Strength Through Diversity: Robust Behavior Learning via Mixture Policies

Tim Seyde*
MIT CSAIL

Wilko Schwarting
MIT CSAIL

Igor Gilitschenski
University of Toronto

Markus Wulfmeier†
DeepMind

Daniela Rus†
MIT CSAIL

Abstract

Efficiency in robot learning is highly dependent on hyperparameters. Robot morphology and task structure differ widely and finding the optimal setting typically requires sequential or parallel repetition of experiments, strongly increasing the interaction count. We propose a training method that only relies on a single trial by enabling agents to select and combine controller designs conditioned on the task. Our Hyperparameter Mixture Policies (HMPs) feature diverse sub-policies that vary in distribution types and parameterization, reducing the impact of design choices and unlocking synergies between low-level components. We demonstrate strong performance on the DeepMind Control Suite, Meta-World tasks and a simulated ANYmal robot, showing that HMPs yield robust, data-efficient learning. [2]

1 Introduction

Real-world autonomous robots require versatile controllers that continuously adapt behavior to changing environmental conditions and task specifications. Reinforcement learning (RL) has driven success in modeling complex control strategies in games [53, 42], simulated robotics [2] and real-world systems [27, 29]. However, efficient learning is often conditioned on good parameter selection and may require tuning for each task or domain [19]. Common approaches to hyperparameter optimization leverage parallel or sequential evaluation strategies. While parallel strategies are computationally costly and need not improve on random search [4], sequential strategies [8, 50] are time-intensive with hybrid approaches trading-off one for the other [25]. Gradient-based methods enable online adaptation at the cost of requiring differentiable objectives. Selecting parameters efficiently is essential to learning capable robot controllers.

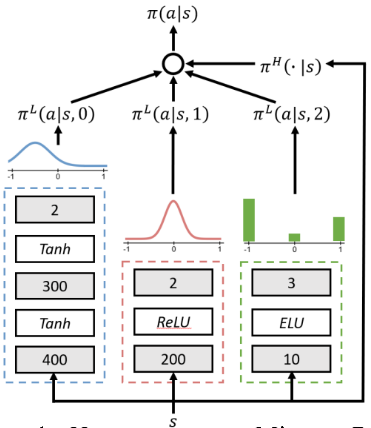

Figure 1: Hyperparameter Mixture Policy (HMP) with diverse low-level distributions. The agent can seamlessly adapt its policy structure to the presented tasks by modulating component activations in a single trial.

The nature of motion planning problems further dictates suitable controller designs for learning. Continuous control can represent intricate transitions in state-action space to yield highly-optimized behaviors through local exploration or generate smooth references for a low-level tracking controller.

*Correspondence to tseyde@mit.edu. †Equal advising.

[2] Please find additional details at https://sites.google.com/view/diversity2021

Discrete control can leverage reduced resolution for coarse exploration and readily encodes bang-bang responses to switching dynamics. Optimal controller selection does not have to be unimodal and can vary with different phases of a task and stages of the learning process. It may then be advantageous to provide agents with a diverse set of controllers that differ in their parameterization. This enables agents to select designs suitable for the presented task and unlock compositional synergies. In order to support this type of compositionality, we extend the perspective of previous work on hierarchical reinforcement learning [51].

In this paper, we propose a hierarchical policy over a diverse mixture of low-level controllers to improve robustness and reduce the necessity for parameter tuning and related data requirements. The low-level controllers are diverse in architecture, hyperparameters, and distribution characteristics to provide the robot with a rich set of controller designs. Our approach then enables learning robots:

- to optimize a set of diverse controller designs concurrently for increased data-efficiency,
- to self-select suitable controllers conditioned on the task for reduced human parameter tuning,
- to compose behaviors from multiple controller designs for exploitation of emergent synergies.

We evaluate performance on a variety of torque-control tasks from the DeepMind Control Suite [52]. Additionally, we investigate learning of PD-control targets for the ANYmal robot in RaiSim [23], which was the foundation for Sim2Real transfer in Lee et al. [27]. Throughout, we demonstrate that enabling agents to operate over a diverse set of controllers guards against individual failure modes and unlocks synergies between different controller designs. While the high-level selector introduces its own hyperparameters, we demonstrate its robustness to loss of state information by modelling unconditional component selection and subjecting the selector to adversarial distractor components.

2 Related Work

The performance of deep RL algorithms is strongly tied to hyperparameter choices [19, 20, 60]. Commonly, hyperparameter optimization is performed in multiple experiments via sets of agents or tasks [5]. Simple parallel strategies include expert selection or grid-search [34, 11], and can be less efficient than random search [4]. Bayesian Optimization provides more structure at the cost of sequential evaluations [50, 48, 49]. Evolutionary strategies [8, 31, 58] enable discontinuous optimization and can evolve parameters at different rates, but typically use sequential evaluation. Population-Based Training (PBT) [25] alternatively evolves parameter variations online in parallel. This requires populations of agents each with their own environment, leading to significant data and computational requirements. Our work is also related to neural architecture search [14]. In particular, similarities can be found to methods that render architecture search differential [40, 32], enabling direct optimization of the effectively-used architecture during a single experiment.

Optimizing hyperparameters during the lifetime of a single agent reduces these requirements. Gradient-based optimization ([3, 57]) yields online adaptation when the objective is differentiable in the parameters. HOOF [39] extends towards non-differential aspects and enables gradient-free off-policy training with hyperparameter schedules. However, the optimized parameters need to directly affect the policy update, precluding e.g. application to architecture search. Related methods have been used to learn architecture schedules [9]. With HMPs, we consider a single agent lifetime to reduce data and computational requirements. We consider policies that vary in their parameters, architecture, and distributional characteristic. Formulating a mixture over these diverse components allows us to evolve multiple controllers in parallel while the agent modulates activation based on expected performance. The approach further enables combining controllers with different parameters.

The problem of hyperparameter choice can further be framed as a contextual bandit problem [41]. In comparison to mixture agents, this formulation does not natively share data across policies with different parameters. Mixtures of specialized motion controllers naturally arise in form of motor primitives [16, 18]. The application of dynamical movement primitives [24] to robot control via a mixture library has been shown in [36]. Mixture distributions have a long-standing history to model diverse and multi-modal data [6]. In RL, they have been applied as a mixture of linear Gaussians in which component specialization is achieved by introducing entropy bounds [10] and provide an inference perspective to the options framework which models an agent via the separation into high-level controller and low-level behaviours [51, 47, 56]. Similarly, quality diversity algorithms evolve diverse skill repertoires which a high-level selector acts on [35, 12, 38]. We build on the training of mixture policies via weighted maximum likelihood optimization, which has been used in connection with information asymmetry to generate diverse, compositional skills [55]. Diversity in mixture policies has been explored to strengthen skill discovery by explicitly optimizing for diversity-related objectives [15, 17, 44, 45] with fixed architectures and a single set of hyperparameters.

3 Preliminaries

We formulate controller optimization as a Markov Decision Process (MDP) defined by the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma\}$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action space, respectively, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ represents the transition distribution, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward mapping, and $\gamma \in [0, 1)$ the discount factor. We define s_t and a_t to be the state and action at time t, respectively. Let $\pi_\theta(a|s)$ denote a policy distribution with parameters θ and define the discounted infinite horizon return $G_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} R(s_{t'}, a_{t'})$, where $s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t)$ and $a_t \sim \pi_\theta(a_t|s_t)$. Our goal is to learn the optimal policy maximizing G_t under unknown dynamics and reward mappings. This is typically done by modeling $\pi_\theta(a_t|s_t)$ as a Gaussian distribution with a neural network predicting the mean and variance from s_t. In this work, we consider a more diverse class of policy distributions.

4 Hyperparameter Mixture Policies

We propose Hyperparameter Mixture Policies (HMP) to train a hierarchical policy over diverse low-level controllers with distinct hyperparameters and distribution characteristics. The resulting mixture is given by

$$\pi_\theta(a|s) = \sum_{i=1}^{M} \pi_\theta^H(i|s) \pi_\theta^L(a|s, i), \qquad (1)$$

with $\pi^H(i|s)$ and $\pi^L(a|s, i)$ as the weight and probability density of component i. Thus, π^H is a high-level selector and π^L a sub-policy from a diverse set of controller designs. The agent then self-selects the most suitable controller for individual phases of a task or stages of the learning process. This enables robust adaptation to a broad range of motion planning problems while reducing the necessity of manual parameter tuning and sequential experiment design.

Algorithm 1: Hyperparameter Mixture Policies

```
Initialize: $N_{\text{step/target}}$: (target) update steps, $N_s$: action samples per state, $\epsilon$: KL bounds
while $k \leq N_{step}$ do
    for $k \leftarrow 1$ to $N_{target}$ do
        Sample batch of trajectories $\tau$ from memory $B$,
        $N_s$ actions from $\pi_{\theta_k}$ to estimate expectations
        Compute gradients over batch for $\pi_\theta$, $\eta$, $\lambda_p$, $Q_\phi$
        $\delta_\pi \leftarrow -\nabla_\theta \sum_{s\sim\mathcal{D}} \sum_{j=1}^{N_s} \Big[\exp\Big(\frac{Q(s,a_j)}{\eta}\Big) \cdot \log \pi_\theta(a_j \mid s)\Big] + \sum_p \lambda_p \Big(\epsilon_p - \mathbb{E}_{s\sim\mathcal{D}}\big[\mathrm{KL}(\pi_\theta || \pi_{\theta_k})\big]\Big)$ (8)
        $\delta_\eta \leftarrow \nabla_\eta g(\eta) = \nabla_\eta \eta\epsilon + \eta \sum_{s\sim\mathcal{D}} \log \frac{1}{N_s} \sum_{j=1}^{N_s} \Big[\exp\Big(\frac{Q(s,a_j)}{\eta}\Big)\Big]$ (5)
        $\delta_{\lambda_p} \leftarrow \nabla_{\lambda_p} \lambda_p \Big(\epsilon_p - \mathbb{E}_{s\sim\mathcal{D}}\big[\mathrm{KL}(\pi_\theta || \pi_{\theta_k})\big]\Big)$ (8)
        $\delta_Q \leftarrow \nabla_\phi \sum_{(s,a)\sim\mathcal{D}} (Q_\phi(s,a) - Q^{\text{ret}})^2$ (10)
        Apply gradients to update $\pi_{\theta_{k+1}}$, $\eta$, $\lambda_p$, $Q_\phi$
    Update target networks $\pi'_\theta = \pi_\theta, Q'_\phi = Q_\phi$
```

4.1 Policy Improvement

We use an actor-critic algorithm where policy improvement relies on two stages as in [1]. First, a non-parametric policy $q(a|s)$ is optimized on samples from the state-action value function Q^π under the constraint of remaining close in expectation to the current parametric policy π_θ. The parametric policy is then updated to better approximate the non-parametric target. By performing the actual policy improvement with a non-parametric policy, we bypass the need for gradient estimation via likelihood ratio [54] or reparametrization trick [26]. In addition, this perspective enables the optimisation of mixture distributions in reinforcement learning without continuous relaxation [28].

Step 1 - Fitting the Non-parametric Policy

As we do not have access to the ground-truth state-action value function Q we employ a learned approximation Q_ϕ, parameterized by ϕ, and formulate the objective at iteration k as

$$\max_q J(q) = \mathbb{E}_{q, s\sim D} \left[Q_\phi(s, a)\right], \qquad (2)$$

$$\text{s.t. } \mathbb{E}_{s\sim D} \left[\mathrm{KL}(q(a|s) || \pi_{\theta_k}(a|s))\right] \leq \epsilon, \qquad (3)$$

where ϵ defines a trust-region around the current parametric policy, π_{θ_k}. This can be solved to provide a closed-form relation in terms of the current parametric policy

$$q_k(a|s) \propto \pi_{\theta_k}(a|s) \exp\Big(\tfrac{Q_\phi(s,a)}{\eta}\Big), \tag{4}$$

where η is computed by minimizing the dual function

$$g(\eta) = \eta\epsilon + \eta\mathbb{E}_{s\sim\mathcal{D}}\left[\log \int \pi_{\theta_k}(a|s) \exp\Big(\tfrac{Q_\phi(s,a)}{\eta}\Big)\mathrm{d}a\right]. \tag{5}$$

Step 2 - Updating the Parametric Policy

The parametric policy π_θ is optimized to approximate the non-parametric policy q by minimizing their KL divergence as

$$\min_\theta L(\theta) = \mathbb{E}_{s\sim\mathcal{D}}\left[\mathrm{KL}(q_k(a|s)||\pi_\theta(a|s))\right]. \tag{6}$$

Plugging in (4) and introducing an additional KL constraint to enable generalization beyond the sample distribution yields

$$\begin{aligned} \max_\theta J(\theta) = \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{\pi_{\theta_k}}\left[\exp\Big(\tfrac{Q_\phi(s,a)}{\eta}\Big)\log\pi_\theta(a|s)\right]\right], \\ \text{s.t. } \mathbb{E}_{s\sim\mathcal{D}}\left[\mathrm{KL}(\pi_{\theta_k}(a|s)||\pi_\theta(a|s))\right] \end{aligned} \tag{7}$$

The update proceeds via a Lagrangian relaxation of the KL constraint enabling the application of gradient-based optimization. We further decouple the KL constraint and define independent constraints for each distributional parameter p in both the high-level selector and low-level control policies [1, 55]. We define these separately per diverse component to accommodate differences in the distributional parameter dynamics between low-level controllers (in comparison to e.g. Wulfmeier et al. [55]). To enable training of diverse distributions with different constraints, we enforce the KL constraints per component. Additional changes to enable training of mixed continuous discrete policies are described in Section 4.3. We obtain updated parameters θ_{k+1} as a solution of

$$\begin{aligned} \max_\theta \min_{\lambda_p>0} L(\theta,\lambda_p) = \mathbb{E}_{s\sim\mathcal{D},\pi_{\theta_k}}\left[\exp\Big(\tfrac{Q_\phi(s,a)}{\eta}\Big)\cdot\log\pi_\theta(a|s)\right] \\ + \sum_p \lambda_p\Big(\epsilon_p - \mathbb{E}_{s\sim\mathcal{D}}\big[\mathrm{KL}(\pi_\theta(a|s)||\pi_{\theta_k}(a|s))\big]\Big) \end{aligned} \tag{8}$$

where we sum over decoupled components, each only varying along its respective parameter p. We also introduce component specific Lagrangian multipliers λ_p and KL bounds ϵ_p. A two component mixture of e.g. a Categorical (α_1) and a Gaussian (μ_2, Σ_2) would then yield $p = \{\alpha_{\text{HL}}, \alpha_1, \mu_2, \Sigma_2\}$.

4.2 Policy Evaluation

In order to stabilize off-policy learning of the state-action value function Q_ϕ we leverage the Retrace algorithm [37]. Here, we truncate the infinite series after 10 steps and bootstrap from the target state-action value network, with details provided in Appendix D. To increase efficiency, we further consider two-step transitions by squashing consecutive timesteps before adding them to memory.

4.3 Combining Continuous and Discrete Distributions

Continuous and discrete policies do not share the same support. Furthermore, actions are subject to numerical cut-off errors. In practice this can result in the action-likelihood of discrete policies being 0 most of the time. To facilitate training with diverse mixtures, we approximate discrete components by piece-wise constant pseudo-densities for backpropagation. Thus, out-of-distribution samples are mapped into the support for probability computation. For query action a and a discrete mixture component i with finite support $\mathcal{C}_i$ we obtain the corresponding piece-wise constant pseudo-density

$$\pi_\theta^L(a|s,i) = \sum_{\tilde{a}\in\mathcal{C}_i} p_i(\tilde{a}|s)\cdot\mathbb{1}_{B_\delta(\tilde{a})}(a), \tag{9}$$

where $p_i(\tilde{a}|s)$ is the probability of $\tilde{a}$ in the original discrete distribution, $\mathbb{1}$ is the indicator function, and $B_\delta(\tilde{a})$ is a ball of radius δ around $\tilde{a}$ (we use $\delta = 0.1$ here). This improves sharing of gradient information between continuous and discrete policies and enables discrete components to train on samples generated by continuous components to accelerate learning.

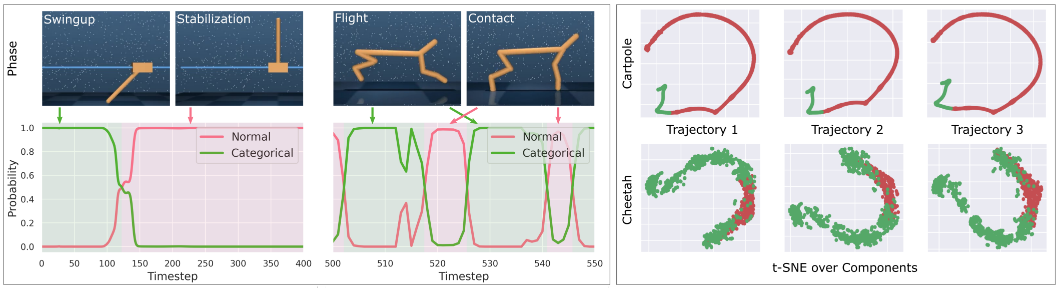

Figure 2: Component specialization within an HMP consisting of a narrow Gaussian ($\sigma_{\text{ini}} = 0.3$) and a coarse Categorical ($n_{\text{bin}} = 2$). On Cartpole, bang-bang control enables fast swing-up and fine-grained control stabilizes the upright. On Cheetah, fine-grained control coordinates the contact phase and bang-bang control retracts the limbs during flight phase. Providing an agent with diverse low-level controllers can unlock synergistic specialization that is consistent across states (t-SNE).

5 Experiments

We benchmark the performance of HMPs on continuous control in the DeepMind Control Suite [52], learning PD-control for ANYmal in RaiSim [22, 23], and manipulation tasks in Metaworld [59]. We further compare to the gradient-free hyperparameter optimizer HOOF [39] in OpenAI Gym [7]. To better isolate the effects of diversity, we consider a standard Gaussian policy and evaluate single parameter variations together with their composition into diverse HMPs. We furthermore investigate HMPs consisting of randomly sampled components and show that strong performance can be recovered. Overall, we find diversity to enable robust learning across tasks and to guard against failure modes of individual components. Our figures visualize performance mean and standard deviation.

5.1 Qualitative Example

We provide an illustrative example of component specialization within a diverse policy in Figure 2. The agent combines a localized Gaussian ($\sigma_{\text{ini}} = 0.3$) with a coarse Categorical ($n_{\text{bin}} = 2$) policy. On a Cartpole swing-up task, the agent leverages bang-bang control for swing-up and continuous control for stabilization. On a Cheetah locomotion task, continuous control coordinates the intricate contact phase and bang-bang control quickly retracts the limbs during the flight phase. Applying t-SNE dimensionality reduction yields consistent clustering across trajectories, indicating consistent component specialization. This aligns with human intuition and highlights the promise of composition, further motivating HMPs and analysis of synergies in heterogeneous mixture policies.

5.2 Heterogeneous Distributions

Compositional Solutions We further evaluate synergies arising from heterogeneous mixtures by combining a narrow Gaussian policy ($\sigma_{\text{ini}} = 0.3$) with a Categorical policy (with $n_{\text{bin}} \in \{2, 9\}$). Figure 3 indicates that performance of the Gaussian significantly improves in combination with a bang-bang policy ($n_{\text{bin}} = 2$) for torque-control (panels 1-5, see also [43]). Conversely, the Gaussian guards against the failure mode induced by bang-bang control on ANYmal (panel 6). We can further replace the Gaussian with a more fine-grained Categorical ($n_{\text{bin}} = 9$) to reach comparable performance on the Control Suite tasks. However, the resulting mixture cannot compensate for signals from the bang-bang controller on ANYmal as discrete control is not well-suited for position reference generation.

Diverse Distributions We broaden our analysis of diverse distributions and consider a Gaussian ($\sigma_{\text{ini}} = 1.0$), Kumaraswamy ($c_{\text{ini}} = 1.0$), Categorical ($n_{\text{bin}} = 5$) and Discrete Gaussian ($n_{\text{bin}} = 5$), as well as their combination into an HMP which we refer to as NKCD. Figure 4 highlights that the HMP is able to solve all tasks, guarding against individual component failure (e.g. K on Quadruped) or premature convergence (e.g. D on Reacher). On ANYmal, the HMP leverages the Kumaraswamy policy to outperform the Gaussian policy, while the Kumaraswamy is suppressed on the Humanoid

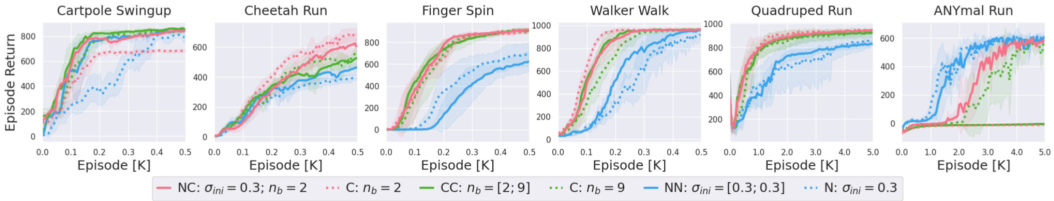

Figure 3: Combining continuous and discrete distributions to unlock synergies. Pairing a narrow Gaussian with a bang-bang controller yields strong performance, guarding against component failure (bang-bang on ANYmal, Gaussian on Finger) and improving on individual performance (HMP on Cartpole). Coarse control can drive exploration, fine-grained control can enable accurate tracking.

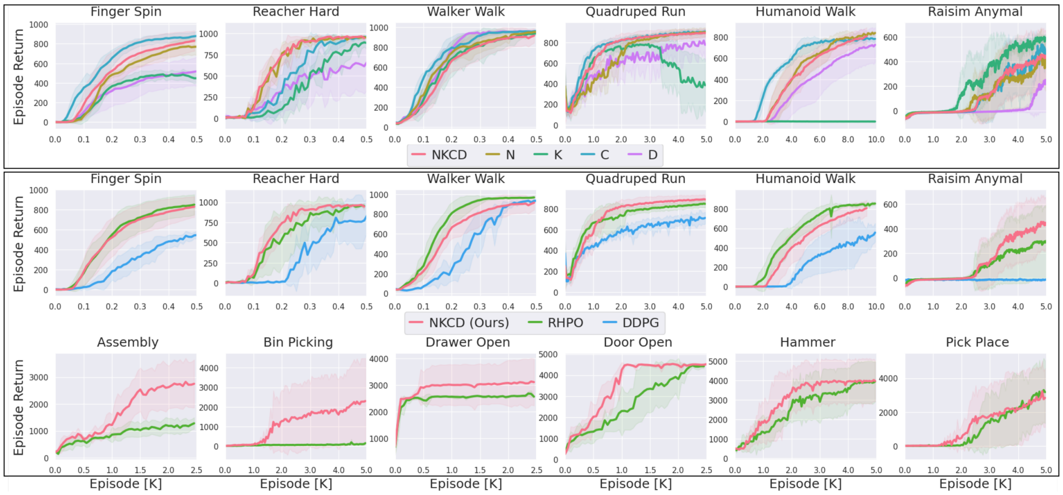

Figure 4: Top - HMP consisting of a Gaussian (N), Kumaraswamy (K), Categorical (C) and Discrete Gaussian (D) heads with individual components for reference. Bottom - HMP instance compared to baselines RHPO and DDPG. Our HMP is robust to sub-policy failure (e.g. K on Humanoid) and yields strong performance especially on the real-world inspired ANYmal and manipulation domains.

to reach strong performance. This underlines the robust performance that diverse HMPs provide by evaluating multiple controller designs jointly, reducing environment-specific tuning and guarding from component failure. We compare the NKCD HMP to the RHPO [55] and DDPG [30] agents. RHPO leverages a homogeneous mixture policy consisting of 4 MPO-type Gaussians ($\sigma_{\text{ini}} = 1.0$). Figure 4 shows that the HMP and RHPO outperform DDPG on all tasks. The HMP performs competitively with RHPO throughout and outperforms RHPO on the real-world inspired ANYmal and manipulation domains. This underlines HMP's ability to enable data-efficient learning by training multiple policy designs in parallel and transferring problem-specific controller selection to the agent.

Random components We consider sampling sub-policies with random hyperparameters. This includes randomizing distribution type, initialization, architecture and activations of each component. Figure 5 provides performance statistics across 10 random instances of an HMP with 10 random components. We observe that random selection yields performance competitive with the optimized RHPO agent. Beyond random selection, the engineer may restrict the space of available sub-policies to inject structural priors into the learning process.

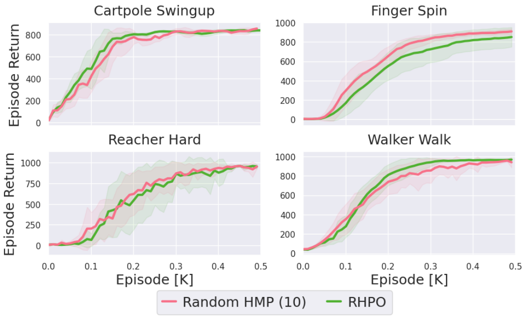

Figure 5: Random sub-policy parameterizations.

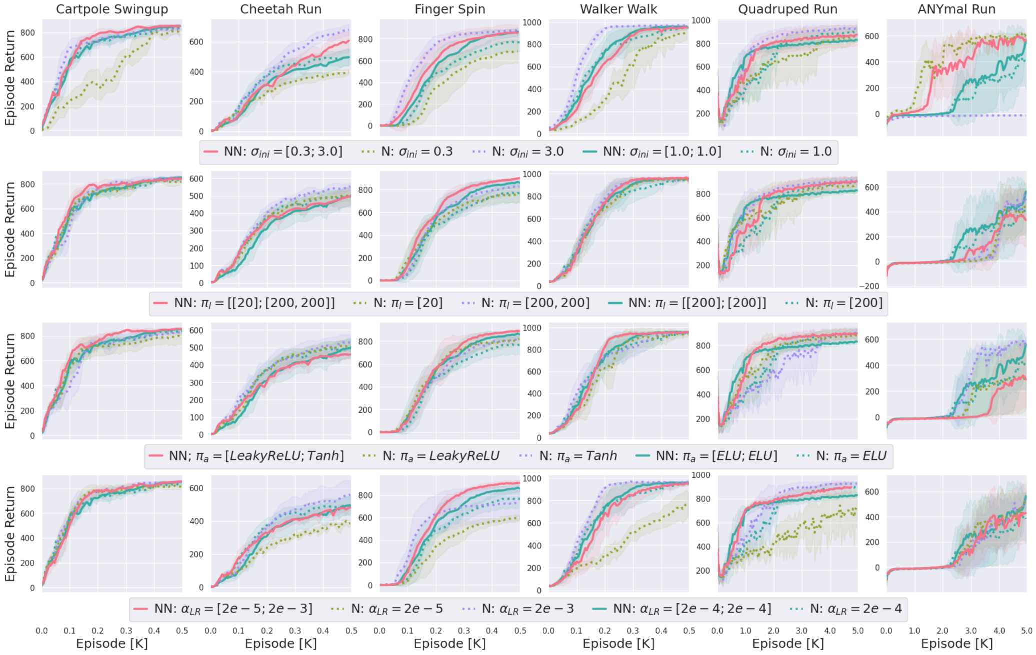

Figure 7: Performance of a diverse mixture (solid red), a homogeneous mixture (solid green), and individual components. The homogeneous mixture uses MPO parameters, while the diverse mixture combines potentially sub-optimal parameter variations. Generally, the diverse mixture performs competitively while guarding against sub-policy failure modes (e.g. $\sigma_{\text{ini}} = 3.0$ on ANYmal, top).

Gradient-free optimization We compare the NKCD HMP to HOOF [39], which introduced a method for gradient-free hyperparameter optimization and evaluated on OpenAI Gym [7]. Their results are provided in Figure 6 for reference. We note that our diverse mixture displays competitve performance on these benchmarks without any fine-tuning for Gym domains.

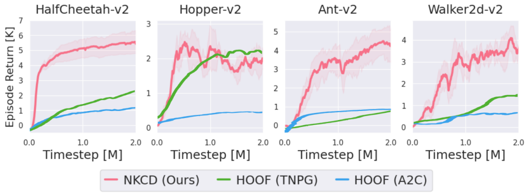

Figure 6: HMP vs. HOOF [39] on OpenAI Gym.

5.3 Hyperparameter Variations

In the following, we evaluate performance of different hyperparameter combinations on continuous control tasks from the DeepMind Control Suite and locomotion on the ANYmal robot in RaiSim. We consider a standard MPO parameterization as the baseline and analyze the impact of combining potentially sub-optimal parameter variations. To account for increased model capacity in mixture policies, we compare to a RHPO-type homogeneous mixture with standard MPO parameters.

Initialization We vary the initial standard deviation of Gaussian policy heads as this can significantly impact the rate of convergence. The bounded action space of the agent is $a \in [-1.0, +1.0]^{|\mathcal{A}|}$. We consider the initial values $\sigma_{\text{ini}} = \{0.3, 1.0, 3.0\}$ with the standard literature value $\sigma^{\text{s}}_{\text{ini}} = 1.0$. Figure 7 (row 1) indicates that the Control Suite tasks favor large variance to drive exploration, while generating position targets on ANYmal requires low variance to avoid instability of the PD controller and subsequent falling. Generally, we find that a diverse policy improves performance over the weaker component, yielding a robust controller that can prevent individual failure modes. This is evident for ANYmal Run, where the high variance policy fails but the diverse mixture succeeds.

Architecture We vary the layer structure with $\pi_l \in \{[20], [200], [200, 200]\}$ and standard value $\pi^{\text{s}}_{\text{l}} = [200]$. Figure 7 (row 2) indicates that performance is robust to architecture variations with

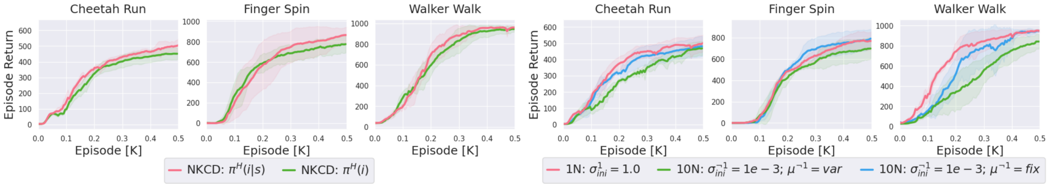

Figure 8: Robustness of HMPs. Left - loss of state-information for a mixture of diverse distributions. Right - pairing a standard Gaussian ($\sigma_{\text{ini}} = 1.0$) with 9 adversarial Gaussians ($\sigma_{\text{ini}} = 10^{-3}$) with learnable or fixed means. The agent still performs well when forced to select a single component for the entirety of an episode, and is able to phase out the poorly-performing adversarial distributions.

increased capacity slightly improving performance. The heterogeneous mixture performs slightly better on the Control Suite tasks while the homogeneous mixture is slightly better on ANYmal Run.

Activations We vary the policy activations with $\pi_{\text{a}} = \{\text{ELU}, \text{Leaky ReLU}, \text{Tanh}\}$ and $\pi_{\text{a}}^{\text{s}} = ELU$. Figure 7 (row 3) shows that the heterogeneous mixture improves performance over both the homogeneous mixture and the individual components on the Control Suite tasks. On ANYmal, combining Leaky ReLU and Tanh activations initially causes quick episode termination and delayed learning. Based on individual performance this is surprising and could indicate that ELU activations are better suited for composition on this task. This is the only instance where we observe reduced performance.

Learning Rate We vary the policy learning rates such that $\alpha_{\text{LR}} \in 2 \times \{10^{-5}, 10^{-4}, 10^{-3}\}$ and $\alpha_{\text{LR}}^{\text{s}} = 2 \times 10^{-4}$. Figure 7 (row 4) indicates that smaller learning rates reduce efficiency. However, pairing a fast with a slow head yields competitive performance, significantly improving over the individual slow component, and outperforms a nominal mixture on the Finger and Quadruped tasks.

5.4 Robustness of the High-level Controller

We evaluate robustness of the high-level module to loss of state information and adversarial components. First, we remove conditioning of the high-level policy on the state and evaluate HMP performance when forced to select a single component for the entirety of an episode. Then, we consider adversarial low-level components by pairing an MPO-type Gaussian head with 9 extremely narrow Gaussian heads that either have variable or fixed means. The narrow Gaussians limit exploration and a constant mean disables active component placement. Figure 8 indicates that the high-level controller is able to adapt its component selection accordingly to yield robust converged performance. This underlines HMP's ability to efficiently select and focus on low-level controllers that are well-suited for a given task, while guarding against potential failure modes of individual components.

6 Conclusion

Finding the right hyperparameters has a considerable impact on performance when enabling robots to learn complex behaviors through interaction and recent progress in machine learning can often be traced back to better hyperparameter settings [46]. A sub-optimal algorithm with thoughtfully tuned hyperparameters easily outperforms a state-of-the-art approach that has not been tuned sufficiently. Tuning requires both domain knowledge and experience with the underlying algorithm. Even then, it still incurs a considerable computational cost that is particularly limiting when relying on real-world data. Our work proposes the use of diverse mixture policies to effectively mitigate this challenge. Moreover, we demonstrate the benefits of combining different distribution types and policy parameterizations from a perspective of compositionality in skill learning. Our Hyperparameter Mixture Policies (HMPs) induce diversity that can help in component specialization during different phases of a task, e.g. where certain movements require either coarse or fine-grained control. It has also the potential to accelerate the learning process, e.g. where more extreme actions enable faster exploration. The approach is easy to use and yields competitive performance across a range of common torque-control benchmark tasks, as well as for generating PD-control targets within a high-fidelity simulation of the ANYmal quadruped, without extensive parameter tuning. While learning algorithms can always benefit from additional tuning, our approach increases robustness and helps to accelerate research in reinforcement learning for complex dynamic robots, in particular when there is no access to extensive computational resources.

Acknowledgments and Disclosure of Funding

Tim Seyde, Wilko Schwarting, Igor Gilitschenski, and Daniela Rus were supported in part by the Office of Naval Research (ONR) Grant N00014-18-1-2830, Qualcomm, and Toyota Research Institute (TRI). This article solely reflects the opinions and conclusions of its authors and not TRI, Toyota, or any other entity. We thank them for their support. The authors further would like to thank Lucas Liebenwein for assistance with cluster deployment, and acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing HPC resources. We would also like to thank the reviewers and program chairs for their helpful feedback and suggestions for improvement.

References

[1] A. Abdolmaleki, J. T. Springenberg, J. Degrave, S. Bohez, Y. Tassa, D. Belov, N. Heess, and M. Riedmiller. Relative entropy regularized policy iteration. *arXiv preprint arXiv:1812.02256*, 2018.

[2] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[3] Y. Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8): 1889–1900, 2000.

[4] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[5] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *25th annual conference on neural information processing systems (NIPS 2011)*, volume 24. Neural Information Processing Systems Foundation, 2011.

[6] C. M. Bishop. Mixture density networks. 1994.

[7] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[8] P. A. Castillo, J. Merelo, A. Prieto, V. Rivas, and G. Romero. G-prop: Global optimization of multilayer perceptrons using gas. *Neurocomputing*, 35(1-4):149–163, 2000.

[9] W. Czarnecki, S. Jayakumar, M. Jaderberg, L. Hasenclever, Y. W. Teh, N. Heess, S. Osindero, and R. Pascanu. Mix & match agent curricula for reinforcement learning. In *International Conference on Machine Learning*, pages 1087–1095. PMLR, 2018.

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

[11] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.

[12] M. Duarte, J. Gomes, S. M. Oliveira, and A. L. Christensen. Evolution of repertoire-based control for robots with complex locomotor systems. *IEEE Transactions on Evolutionary Computation*, 22(2):314–328, 2017.

[13] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester. An empirical investigation of the challenges of real-world reinforcement learning. *arXiv preprint arXiv:2003.11881*, 2020.

[14] T. Elsken, J. H. Metzen, F. Hutter, et al. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21, 2019.

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

[16] S. F. Giszter, F. A. Mussa-Ivaldi, and E. Bizzi. Convergent force fields organized in the frog's spinal cord. *Journal of neuroscience*, 13(2):467–491, 1993.

[17] A. Goyal, S. Sodhani, J. Binas, X. B. Peng, S. Levine, and Y. Bengio. Reinforcement learning with competitive ensembles of information-constrained primitives. *arXiv preprint arXiv:1906.10667*, 2019.

[18] C. B. Hart and S. F. Giszter. A neural basis for motor primitives in the spinal cord. *Journal of Neuroscience*, 30(4):1322–1336, 2010.

[19] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[20] P. Henderson, J. Romoff, and J. Pineau. Where did my optimum go?: An empirical analysis of gradient descent optimization in policy gradient methods. *arXiv preprint arXiv:1810.02525*, 2018.

[21] M. Hoffman, B. Shahriari, J. Aslanides, G. Barth-Maron, F. Behbahani, T. Norman, A. Abdolmaleki, A. Cassirer, F. Yang, K. Baumli, S. Henderson, A. Novikov, S. G. Colmenarejo, S. Cabi, C. Gulcehre, T. L. Paine, A. Cowie, Z. Wang, B. Piot, and N. de Freitas. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020. URL `https://arxiv.org/abs/2006.00979`.

[22] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44. IEEE, 2016.

[23] J. Hwangbo, J. Lee, and M. Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, 2018. URL `www.raisim.com`.

[24] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1398–1403. IEEE, 2002.

[25] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[26] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[27] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020.

[28] A. C. Li, C. Florensa, I. Clavera, and P. Abbeel. Sub-policy adaptation for hierarchical reinforcement learning. *arXiv preprint arXiv:1906.05862*, 2019.

[29] T. Li, R. Calandra, D. Pathak, Y. Tian, F. Meier, and A. Rai. Planning in learned latent action spaces for generalizable legged locomotion. *IEEE Robotics and Automation Letters*, 6(2): 2682–2689, 2021.

[30] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[31] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*, 2017.

[32] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[33] P. A. Mitnik. New properties of the kumaraswamy distribution. *Communications in Statistics-Theory and Methods*, 42(5):741–755, 2013.

[34] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[35] J.-B. Mouret and J. Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[36] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3): 263–279, 2013.

[37] R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare. Safe and efficient off-policy reinforcement learning. *arXiv preprint arXiv:1606.02647*, 2016.

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

[39] S. Paul, V. Kurin, and S. Whiteson. Fast efficient hyperparameter tuning for policy gradient methods. 2019.

[40] S. Saxena and J. Verbeek. Convolutional neural fabrics. *arXiv preprint arXiv:1606.02492*, 2016.

[41] T. Schaul, D. Borsa, D. Ding, D. Szepesvari, G. Ostrovski, W. Dabney, and S. Osindero. Adapting behaviour for learning progress. *arXiv preprint arXiv:1912.06910*, 2019.

[42] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[43] T. Seyde, I. Gilitschenski, W. Schwarting, B. Stellato, M. Riedmiller, M. Wulfmeier, and D. Rus. Is bang-bang control all you need?: Solving continuous control with bernoulli policies. *Advances in Neural Information Processing Systems*, 35, 2021.

[44] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

[45] A. Sharma, M. Ahn, S. Levine, V. Kumar, K. Hausman, and S. Gu. Emergent real-world robotic skills via unsupervised off-policy reinforcement learning. *arXiv preprint arXiv:2004.12974*, 2020.

[46] P. T. Sivaprasad, F. Mai, T. Vogels, M. Jaggi, and F. Fleuret. Optimizer benchmarking needs to account for hyperparameter tuning. In *International Conference on Machine Learning*, pages 9036–9045. PMLR, 2020.

[47] M. Smith, H. Hoof, and J. Pineau. An inference-based policy gradient method for learning options. In *International Conference on Machine Learning*, pages 4703–4712. PMLR, 2018.

[48] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. *arXiv preprint arXiv:1206.2944*, 2012.

[49] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust bayesian neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4141–4149, 2016.

[50] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

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

[52] Y. Tassa, S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, and N. Heess. dm_control: Software and tasks for continuous control. *arXiv preprint arXiv:2006.12983*, 2020.

[53] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[54] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[55] M. Wulfmeier, A. Abdolmaleki, R. Hafner, J. T. Springenberg, M. Neunert, T. Hertweck, T. Lampe, N. Siegel, N. Heess, and M. Riedmiller. Compositional transfer in hierarchical reinforcement learning. *arXiv preprint arXiv:1906.11228*, 2019.

[56] M. Wulfmeier, D. Rao, R. Hafner, T. Lampe, A. Abdolmaleki, T. Hertweck, M. Neunert, D. Tirumala, N. Siegel, N. Heess, et al. Data-efficient hindsight off-policy option learning. *arXiv preprint arXiv:2007.15588*, 2020.

[57] Z. Xu, H. van Hasselt, and D. Silver. Meta-gradient reinforcement learning. *arXiv preprint arXiv:1805.09801*, 2018.

[58] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, pages 1–5, 2015.

[59] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020.

[60] B. Zhang, R. Rajan, L. Pineda, N. Lambert, A. Biedenkapp, K. Chua, F. Hutter, and R. Calandra. On the Importance of Hyperparameter Optimization for Model-based Reinforcement Learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

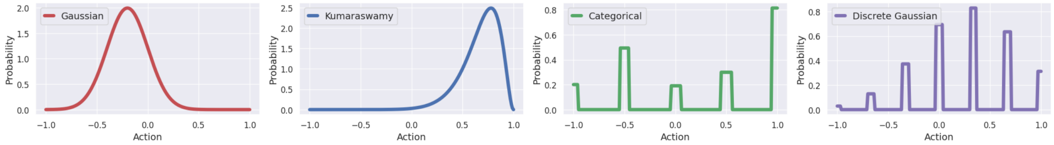

Figure 9: Probability functions of the diverse distribution types considered here. Left to right: Gaussian, Kumaraswamy, Categorical, and discrete Gaussian. We visualize a 5-bin Categorical and a 7-bin discrete Gaussian with their pseudo-probabilities. Our HMPs can combine diverse low-level controllers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

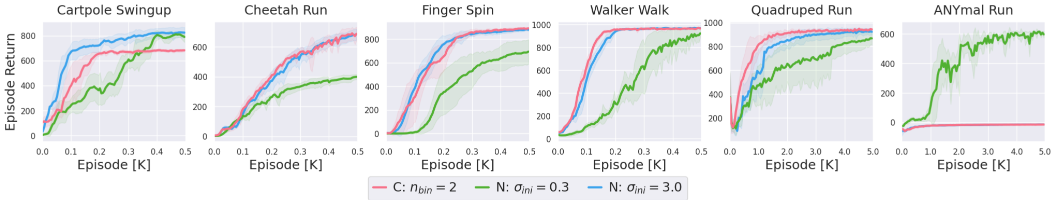

Figure 10: Our approach yields robust performance when tasked to solve continuous control with discrete actions. For instance, the coarse Categorical (C, $n_{\text{bin}} = 2$) performs competitively to the wide Gaussian (N, $\sigma_{\text{ini}} = 3.0$) on the torque controlled DeepMind Control Suite 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	×0.25,×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

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

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.

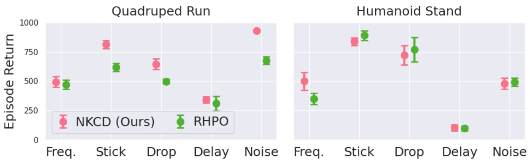

Figure 11: HMP under real-world disturbances. Our diverse mixtures can improve robustness over homogeneous baselines and aid in generalization.