q}_j| - |\dot{q}_j^b|) \end{aligned}$$

is used, where $\mathbf{q}$, $\dot{\mathbf{q}}$, $\mathbf{q}^b$, and $\dot{\mathbf{q}}^b$ are the robot's joint positions and velocities as well as their respective bounds. To learn low-energy motions, the per-time action (torque) cost

$$\tau_t = \sum_i^K (a_t^i)^2,$$

is used. The resulting temporal sparse reward is given as

$$R_{\text{tot}} = \begin{cases} -R_{\text{rod}} - R_{\text{rod\_rotation}} - 0.02\tau_t - \text{err}(\mathbf{q}, \dot{\mathbf{q}}) & t < T, \\ -R_{\text{rod}} - R_{\text{rod\_rotation}} - 0.02\tau_t - \text{err}(\mathbf{q}, \dot{\mathbf{q}}) & \\ -350R_{\text{goal}} - 200R_{\text{rotation}} & t = T, \end{cases}$$

where $T = 100$ is the horizon of the episode. The reward gives relevant information to solve the ask only in the last time step of the episode, which makes exploration hard.

Further Visualizations of learned skills. We show additional plots of the box's trajectories in the box pushing task in Fig. 8.

Hyperparameters are listed in the Table 6.

D.3 Extended 5-Link Reacher Task

Environment. In the 5-Link Reacher task, a 5-link planar robot has to reach a goal position with its tip. The reacher's initial position is straight to the right. This task is difficult to solve, as it introduces multi-modality in the behavior space. [16] avoided this multi-modality by constraining the y coordinate of the goal position to $y \geq 0$, i.e. the first two quadrants. We adopt the 5Link-Reacher task by increasing the context space to the full space, i.e. all four quadrants. We consider 5 basis functions per joint leading to a 25-dimensional **parameter space**. We consider the **sparse reward function** presented in [16] as

$$R_{\text{tot}} = \begin{cases} -\tau_t & t < T, \\ -\tau_t - 200R_{\text{goal}} - 10R_{\text{vel}} & t = T, \end{cases}$$

where

$$R_{\text{goal}} = \|\mathbf{p} - \mathbf{p}_{goal}\|_2$$

and

$$\tau_t = \sum_i^K (a_t^i)^2.$$

The sparse reward only returns the task reward in the last time step T and additionally adds a velocity penalty $R_{\text{vel}} = \sum_i^K (\dot{q}_T^i)^2$. The joint velocities are denoted as$\dot{\mathbf{q}}$. This velocity penalty avoids overshooting in the last time step.

Hyperparameters are listed in the Table 3.

D.4 Robot Mini Golf Task

Environment. In the robot mini golf task the agent needs to hit a ball while avoiding the two obstacles, such that it passes the tight goal to achieve a bonus. The **context space** consists of the ball's initial x-position $x_{ball} \in [0.25m, 0.6m]$, the XY positions of the green obstacle $x_{obs} \in [0.3, 0.6]$ and $y_{obs} \in [-0.5, -0.1]$ and the x positions of the goal $x_{ball} \in [0.25, 0.6]$. The **parameter space** is 29 dimensional resulting from the 4 basis functions per joint and an additional duration parameter which allows the robot to learn the duration of the trajectory. The robot starts always at the same position. The **reward** function consists of three stages:

$$R_{task} = \begin{cases} -0.0005 \cdot \tau_t & \text{if cond. 1,} \\ 0.2 - 0.2 \tanh\left(\min ||\mathbf{p}_r - \mathbf{p}_b||\right) & \text{if cond. 2,} \\ 2 - 2\tanh\left(\min ||\mathbf{p}_b - \mathbf{p}_g||\right) & \\ - \tanh\left(||\mathbf{p}_{b,y} - \mathbf{p}_{thresh,y}||\right) & \text{if cond. 3,} \\ 6 & \text{if cond. 3,} \end{cases}$$

where the individual conditions are

- cond. 1: the end of the episode is not reached,
- cond. 2: the end of the episode is reached and the robot did not hit the ball,
- cond. 3: the end of the episode is reached and the robot has hit the ball, but the ball didn't pass the goal
- cond. 4: the end of the episode is reached, robot has hit the ball and the ball has passed the goal for at least 0.75m

The episode ends when the maximum horizon length $T = 100$ is achieved. We again make use of the advantage that we obtain the whole desired trajectory ahead of the environment interaction, such that we can collect some samples without physical simulation. The reward function based on this desired trajectory is defined as

$$r_{traj} = \sum_{(i,j)} |\tau_{ij}^d| - |q_j^b|, \quad (i, j) \in \{(i, j) \mid |\tau_{ij}^d| > |q_j^b|\}$$

where τ^d is the desired trajectory, i is the time index, j is the joint index, q^b is the joint position upper bound. The desired trajectory is considered as invalid if $r_{traj} < 0$, an invalid trajectory will not be executed on the robot. Additionally, we provide a punishment, if the agent samples invalid duration times

$$r_{dur} = -3\left(\max(0, t_d - td, max) + \max(0, t_{d,mint} - t_d)\right),$$

where $t_{d,max} = 1.7s, t_{d,min} = 0.45s$ and t_d is the duration in seconds chosen by the agent. The overall reward is defined as:

$$r = \begin{cases} r_{traj}, -20(r_{traj} + r_{dur}) - 5 & \text{if invalid duration,} \\ & \text{or trajectory} \\ r_{task}, & \text{otherwise.} \end{cases}$$

Hyperparameters are listed in the Table 7.

E Additional Evaluations

We provide additional diverse skills to the Box Pushing Obstacle task in Fig. 8. In Fig. 10 we provide additional diverse strikes to fixed ball's desired landing positions on the **TT-H** task.

Furthermore, we analyze Di-SkilL's performance on the hopper jump task in more detail. In Fig. 9a we observe that the mean return is on par with BBRL, similar to the achieved goal distance in Fig. 9c. However, there is a small gap in the max height, where BBRL jumps slightly higher (see Fig. 9b. Given that the mean return is on par, one would expect that the maximum jump height is on par as well. However, Di-SkilL optimizes the remaining terms in the objective of the hopper jump task such as the healthy reward (see Appendix D), which explains this gap.

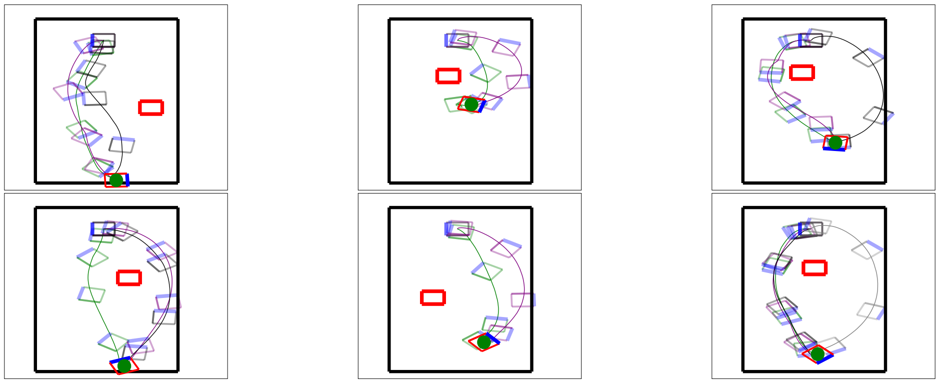
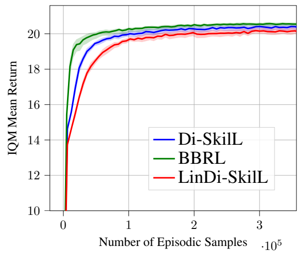
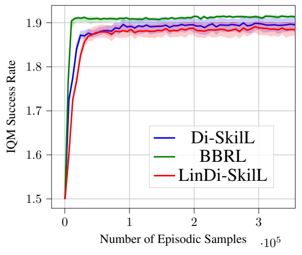


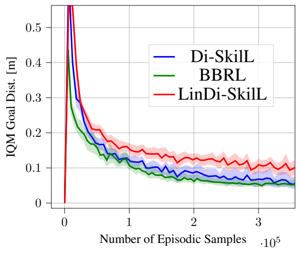
Figure 8: **Additional Diverse Skills for the Box Push Obstacle Task learned by Di-SkilL.** We fix the contexts and sample experts which we subsequently execute. This leads to diverse behaviors in the motion primitive parameter space$\boldsymbol{\theta}$ which leads to different trajectories of the pushed box on the table.



(a) IQM Mean Return **HJ**



(b) IQM Max. Height Jump **HJ**



(c) IQM Goal Distance **HJ**

Figure 9: Additional Analysis of the Hopper Jump (HJ) task.

F Hyperparameters

We list the hyperparameters for all algorithms on all environments in the following tables.

add component every iteration	1000
fine tune all components every iteration	50
number component adds	1
number initial components	1
number total components	20
number traj. samples per component per iteration	200
α	0.0001
β	0.5
expert KL-bound	0.01
context KL-bound	0.01

Table 1: Hyperparameters for SVSL on TT

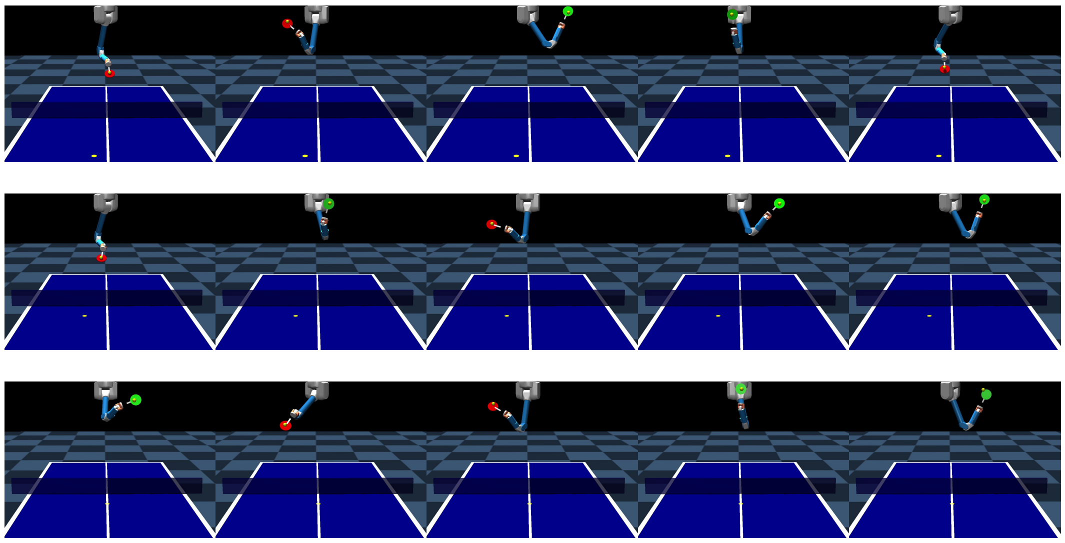


Figure 10: Di-SkilL's **Diverse Skills** for the **TT-H** task. We fixed the ball's desired landing position and varied the serving landing position and the ball's initial velocity. Di-SkilL can return the ball in different striking types. Note that each row represents a different desired ball landing position.

	Di-SkilL	**BBRL**
critic activation	tanh	tanh
hidden sizes critic	[8,8]	[32, 32]
initialization	orthogonal	orthogonal
lr critic	0.0003	0.0003
optimizer critic	adam	adam
ciritc epochs	100	100
activation context distribution	tanh	–
epochs context distribution	100	–
hidden sizes context distr	[16,16]	–
initialization	orthogonal	–
lr context distribution	0.0001	–
optimizer context distr	adam	–
batch size per component	50	209
number samples from environment distribution	5000	–
number samples per component	50	209
normalize advantages	True	True
expert activateion	tanh	tanh
epochs	100	100
hidden sizes expert	[64]	[32]
lr policy	0.0003	0.0003
covariance type	full	full
alpha	0.001	–
beta	4	–
number components	5	–
covariance bound	0.005	0.001
mean bound	0.05	0.05
projection type	KL	KL
trust region coefficient	100	25

Table 2: Hyperparameters for Di-SkilL and BBRL on TT.

	Di-SkilL	**BBRL**	**LinDi-SkilL**	**PPO**
critic activation	tanh	tanh	tanh	tanh
hidden sizes critic	[32,32]	[32, 32]	[32, 32]	[32, 32]
initialization	orthogonal	orthogonal	orthogonal	orthogonal
lr critic	0.0003	0.0003	0.0003	0.0003
optimizer critic	adam	adam	adam	adam
ciritc epochs	100	100	100	10
activation context distribution	tanh	–	tanh	–
epochs context distribution	100	–	100	–
hidden sizes context distr	[16,16]	–	[16, 16]	–
initialization	orthogonal	–	orthogonal	–
lr context distribution	0.0001	–	0.0001	–
optimizer context distr	adam	–	adam	–
batch size per component	25	240	25	512 (32 minibatches)
number samples from environment distribution	5000	–	5000	–
number samples per component	25	240	25	16384
normalize advantages	True	True	True	True
expert activateion	tanh	tanh	–	tanh
epochs	100	100	100	10
hidden sizes expert	[32,32]	[64,64]	–	[32, 32]
lr policy	0.0003	0.0003	0.0003	0.0003
covariance type	full	full	full	diagonal
alpha	0.01	–	0.01	–
beta	8	–	8	–
number components	10	–	10	–
covariance bound	0.001	0.005	0.0005	–
mean bound	0.05	0.05	0.05	–
projection type	KL	KL	KL	–
trust region coefficient	100	25	100	–
discount factor	1	1	1	1

Table 3: Hyperparameters for Di-SkilL, BBRL, LinDi-SkilL, and PPO on 5LR. We used all code-level optimization [62] needed for PPO. The implementation is based on the source code from [25].

	Di-SkilL	**BBRL**	**LinDi-SkilL**
critic activation	tanh	tanh	tanh
hidden sizes critic	[8,8]	[32, 32]	[8,8]
initialization	orthogonal	orthogonal	orthogonal
lr critic	0.0003	0.0003	0.0003
optimizer critic	adam	adam	adam
ciritc epochs	100	100	100
activation context distribution	tanh	–	tanh
epochs context distribution	100	–	100
hidden sizes context distr	[16,16]	–	[16, 16
initialization	orthogonal	–	orthogonal
lr context distribution	0.0001	–	0.0001
optimizer context distr	adam	–	adam
batch size per component	50	209	50
number samples from environment distribution	5000	–	5000
number samples per component	50	209	50
normalize advantages	True	True	True
expert activateion	tanh	tanh	–
epochs	100	100	
hidden sizes expert	[128]	[32,32]	–
lr policy	0.0003	0.0003	0.0003
covariance type	full	full	full
alpha	0.001	–	0.001
beta	0.5	–	0.5
number components	10	–	10
covariance bound	0.005	0.0005	0.001
mean bound	0.05	0.05	0.05
projection type	KL	KL	KL
trust region coefficient	100	25	100

Table 4: Hyperparameters for Di-SkilL, BBRL, and LinDi-SkilL for the Hard Table Tennis Task (TT-H).

	Di-SkilL	**BBRL**	**LinDi-SkilL**
critic activation	tanh	tanh	tanh
hidden sizes critic	[64,64]	[64, 64]	[64,64]
initialization	orthogonal	orthogonal	orthogonal
lr critic	0.0001	0.0001	0.0001
optimizer critic	adam	adam	adam
ciritc epochs	100	100	100
activation context distribution	tanh	–	tanh
epochs context distribution	100	–	100
hidden sizes context distr	[16,16]	–	[16, 16]
initialization	orthogonal	–	orthogonal
lr context distribution	0.0001	–	0.0001
optimizer context distr	adam	–	adam
batch size per component	80	200	80
number samples from environment distribution	1000	–	1000
number samples per component	80	200	80
normalize advantages	True	True	True
expert activateion	tanh	tanh	–
epochs	100	100	100
hidden sizes expert	[32, 32]	[32,32]	–
lr policy	0.0003	0.0003	0.0003
covariance type	full	full	full
alpha	0.01	–	0.01
beta	8	–	8
number components	3	–	3
covariance bound	0.005	0.05	0.005
mean bound	0.05	0.1	0.05
projection type	KL	KL	KL
trust region coefficient	100	25	100

Table 5: Hyperparameters for Di-SkilL, BBRL, and LinDi-SkilL for the Hopper Jump Task (HJ).

	Di-SkilL	**BBRL**	**LinDi-SkilL**	**PPO**
critic activation	tanh	tanh	tanh	tanh
hidden sizes critic	[32,32]	[32, 32]	[32, 32]	[256, 256]
initialization	orthogonal	orthogonal	orthogonal	orthogonal
lr critic	0.0003	0.0003	0.0003	0.0001
optimizer critic	adam	adam	adam	adam
ciritc epochs	100	100	100	10
activation context distribution	tanh	–	tanh	–
epochs context distribution	100	–	100	–
hidden sizes context distr	[16,16]	–	[16, 16]	–
initialization	orthogonal	–	orthogonal	–
lr context distribution	0.0001	–	0.0001	–
optimizer context distr	adam	–	adam	–
batch size per component	50	500	50	410 (40 minibatches)
number samples from environment distribution	5000	–	5000	–
number samples per component	50	500	50	16384
normalize advantages	True	True	True	True
expert activateion	tanh	tanh	–	tanh
epochs	100	100	100	10
hidden sizes expert	[64,64]	[64,64	–	[256, 256]
lr policy	0.0003	0.0003	0.0003	0.0001
covariance type	full	full	full	diagonal
alpha	0.01	–	0.0001	–
beta	64	–	64	–
number components	10	–	10	–
covariance bound	0.005	0.0005	0.001	–
mean bound	0.05	0.05	0.05	–
projection type	KL	KL	KL	–
trust region coefficient	100	25	100	–
discount factor	1	1	1	1

Table 6: Hyperparameters for Di-SkilL, BBRL, LinDi-SkilL, and PPO for Box Pushing Obstacle task (BPO). We used all code-level optimization [62] needed for PPO. The implementation is based on the source code from [25].

	Di-SkilL	**BBRL**	**LinDi-SkilL**
critic activation	tanh	tanh	tanh
hidden sizes critic	[32,32]	[32, 32]	[32, 32]
initialization	orthogonal	orthogonal	orthogonal
lr critic	0.0003	0.0003	0.0003
optimizer critic	adam	adam	adam
ciritc epochs	100	100	100
activation context distribution	tanh	–	tanh
epochs context distribution	100	–	100
hidden sizes context distr	[16,16]	–	[16, 16]
initialization	orthogonal	–	orthogonal
lr context distribution	0.0001	–	0.0001
optimizer context distr	adam	–	adam
batch size per component	50	500	50
number samples from environment distribution	5000	–	5000
number samples per component	50	500	50
normalize advantages	True	True	True
expert activateion	tanh	tanh	–
epochs	100	100	100
hidden sizes expert	[64,64]	[128,128]	–
lr policy	0.0003	0.0003	0.0003
covariance type	full	full	full
alpha	0.0001	–	0.0001
beta	1	–	1
number components	10	–	10
covariance bound	0.005	0.001	0.001
mean bound	0.05	0.05	0.01
projection type	KL	KL	KL
trust region coefficient	100	25	100

Table 7: Hyperparameters for Di-SkilL, BBRL, and LinDi-SkilL for the mini golf task.