A Derivations

In the following we derive our objectives from Section 3.

A.1 Maximum Entropy Skill Learning with Curriculum

We start our derivation from the KL-regularized maximum entropy objective

$$\max_{\pi(\mathbf{c},\boldsymbol{\theta})} \mathbb{E}_{\pi(\mathbf{c})}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c})}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta})\right] + \alpha \mathrm{H}\left[\pi(\boldsymbol{\theta}|\mathbf{c})\right]\right] - \beta \mathrm{KL}\left(\pi(\mathbf{c}) \parallel p(\mathbf{c})\right) \tag{8}$$

We use following identities

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

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

$$\pi(\mathbf{c}) = \sum_o \pi(\mathbf{c}|o)\pi(o)$$

$$\log \pi(\mathbf{c}) = \log \frac{\pi(\mathbf{c}|o)\pi(o)}{\pi(o|\mathbf{c})}$$

and insert them into our objective in Equation (8), which leads to

$$\begin{aligned} L = &\sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) - \alpha \log \pi(\boldsymbol{\theta}|\mathbf{c},o) - \alpha \log \pi(o|\mathbf{c}) + \alpha \log \pi(o|\mathbf{c},\boldsymbol{\theta})\right]\right] \\ &+ \sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[-\beta \log \pi(\mathbf{c}|o) + \beta \log \pi(o|\mathbf{c}) - \beta \log \pi(o)\right]. \end{aligned} \tag{9}$$

Note that we have dropped the $\log p(\mathbf{c})$ term since we assume $p(\mathbf{c})$ to be uniformly distributed in a given interval.
By further rearranging the terms we can reformulate this objective as

$$\begin{aligned} L = &\sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) - \alpha \log \pi(\boldsymbol{\theta}|\mathbf{c},o) + \alpha \log \pi(o|\mathbf{c},\boldsymbol{\theta})\right]\right] \\ &+ \sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[-\beta \log \pi(\mathbf{c}|o) + (\beta-\alpha) \log \pi(o|\mathbf{c}) - \beta \log \pi(o)\right]. \end{aligned} \tag{10}$$

As stated in the paper, we can not optimize this objective. Therefore we introduce the auxiliary distributions $\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})$ and $\tilde{\pi}(o|\mathbf{c})$ as

$$\begin{aligned} L = &\sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) - \alpha \log \pi(\boldsymbol{\theta}|\mathbf{c},o) + \alpha \log \pi(o|\mathbf{c},\boldsymbol{\theta}) + \alpha \log \tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right.\right. \\ &\left.\left. -\alpha \log \tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right]\right] + \sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[-\beta \log \pi(\mathbf{c}|o) - \beta \log \pi(o) + (\beta-\alpha) \log \pi(o|\mathbf{c})\right. \\ &\left. +(\beta-\alpha) \log \tilde{\pi}(o|\mathbf{c}) - (\beta-\alpha)\tilde{\pi}(o|\mathbf{c})\right]. \end{aligned}$$

We can rearrange the terms further to

$$\begin{aligned} L = &\sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha \log \tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right] + (\beta-\alpha) \log \tilde{\pi}(o|\mathbf{c})\right] \\ &+ \alpha\mathbb{E}_{\pi(o),\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathcal{H}\left(\pi(\boldsymbol{\theta}|\mathbf{c},o)\right)\right] + \beta\mathbb{E}_{\pi(o)}\left[\mathcal{H}\left(\pi(\mathbf{c}|o)\right)\right] + \beta\mathcal{H}\left(\pi(o)\right) \\ &+ \alpha\mathbb{E}_{\pi(\mathbf{c}),\pi(\boldsymbol{\theta}|\mathbf{c})}\left[\mathrm{KL}\left(\pi(o|\mathbf{c},\boldsymbol{\theta})||\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right)\right] + (\beta-\alpha)\mathbb{E}_{\pi(\mathbf{c})}\left[\mathrm{KL}\left(\pi(o|\mathbf{c})||\tilde{\pi}(o|\mathbf{c})\right)\right]. \end{aligned} \tag{11}$$

A.2 Lower-Bound Decomposition for Component Distributions

By observing that not all terms depend on $\pi(\boldsymbol{\theta}|\mathbf{c},o)$, we extract only the important ones from the objective in Equation (11) as

$$\begin{aligned} \tilde{L}_c = &\sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[\mathbb{E}_{\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathrm{R}(\mathbf{c},\boldsymbol{\theta}) + \alpha \log \tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right]\right] + \alpha\mathbb{E}_{\pi(o),\pi(\boldsymbol{\theta}|\mathbf{c},o)}\left[\mathcal{H}\left(\pi(\boldsymbol{\theta}|\mathbf{c},o)\right)\right] \\ &+ \alpha\mathbb{E}_{\pi(\mathbf{c}),\pi(\boldsymbol{\theta}|\mathbf{c})}\left[\mathrm{KL}\left(\pi(o|\mathbf{c},\boldsymbol{\theta})||\tilde{\pi}(o|\mathbf{c},\boldsymbol{\theta})\right)\right]. \end{aligned}$$

Since the KL is always positive, we can write the lower bound to this objective as

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

such that $\tilde{L}_c \geq L_c$ always holds. After maximizing L_c w.r.t. $\pi(\boldsymbol{\theta}|\mathbf{c},o)$ we tighten the lower bound by updating the responsibilities as described in the paper.

A.3 Lower Bound Decomposition for Component-wise Context Distributions

By neglecting all terms which do not depend on $\pi(\mathbf{c}|o)$ in objective (11), we can write the objective for optimizing w.r.t. $\pi(\mathbf{c}|o)$ as

$$\begin{aligned}\tilde{L}_k = &\sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[L_c(o,\mathbf{c}) + (\beta-\alpha)\log\tilde{\pi}(o|\mathbf{c})\right] + \beta\mathbb{E}_{\pi(o)}\left[\mathcal{H}\left(\pi(\mathbf{c}|o)\right)\right] \\ &+ (\beta-\alpha)\mathbb{E}_{\pi(\mathbf{c})}\left[\mathrm{KL}\left(\pi(o|\mathbf{c})||\tilde{\pi}(o|\mathbf{c})\right)\right],\end{aligned}$$

where

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

Again we can observe that the KL term is always positive and we can write the lower bound to $\tilde{L}_k$ as

$$L_{c,k} = \sum_o \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[L_c(o,\mathbf{s}) + (\beta-\alpha)\log\tilde{\pi}(o|\mathbf{c})\right] + \beta\mathbb{E}_{\pi(o)}\left[\mathcal{H}\left(\pi(\mathbf{c}|o)\right)\right],$$

such that $\tilde{L_k} \geq L_{c,k}$ holds. By setting the auxillary distributions to the responsibilities of the updated model, we tighten the bound.

A.4 Lower Bound Decomposition for Prior Weights

Finally we can write down the objective for updating the prior weights $\pi(o)$ as

$$L_p = \sum_o L_{c,k}(o) + \beta\mathcal{H}\left(\pi(o)\right), \tag{12}$$

where

$$L_{c,k}(o) = \pi(o)\mathbb{E}_{\pi(\mathbf{c}|o)}\left[L_c(o,\mathbf{s}) + (\beta-\alpha)\log\tilde{\pi}(o|\mathbf{c})\right] + \beta\mathrm{H}\left(\pi(\mathbf{c}|o)\right).$$

Some context regions naturally lead to low reward due to for example high action regularizations. Still, these context regions should be discovered, even if the auxiliary rewards $\tilde{\pi}(o|\mathbf{c})$ are not sufficient. For this purpose, we calculate a value function, which reveals the mean reward in a context. If solely one component covers this context region, it will get a high value for updating its weight $\pi(o)$, although the task reward might be bad compared to other components in other regions. We use importance sampling, to calculate the Value function

$$V(\mathbf{c}) = \sum_o \frac{\pi(o|\mathbf{c})}{\hat{\pi}(o|\mathbf{c})}\int_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta}|\mathbf{c},o)\mathrm{R}(\mathbf{c},\boldsymbol{\theta})d\boldsymbol{\theta}, \tag{13}$$

where $\hat{\pi}(o|\mathbf{c})$ are the gating probabilities before starting to update the weights $\pi(o)$ at the end of our algorithm (see Algorithm 1). We completely resample the small replay buffer for all components before starting to update $\pi(o)$. We use the Nadaraya Watson predictor to get $V(\mathbf{c})$. Thus, the update rule for $\pi(o)$ is given by

$$L_p = \sum_o L_{c,k}(o) - V(\mathbf{c}) + \beta\mathcal{H}\left(\pi(o)\right). \tag{14}$$

Algorithm 1 Versatile Skill Learning

Input: $\alpha,\ \beta,\ \beta_w$, N, K, H
Output: $\pi(\boldsymbol{\theta}|\mathbf{c})$

```
1: for k = 1 to N do
2:    π(θ|c, o), π(c|o) ← randomly_add_component()
3:    π(o) = 1/k, ∀o
4:    for i = 1 to K do
5:       if i% H is 0 then
6:          update_all_components(α) to obj. (5) each
7:          update_all_context_distributions(α, β) to obj. (6) each
8:       else
9:          π(θ|c, o) ← update_component_k(α) to obj. (5)
10:         π(c|o) ← update_context_distribution_k(α, β) to obj (6)
11:      end if
12:   end for
13:   if component_k in local optimum then
14:      delete_component_k()
15:   end if
16: end for
17: π(o) ← update_prior_weights(β_w) to obj. (14)
18: delete_redundant_components()
```

B Algorithmic Details

We describe algorithmic details in the algorithm box 1. We start with one component, which is randomly added and incrementally add components after training the lastly added component for K iterations. We update the component and context distributions individually. Note that there is no strict order on updating the components or context distributions first.

Every H iterations we fine-tune all components, if more than one component is available. This allows the already trained components to adjust for the newly added one. After finishing training the lastly added component k, we check if it converged to a local optimum. There are different ways on checking that. For example by comparing the entropy, or achieved task reward to already trained components. This step is not absolutely necessary since for the upcoming fine tune steps, the component might improve. For the Beer Pong and Table Tennis task, we have disabled the deletion step in line 13.

As last step we update the prior weights $\pi(o)$ and delete all components, which are below a threshold value (e.g. 10^{-5}) for $\pi(o)$.

Note that in some experiments, it might happen that highly versatile skills result in worse rewards, e.g. through action regularization, although the task is solved successfully. In order to avoid deleting these diverse skills from the skill library at the end of the optimization, a higher β value, β_w for the entropy of $\pi(o)$ in Equation 14 can be used. This higher β_w value will encourage to put weights on skills which are not achieving highest reward. Although those skills still will have lower weight compared to other components, they are kept in the skill set and might be useful. For example when the model should be adapted to an environmental change. We made use of a different β_w value only once in the Beer Pong task (see Experimental Details C).

Note that we have used a variant of MORE [17] for optimizing obj. 6, as described in [20] and a variant of Contextual MORE [18] for optimizing obj. 5 as described in [21].

Also note, that for updating each component we use a small replay buffer. The newly taken samples replace the oldest samples in the buffer. For the updates we simply use all samples in the buffer for each component.

C Experimental Details

We describe the different hyper parameters and environment specifications used in the experiments part here. Please note: We included a context punishment in all reward functions for all experiments for contexts which were sampled out of the context range defined by the environment. This encourages the context distributions $\pi(\mathbf{c}|o)$ to stay in the valid context region as defined from the environment. We report the reward functions and the environment specifications in the sub section for each experiment.

As for the hyper parameters for all experiments for our Algorithm, we provide a summary in Table 1. Following the notation from Algorithm 1, β_w is the entropy coefficient for $\pi(o)$ in Objective 14, N is the number of incrementally added components, K is the number of iterations one component is trained and H describes the number that every H. iteration all components are updated.

For the hyper parameters for all experiments for HiREPS and LaDiPS, we provide a summary in Table 2 and 3 respectively.

Hyper Parameter (Ours)	α	β	β_w	N	K	H
Planar Reacher (Ablation)	varies (see section 4.1)	1.0	1.0	60	350	50
Beer Pong Task	0.001	0.5	2.5	70	750	50
Table Tennis Task	0.00001	0.2	0.2	50	800	50

Table 1: Hyperparameters of our algorithm for all environments.

Hyper Parameter (HiREPS)	ϵ	κ	N
Planar Reacher (Ablation)	0.5	0.99	60
Beer Pong Task	0.3	0.9	70
Table Tennis Task	0.3	0.9	50

Table 2: Hyperparameters of HiREPS for all environments.

Hyper Parameter (LaDiPS)	ϵ	ϵ_{gating}	α	N
Beer Pong Task	0.1	0.05	$\log 30$	70
Table Tennis Task	0.01	0.02	$\log 30$	50

Table 3: Hyperparameters of LaDiPS.

C.1 Ablation Studies

For the Planar Reaching task we consider a two dimensional context space. Note that we have not used any Movement Primitive representation for the policy. Instead, the sampled θ from our search distributions directly represent the angles of each joint of the reacher task.

We consider a two dimensional context space x, y, where $x_{min} = 4.5,\ x_{max} = 7$ and $y_{min} = -6,\ y_{max} = 6$.

The reward function is given by:

$$R(\boldsymbol{\theta}, \mathbf{c}) = -||\boldsymbol{\theta}||_2^2 - 2 \cdot ||\mathbf{f}(\boldsymbol{\theta}) - \mathbf{c}||_2^2 - l_c(\mathbf{c}) - l_o(\boldsymbol{\theta}),$$

where

$$l_c(\mathbf{c}) = \begin{cases} 10, & c \notin C \\ 0, & else \end{cases}, \quad l_o(\mathbf{c}) = \begin{cases} 3, & b_{1:10} \in O \\ 0, & else \end{cases},$$

where l_c are the costs if a context $\mathbf{c}$ is not within the valid context range C and l_o are the costs, if one of the ten links b_i has a collision with the obstacles of the environment. $\mathbf{f}(\boldsymbol{\theta})$ are the forward kinematics of the planar reacher. Note that the angles $\boldsymbol{\theta}$ are always normalized into a range $[-\pi, \pi]$.

C.2 Beer Pong

For the Beer Pong task we consider a two dimensional context space x, y positions of the cup on the table, where $x_{min} = -0.32,\ x_{max} = 0.32$ and $y_{min} = -2.2,\ y_{max} = -1.2$. We use ProMPs as policy parameterizations. We also learn the length of the trajectory, which might lead to invalid, i. e. negative trajectory length. We use a punishment term, if the trajectories are out of a reasonable

range and do not execute this sample on the environment. We do the same for the context samples. Furthermore, we do not execute trajectories, which violate the joint constraints of the robot. Since these samples are not executed on the environment, they are not counted as "taken samples". For the case that those restrictions are not violated, the reward function is given as

$$R(\boldsymbol{\theta}, \mathbf{c}) = \begin{cases} -4 - min(||p_{c,top} - p_{b,1:T}||_2^2) - 0.5||p_{c,bottom} - p_{b,T}||_2^2 - 10^{-4}\tau, & \text{if cond. 1} \\ -2 - min(||p_{c,top} - p_{b,1:T}||_2^2) - 0.5||p_{c,bottom} - p_{b,T}||_2^2 - 10^{-4}\tau, & \text{if cond. 2} \\ ||p_{c,bottom} - p_{b,T}||_2^2 + 1.5 \cdot 10^{-4}\tau, & \text{if cond. 3} \end{cases},$$

where $p_{c,top}$ is the position of the top edge of the cup, $p_{c,bottom}$ is the ground position of the cup, $p_{b,t}$ is the position of the ball at time point t and τ is the squared mean torque over all joints during one rollout. The different conditions are:

- cond. 1: The ball is not in the cup and had no table contact
- cond. 2: The ball is not in the cup and had table contact
- cond. 3: The ball is in the cup.

For the case that context samples were sampled out of the range we provide a high negative reward

$$R_c(\boldsymbol{\theta}, \mathbf{c}) = -30 - d_c^2,$$

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

For the case that the duration of the trajectory was sampled out of the range we provide a negative reward as

$$R_\theta(\mathbf{c}, \boldsymbol{\theta}) = \begin{cases} -30 - 10 \cdot (l - 0.1)^2, & \text{if } l < 0.1 \\ -30 - 10 \cdot (l - 1.3)^2, & \text{if } l > 1.3 \end{cases},$$

where l is the duration of the Trajectory in seconds. If the joint constraint limits are violated by the ProMP's trajectory, we give a punishment of -30.

C.2.1 On Versatility of our Solutions on the Beer Pong Task

In this section we want to compare the learned solutions of LaDiPS [9] and our method. As LaDiPS uses experience-sharing between components, it outperforms our method in terms of sample efficiency, but does not find as qualitative solutions as our method, which is reflected in the end-rewards (see Fig. 2a), where we can clearly achieve a higher value. In addition to the more qualitative solutions, we are able to learn a mixture model with higher expected entropy, which can be seen in Fig. 4). Our method is able to achieve more qualitative skills with a much higher entropy compared to LaDiPS and HiREPS, indicating that the proposed solutions by our method are more versatile than the others. Please note that we have defined a fine grid of contexts and sampled for each context $\mathbf{c}$ 1000 parameter vectors $\boldsymbol{\theta}$ from the mixture models to calculate the expected entropy. We have repeated this procedure for all 20 different trials/seeds and report the mean expected entropy together with two times the standard error. In addition to the expected mixture entropy we have conducted a qualitative comparison, where we have picked the first model for both, our method and LaDiPS and have let it run for twelve different contexts. For each context we have sampled 100 times from the mixture model and executed the mean of the sampled components on the environment. Note that we only have taken those ball trajectories into account that lead to a successful solution. For LaDiPS all observed solutions for all contexts showed only one mode, in which the ball bounced once on the table and then landed in the cup. For our method the same solution could be seen for six contexts. For three contexts, our method threw the ball directly into the cup, whereas for another three contexts three different ball trajectories could be observed: i) direct throw, ii) bouncing ones on the table ii) bouncing once on the table and once on the wall.

C.3 Table Tennis

For the Table Tennis task we consider a four dimensional context space $[\mathbf{b^{des}}, \mathbf{b^{inc}}] \in \mathbb{R}^4$. We have the desired landing position of outgoing ball $\mathbf{b^{des}} = [x^{des}, y^{des}] \in \mathbb{R}^2$ and the initial positions of the incoming ball $\mathbf{b^{inc}} = [x^{inc}, y^{inc}] \in \mathbb{R}^2$ where $x^{des} \in [-1.2, -0.2]$, $y^{des} \in$

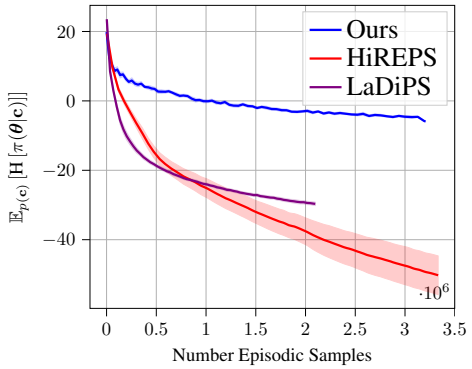


Figure 4: Expected Mixture Entropies.

$[-0.6,\ 0.6]$, $x^{inc} \in [-1.2,\ -0.2]$, $y^{inc} \in [-0.65,\ 0.65]$. We fix the initial ball velocity and z^{inc} position so we can transform the initial ball position to the incoming ball landing position on the table. We use ProMPs as policy parameterizations and learn the length of the trajectory, which might lead to invalid, i. e. negative trajectory length. We use a punishment term, if the length of trajectories are out of a reasonable range and do not execute this sample on the environment. We do the same for the context samples. Furthermore, we do not execute trajectories, which violate the joint constraints of the robot and return a punishment. Since these samples are not executed on the environment, they are not counted as "taken samples". For the case that those restrictions are not violated, the reward function is given as

$$R(\boldsymbol{\theta}, \mathbf{c}) = \begin{cases} 0.2 \cdot (1 - tanh(\min_t(||\mathbf{b_t} - \mathbf{r_t}||_2^2)), & \text{if cond. 1} \\ 2 \cdot (1 - tanh(\min_t(||\mathbf{b_t} - \mathbf{r_t}||_2^2)) \\ + (1 - tanh(\min_t(||\mathbf{b^{des}} - \mathbf{b_{t,xy}}||_2^2)), & \text{if cond. 2} \\ 2 \cdot (1 - tanh(\min_t(||\mathbf{b_t} - \mathbf{r_t}||_2^2)) \\ + 4 \cdot (1 - tanh((||\mathbf{b^{des}} - \mathbf{b^{land}}||_2^2)) + 1_{\{b_x^{land} < 0\}}, & \text{if cond. 3} \end{cases},$$

where $\mathbf{b_t} \in \mathbb{R}^3$ is the position of the ball at time point t, $\mathbf{b_{t,xy}} \in \mathbb{R}^2$ is the x, y position of the ball at time point t, $\mathbf{r_t} \in \mathbb{R}^3$ is the position of the racket at time point t, $\mathbf{b^{land}} \in \mathbb{R}^2$ is the real x, y landing position point. The different conditions are:

- cond. 1: The ball had no racket contact.
- cond. 2: The ball had racket contact but then no table contact.
- cond. 3: The ball had racket contact and then table contact.

For the case that context samples were sampled out of the range we provide a high negative reward

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

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

For the case that the duration of the trajectory was sampled out of the range we provide a negative reward as

$$R_\theta(\mathbf{c}, \boldsymbol{\theta}) = \begin{cases} -3||l - 0.1||, & \text{if } l < 0.1 \\ -3||l - 5||, & \text{if } l > 5 \end{cases},$$

where l is the duration of the Trajectory in seconds.

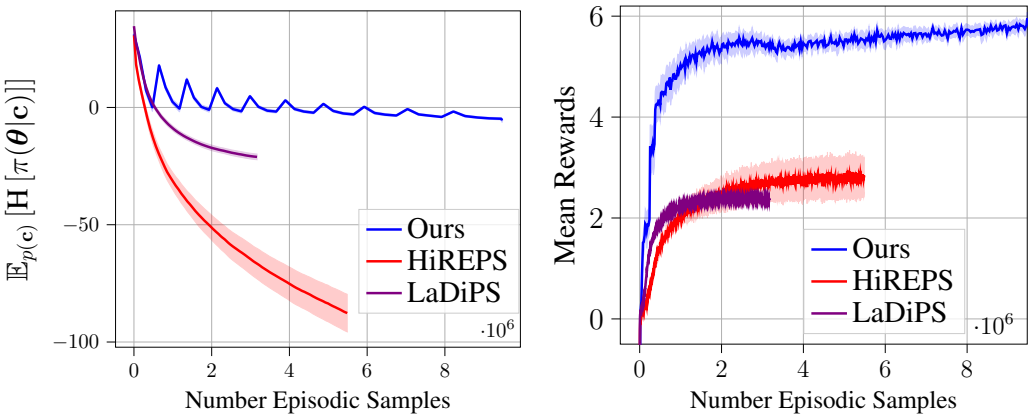
For the case that the trajectory was sampled out of the joint constraints we provide a negative reward as

$$R_c(\boldsymbol{\theta}, \mathbf{c}) = -1 \cdot \frac{1}{T} \sum_i^T d_{qpos,t},$$

where $d_{qpos,t}$ is the distance of the planned joint positions to the valid joint position region.

C.3.1 On Versatility of our Solutions on the Table Tennis Task

We compare the versatility of learned solutions of LaDiPS [9], HiREPS [8] and ours. We plot the expected mixture entropies of each algorithm in Fig. 5a. For each of the 1600 uniformly context samples we have sampled 1000 parameter vectors $\boldsymbol{\theta}$ from the mixture model to calculate the mixture entropy. We repeat this for all models resulting from 20 seeds/trials and plot the average of the entropies in Fig. 5a. Please note that the fluctuations of the entropy curve in Fig. 5a of our algorithm are due to adding components during training.



(a) (b)

Figure 5: **Expected Mixture Entropies** (left) and **performance on the table tennis task** (right). Note that the performance graph is the same as in Fig. 2b, but with the full graph of our method to train 70 components. The graph in Fig. 2b was clipped in the x-axis to maintain overview.

We are able to achieve the highest entropy while clearly outperforming both methods in terms of the achieved task reward (see Fig. 5b), which indicates that we are able to learn more qualitative skills.

C.4 A Comparison Between Episodic- and Step-Based Policy Search

The step-based and the episodic policy search setting have their own benefits and limitations. For example, while in the step-based case we usually explore the action space using uncorrelated noise, the episodic setting uses correlated distributions directly in the parameter space of the controller [3]. Yet, step-based approaches can use the data more efficiently as they decompose the episodes into single time steps. Hence, it is interesting to compare these approaches. We compare the performance of our algorithm against PPO, a well-known Deep Reinforcement Learning method [34]. We want to analyze when episodic exploration is beneficial over step-based exploration and vice-versa for different sparsity levels of the reward. For this purpose, we use the Reacher task from OpenAI gym [36]. In this task, the agent has to reach a goal position with its tip. In order to consider a more difficult task, we extend the original reacher set-up and show the considered environment in Figure 6). First, we increase the number of links from two to five and we fix the initial state position to be initialized on the horizontal to the left (see Fig. 6), instead of initializing on the horizontal to the right side as it is in the original version.

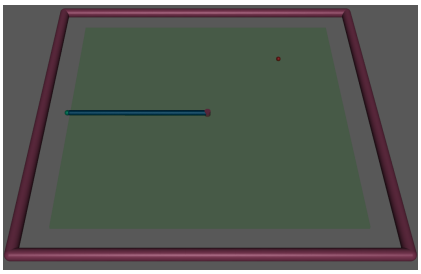


Figure 6: 5Link Reacher Set-up. The reacher's tip needs to reach the red goal position. Adapted from [36].

Furthermore, we fix the initial joint velocity to be zero. For the original 2-Link case, the goal reaching space is the circle around the agent's base, whereas in our case we consider the right half-circle around the agent's base, where the radius is the agent's total length. By initializing the agent on the

left side and only considering goal positions on the right side, we make the task harder. We compare our method to the popular PPO algorithm [34] and use the implementation from [37].

In order to examine the different policy search strategies we consider three different markovian reward functions in the same set-up. The first is given by

$$R_t(\mathbf{s}, \mathbf{a}) = -||\mathbf{t}(\mathbf{s}) - \mathbf{g}||_2 - \sum_i a_i^2, \tag{15}$$

where $\mathbf{s}$ is the current state, $\mathbf{a}$ is the current action, $t(\mathbf{s})$ is the tip's current position and $\mathbf{g}$ is the goal position. Except for $\mathbf{g}$, which is re-sampled at each episode, the values are given at the current time-step t. One episode lasts for $T = 200$ time-steps.

This reward function is also given by [36] and encodes task information, i.e. tip distance to the goal, in each time-step. In the following we only change the reward function.

The second reward function is given as

$$R_t(\mathbf{s}, \mathbf{a}) = \begin{cases} -\sum_i a_i^2, & \text{if } t < 190 \\ -||\mathbf{t}(\mathbf{s}) - \mathbf{g}||_2 - \sum_i a_i^2, & \text{if } t \geq 190 \end{cases}. \tag{16}$$

Characteristic for this reward function is that it will only return task information, i.e. tip-distance to goal, in the last 10 steps of the episode.

The third reward function is given as

$$R_t(\mathbf{s}, \mathbf{a}) = \begin{cases} -\sum_i a_i^2 - \sum_i v_i^2, & \text{if } t < 198 \\ -500 \cdot ||\mathbf{t}(\mathbf{s}) - \mathbf{g}||_2 - \sum_i a_i^2 - 1000 \cdot \sum_i v_i^2, & \text{if } t \geq 198 \end{cases}, \tag{17}$$

where $\mathbf{v}$ is the joint velocity at time-step t. This reward function is very similar to the reward function from Eq. (16). However, it only returns task information in the last two steps of the episode, which makes the task very difficult.

The reward functions in Equations (16, 17) can be motivated from the optimal control point of view. There, usually controllers with minimum energy consumption are seeked giving rise to punishments to taken control actions in the first steps as it is done by the proposed reward functions in Equations (16, 17).

For PPO we use the same observation space as described in [36], but for the Objectives (16, 17) we augment this observation space with the current time-step t.

The context $\mathbf{c}$ for our method is given through the two-dimensional goal position vector $\mathbf{g}$.

We report the average performance and two times the standard error over 20 seeds/trials for each experiment in Fig. 7). The performances for the reward function in Equation (15) are shown in Fig. 7a). Clearly, PPO outperforms our algorithm, indicating that task information in each time-step helps exploring in the raw action space to find solutions. The performances for the reward function in Equation (16) is shown in Fig. 7b). While PPO also performs well, we can observe that our algorithm can make use of the exploration in the parameter space and outperform PPO. This effect can be seen much more clearly for the reward funcion in Equation (17). Fig. 7c) shows that PPO has clear problems to solve this task. For the settings given by the reward functions in Equations (16, 17) exploration in the parameter space as done by episodic policy search methods lead to better performance, whereas exploration in the raw action space leads to converging to a local optimum.

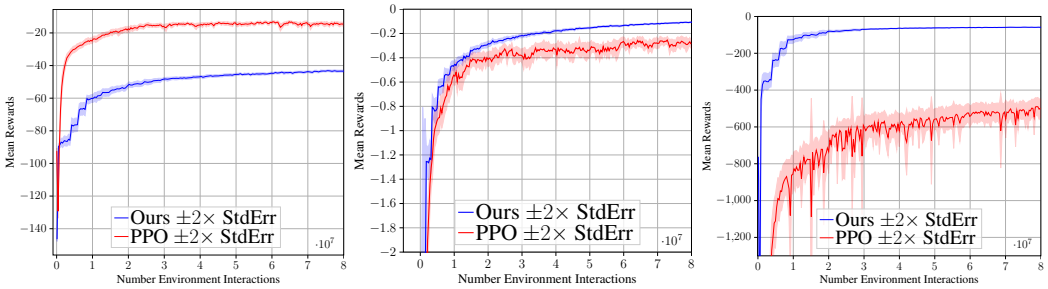
C.4.1 Hyperparameters

The hyperparameters for our algorithm are given in the following Table. Please note that we used punishments,

Hyper Parameter (Ours)	α	β	β_w	N	K	H
Reacher Reward (15)	$1e^{-6}$	15	15	40	250	50
Reacher Reward (16)	0.01	1	2.5	40	250	50
Reacher Reward (17)	1	25	25	40	800	50

Table 4: Hyperparameters of our algorithm for the reacher environment.

if contexts were sampled out of the context range (which is the right half circle around the base with radius 0.5), we provided a quadratic punishment to the distance of the sample to the valid context



(a) Performances for Reward (15) (b) Performance for Reward (16) (c) Performance for Reward (17)

Figure 7: **A Comparison of Episodic Policy Search to Step-based Policy Search.** We compare our episodic policy search method to the popular step-based approach PPO. We analyze the performance on three different reward types. For (a) we consider the reward in Equation (15). Here, task-information are returned in each time-step. For(b) we consider the reward in Equation (16). Here, task-information are returned in the last 10 steps of the episode. For (c) we consider the reward in Equation(17), where only at the last 2 time-steps reward information are returned.

bound. We multiplied this distance with a factor of 1000 for the case of reward function (15, 16) and with a factor of 50000 for the case of reward function (17).

We normalized the observations and rewards for PPO and used the same hyperparameters for all three experiments, which we summarized in the following table

Hyp. Params (PPO)	α	N	GAE λ	γ	K	ϵ	network structure
Reacher Reward (15)	$1e^{-4}$	16384	0.95	0.99	64	0.2	[64, 64]
Reacher Reward (16)	$1e^{-4}$	16384	0.95	0.99	64	0.2	[64, 64]
Reacher Reward (17)	$1e^{-4}$	16384	0.95	0.99	64	0.2	[64, 64]

Table 5: Hyperparameters of PPO for the reacher environment. Note that α is the learning rate, N is the number of rollouts, γ is the discount factor, K is the batch size, ϵ is the clip value for the importance ratio. The layer structure is for both: the policy network as well as the value function network.