Appendix

A. State Entropy Estimator

To approximate state entropy, we employ the simplified version of particle-based entropy estimator (Beirlant et al., 1997; Singh et al., 2003). Specifically, let $\mathbf{s}$ be a random variable with a probability density function p whose support is a set $\mathcal{S} \subset \mathbb{R}^q$. Then its differential entropy is given as $\mathcal{H}(\mathbf{s}) = -\mathbb{E}_{\mathbf{s}\sim p(\mathbf{s})}[\log p(\mathbf{s})]$. When the distribution p is not available, this quantity can be estimated given N i.i.d realizations of $\{\mathbf{s}_i\}_{i=1}^N$ (Beirlant et al., 1997). However, since it is difficult to estimate p with high-dimensional data, particle-based k-nearest neighbors (k-NN) entropy estimator (Singh et al., 2003) can be employed:

$$\widehat{\mathcal{H}}(\mathbf{s}) = \frac{1}{N}\sum_{i=1}^{N} \log \frac{N \cdot ||\mathbf{s}_i - \mathbf{s}_i^k||_2^q \cdot \widehat{\pi}^{\frac{q}{2}}}{k \cdot \Gamma(\frac{q}{2}+1)} + C_k \tag{6}$$

$$\propto \frac{1}{N}\sum_{i=1}^{N} \log ||\mathbf{s}_i - \mathbf{s}_i^k||_2, \tag{7}$$

where $\widehat{\pi}$ is the ratio of a circle's circumference to its diameter, $\mathbf{s}_i^k$ is the k-NN of $\mathbf{s}_i$ within a set $\{\mathbf{s}_i\}_{i=1}^N$, $C_k = \log k - \Psi(k)$ a bias correction term, Ψ the digamma function, Γ the gamma function, q the dimension of $\mathbf{s}$, and the transition from (6) to (7) always holds for $q > 0$. Then, from Equation 7, we define the intrinsic reward of the current state $\mathbf{s}_t$ as follows:

$$r^{\mathtt{int}}(\mathbf{s}_t) = \log(||\mathbf{s}_t - \mathbf{s}_t^k||).$$

Hyperparameter	Value	Hyperparameter	Value
Initial temperature	0.1	Hidden units per each layer	1024 (DMControl), 256 (Meta-world)
Learning rate	0.0003 (Meta-world), 0.001 (cheetah)	Batch Size	1024 (DMControl), 512 (Meta-world)
	0.0001 (qauadruped), 0.0005 (walker)	Optimizer	Adam (Kingma & Ba, 2015)
Critic target update freq	2	Critic EMA τ	0.005
(β_1, β_2)	$(.9, .999)$	Discount γ	.99

Table 1. Hyperparameters of the SAC algorithm. Most hyperparameters values are unchanged across environments with the exception for learning rate.

Hyperparameter	Value	Hyperparameter	Value
GAE parameter λ	0.92	Hidden units per each layer	1024 (DMControl), 256 (Meta-world)
Learning rate	0.0003 (Meta-world), 0.0001 (quadruped)	Batch Size	512 (cheetah), 128 (Otherwise)
	$5e^{-5}$ (quadruped, Walker)	# of timesteprs per rollout	100 (cheetah, Walker), 500 (quadruped)
# of environments per worker	16 (quadruped, cheetah), 32 (Walker)	PPO clip range	0.2
Entropy bonus	0.0	Discount γ	.99

Table 2. Hyperparameters of the PPO algorithm. Most hyperparameters values are unchanged across environments with the exception for learning rate.

B. Experimental Details

Training details. For our method, we use the publicly released implementation repository of the SAC algorithm (`https://github.com/denisyarats/pytorch_sac`) with a full list of hyperparameters in Table 1. On the DMControl environments, we use segments of length 50 and a frequency of teacher feedback (K in Algorithm 2) of 20K timesteps, which corresponds to roughly 20 episodes. We choose the number of queries per feedback session $M = 140, 70, 40$ for the maximum budget of 1400, 700, 400 on Walker and Cheetah, and choose $M = 70, 35, 20$ for the maximum budget of 1400, 700, 400 on Quadruped. For Meta-world, we use segments of length 10 and set $M = 64, K = 2400$ for the maximum budget of 2500, 5000, and 10000 on Drawer Close, Window Open, Door Open, and Button Press and $M = 128, K = 4800$ for maximum budget of 25000, 50000 on Sweep Into and Drawer Open.

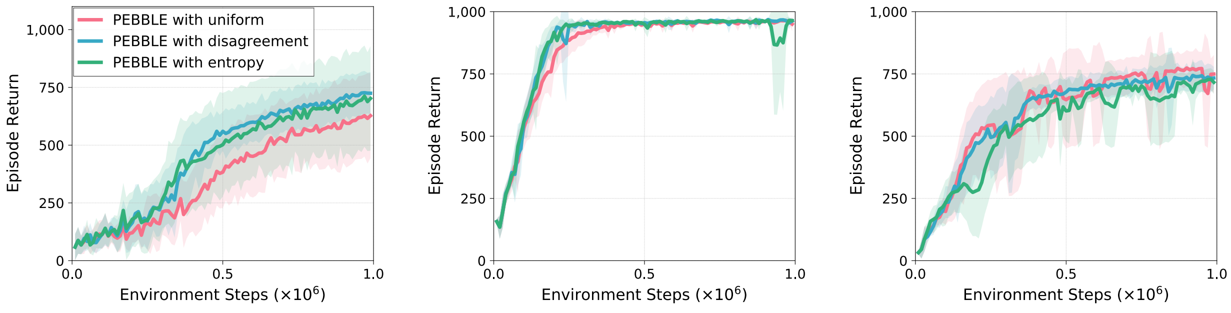
For preference PPO, we use the publicly released implementation repository of the PPO algorithm (`https://github.com/DLR-RM/stable-baselines3`) with a full list of hyperparameters in Table 2. We choose the number of queries per feedback session $M = 70, 45$ for the maximum budget of 2100, 1400 on the DMControl environments. For the reward model, we use same setups for our method. For Meta-world, we use segments of length 10 and set $M = 256, K = 2400$ for all environments and budgets of feedback.

Reward model. For the reward model, we use a three-layer neural network with 256 hidden units each, using leaky ReLUs. To improve the stability in reward learning, we use an ensemble of three reward models, and bound the output using tanh function. Each model is trained by optimizing the cross-entropy loss defined in (4) using ADAM learning rule (Kingma & Ba, 2015) with the initial learning rate of 0.0003.

Environments. We follow the standard evaluation protocol for the benchmark locomotion tasks from DMControl. The Meta-world single-task benchmark involves training and testing on a single instantiation (fixed reset and goal) of the task. To constitute a more realistic single-task manipulation setting, we randomize the reset and goal positions in all our experiments. We also use new reward function, which are nicely normalized and make the tasks stable.

C. Effects of Sampling Schemes

Figures 8 and 9 show the learning curves of PEBBLE with various sampling schemes. For Quadruped, we find that the uncertainty-based sampling schemes (using ensemble disagreement or entropy) are superior to the naive uniform sampling scheme. However, they did not lead to extra gains on relatively simple environments, like Walker and Cheetah, similar to observations from Ibarz et al. (2018). Similarly, on the robotic manipulation tasks, we find little difference in performance for simpler tasks (Drawer Close, Window Open). However, we find that the uncertainty-based sampling schemes generally fare better on the other environments.



(a) Quadruped (b) Walker (c) Cheetah

Figure 8. Learning curves of PEBBLE with 1400 pieces of feedback by varying sampling schemes. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs.

D. Examples of Selected Queries

Figure 10, 11 and 12 show some examples from the selected queries to teach the agents.

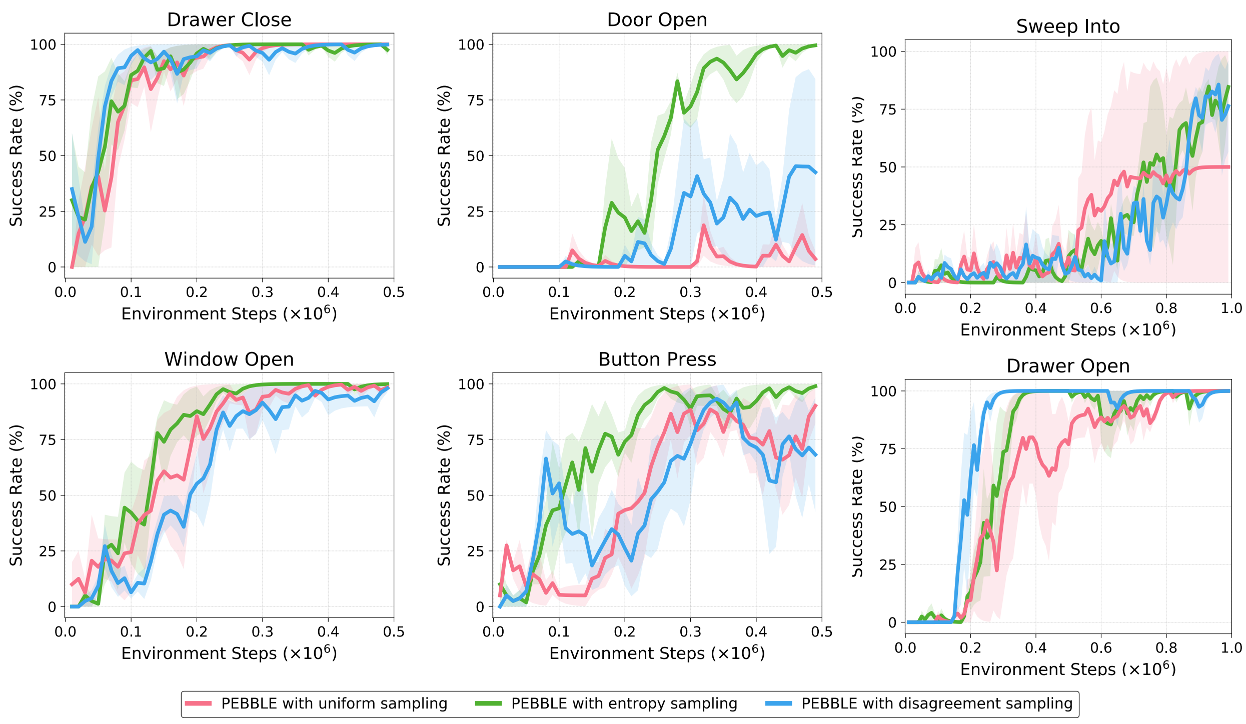
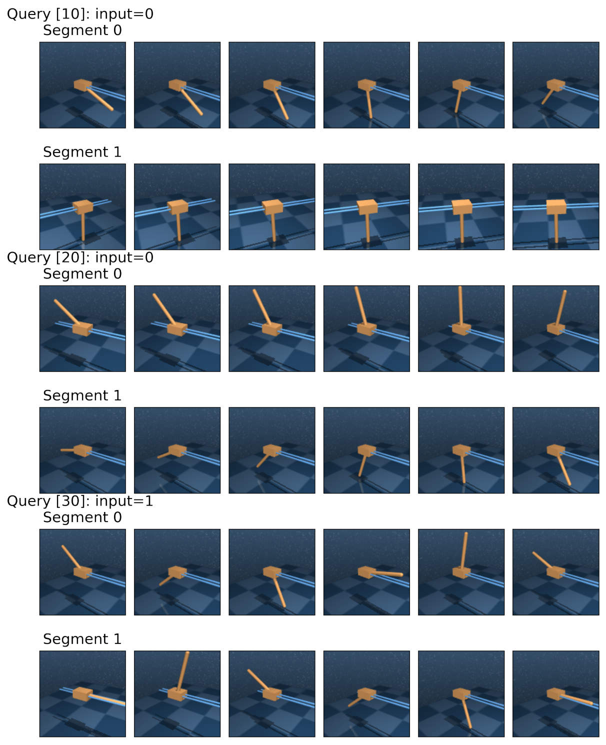
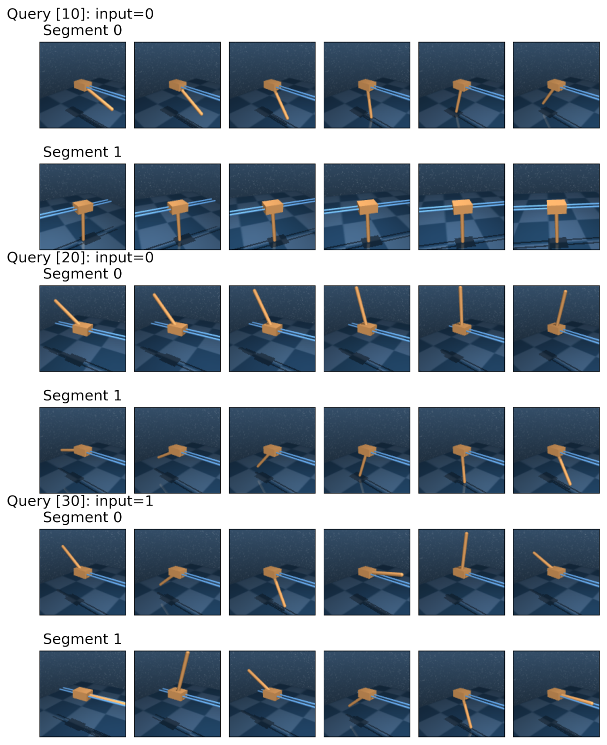


Figure 9. Learning curves of PEBBLE with various sampling schemes on the Meta-world tasks. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs.

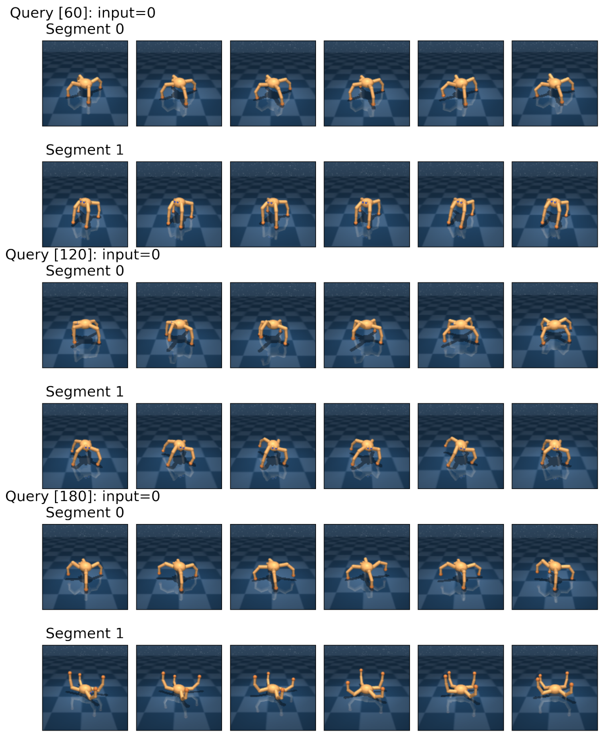


(a) Clock-wise windmill

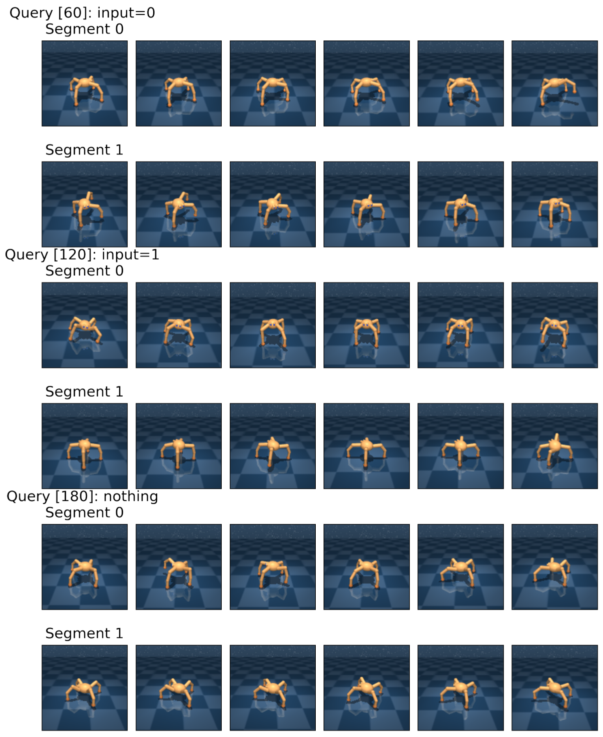


(b) Counter clock-wise windmill

Figure 10. Examples from the selected queries to teach the Cart agent.



(a) Waving left front leg



(b) Waving right front leg

Figure 11. Examples from the selected queries to teach the Quadruped agent.

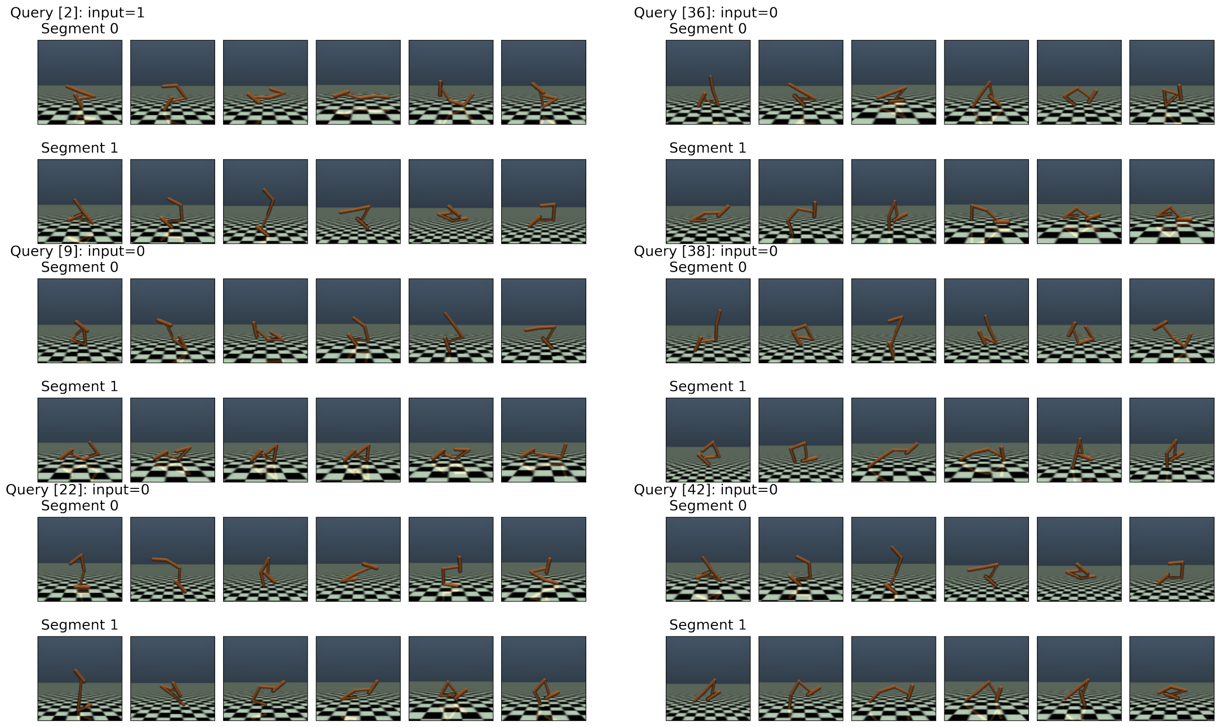


Figure 12. Examples from the selected queries to teach the Hopper agent.