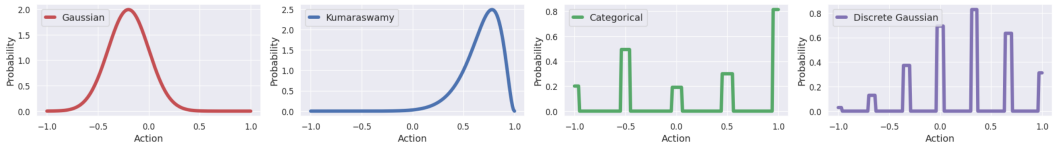

Figure 9: Probability functions of the diverse distribution types considered here. Left to right: Gaussian, Kumaraswamy, Categorical, and discrete Gaussian. We visualize a 5-bin Categorical and a 7-bin discrete Gaussian with their pseudo-probabilities. Our HMPs can combine diverse low-level controllers with different distribution types to yield compositional synergies as in Sections 5.1& 5.2.

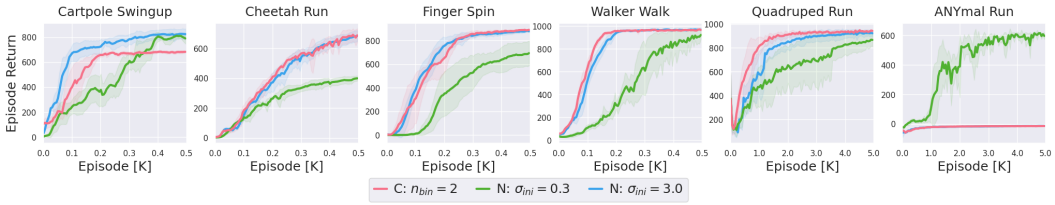

Figure 10: Our approach yields robust performance when tasked to solve continuous control with discrete actions. For instance, the coarse Categorical (C, $n_{\text{bin}} = 2$) performs competitively to the wide Gaussian (N, $\sigma_{\text{ini}} = 3.0$) on the torque controlled DeepMind Control Suite tasks, while improving on the narrow Gaussian (N, $\sigma_{\text{ini}} = 0.3$). This trend is reversed for generating stable PD-targets on ANYmal, underlining the promise of deferring task-specific controller choice to the agent.

A Distributions

The probability functions of each distribution type considered here are provided in Figure 9. The discrete distributions leverage the pseudo-densities as defined in (9) for improved backpropagation. Applications of RL to continuous control typically employ continuous distributions and the Gaussian distribution is a standard choice. Additionally, we consider the Kumaraswamy distribution as an alternative to the Beta distribution, as it is also capable of exhibiting skewness while being significantly easier to reparameterize than the Beta distribution [33].

We further investigate synergies with discrete distributions and consider the Categorical distribution as well as a discrete Gaussian. The support of both discrete distributions is a regular 1d grid with a predefined number of elements n. For the categorical, we learn the probability weights w_i for each element in its support individually. The discrete Gaussian allows for enforcing unimodality in a discrete setting. Thus, for the discrete Gaussian, we define the probability w_i for each of its support's elements x_i by

$$w_i := \frac{f(x_i)}{\sum_{j=1}^{n} f(x_j)},$$

where $f(\cdot)$ is the density of a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with the mean μ and standard deviation σ being predicted by the neural network.

B Discrete Actions in Continuous Control

HMPs work well with distribution heads that differ from the standard Gaussian assumption. Interestingly, we find that our approach performs robustly even when forced to solve continuous control tasks with discrete policy distributions. Figure 10 compares a coarse Categorical ($n_{\text{bin}} = 2$) to two Gaussian policies ($\sigma_{\text{ini}} \in \{0.3, 3.0\}$). We observe that the Categorical yields peak performance on the Walker and Quadruped tasks, while achieving high performance on the Humanoid task significantly faster than the Gaussian distributions. As expected, coarse discrete control is ill-suited for generating position targets on ANYmal. This provides another perspective on the importance of hyperparameter choices, diversity, and enabling the robot to self-select suitable controllers.

Disturbance	Control Freq.	Obs. Stuck	Obs. Drop	Obs. Delay	Obs. Noise
Parameters	Scale	Prob. ; Steps	Prob. ; Steps	Steps	Std. Dev.
Value	$\times 0.25, \times 0.5$	$(0.05; 5), (0.01; 1)$	$(0.05; 5), (0.01; 1)$	$6, 3$	$0.3, 0.1$

Table 1: Disturbances used to evaluate transfer robustness, provided as Quadruped, Humanoid.

C Disturbance Parameters

The experiments on transfer robustness of a converged policy use the disturbance parameters in Table 1. The control frequency disturbance down-samples the control by the value indicated for the Quadruped and Humanoid domains. For the observation disturbances, we selected the medium and easy disturbances from the Real-World RL Challenge framework [13] for the Quadruped and Humanoid, respectively. The Stuck sensor disturbance does not update a sensor reading for several timesteps, while the Dropped sensor disturbance zeros a sensor reading for several timesteps. Both disturbances are probabilistic, taking effect with a fixed probability and lasting for a fixed number of timesteps. The observation delay shifts all observation by a fixed number of timesteps, while the observation noise applies additive white Gaussian noise with the specified standard deviation.

D Policy Evaluation via Retrace

In order to stabilize off-policy learning of the state-action value function Q_ϕ we leverage the Retrace algorithm [37]. The optimization objective is therefore

$$\min_\phi L(\phi) = \mathbb{E}_{\tau \sim D}\left[\left(Q_t^{ret} - Q_\phi(s_t, a_t)\right)^2\right]. \tag{10}$$

The Retrace targets are computed as

$$Q_t^{ret} = Q_{\phi'}(s_t, a_t) + \sum_{j=t}^{\infty} \gamma^{j-t} \left(\prod_{k=t+1}^{j} c_k\right) [r(s_j, a_j) + \mathbb{E}_{\pi(a|s_{j+1})}[Q_{\phi'}(s_{j+1}, a)] - Q_{\phi'}(s_j, a_j)], \tag{11}$$

where $Q_{\phi'}$ refers to a target network for the state action value function, $c_k = \min\left(1, \frac{\pi(a_k|s_k)}{b(a_k|s_k)}\right)$ to the trace coefficients, and $b(a|s)$ denotes the probabilities under the behavior policy. The infinite sequence is truncated after 10 steps and we bootstrap from the target network. To increase efficiency, we consider two-step transitions by squashing consecutive timesteps before adding them to memory.

E Implementation Details

Our implementation builds on MPO as provided by the Acme library [21] and extends it to the hierarchical setting, enables application with diverse sub-policy heads (distribution type, parameterization) and implements Retrace [37] for data-efficient off-policy learning. Throughout, we follow the MPO parameters described in [1] and introduce the decoupled KL bounds for non-Gaussian distributions as $\epsilon_K = [10^{-1}, 10^{-1}]$, $\epsilon_C = [10^{-1}]$, $\epsilon_D = [10^{-1}, 10^{-1}]$. Furthermore, the high-level selector shares its torso with the low-level controllers and employs a Categorical head with logits predicted from a single fully-connected layer of width 100. Our experimental results are reported with mean and one standard deviation over 8 random seeds for the NKCD HMP comparison to RHPO and 4 random seeds for the remaining experiments. Experiments were run on 4 CPU cores in combination with a single GPU (Nvidia V100).

F Realworld disturbances

We also evaluate robustness to disturbances in the Real-World RL Challenge framework [13]. We consider down-sampling of the controls and sensor degradation as specified in Appendix C. Figure 11 indicates that diversity can improve robustness in these real-world inspired domains.

Figure 11: HMP under real-world disturbances. Our diverse mixtures can improve robustness over homogeneous baselines and aid in generalization.