

## A Appendix

### A.1 Unsupervised Skill Discovery Algorithms

**DIAYN: Diversity Is All You Need** DIAYN Eysenbach et al. [4] aims to learn a skill-conditioned policy  $\pi_\theta(\mathbf{a} \mid \mathbf{s}, \mathbf{z})$  by maximizing the mutual information (MI) between states and skills  $I(\mathbf{S}; \mathbf{Z}) \triangleq D_{\text{KL}}(p(\mathbf{s}, \mathbf{z}) \parallel p(\mathbf{s})p(\mathbf{z})) \equiv \mathcal{H}(\mathbf{Z}) - \mathcal{H}(\mathbf{Z} \mid \mathbf{S})$ , where  $\mathcal{H}(\cdot)$  denotes the Shannon or differential entropy. Intuitively, minimizing  $\mathcal{H}(\mathbf{Z} \mid \mathbf{S})$  means that the skill  $\mathbf{z}$  should be easy to infer given the state  $\mathbf{s}$ . This is implemented by learning a discriminator  $q_\phi(\mathbf{s} \mid \mathbf{z})$  that approximates the posterior  $p(\mathbf{s} \mid \mathbf{z})$ . The policy and discriminator form a cooperative game: the discriminator predicts the skill that led to the policy visiting certain states, while the policy seeks to visit states that make it easy for the discriminator to identify the skill. The resulting reward is

$$r_{\text{DIAYN}}(\mathbf{s}, \mathbf{a}, \mathbf{z}) = \log q_\phi(\mathbf{z} \mid \mathbf{s}) - \log p(\mathbf{z}) \quad (2)$$

Note,  $\log p(\mathbf{z})$  only needs to be included in the reward term if  $p(\mathbf{z})$  is not uniform. The discriminator is trained by maximizing the log-likelihood of the posterior.

$$\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}), \mathbf{s} \sim \pi_\theta(\mathbf{z})} \log q_\phi(\mathbf{z} \mid \mathbf{s}) \quad (3)$$

In practice, the discriminator predicts parameters of a distribution over the skill space, which depends on the selection of the prior  $p(\mathbf{z})$ .

The authors of the original paper use a categorical distribution for the prior  $p(\mathbf{z})$ . and noted that learning with continuous skill distributions such as uniform or Gaussian distribution degrades performance. [34] showed however, that to learn more skills, using a continuous distribution yields better results than using a large number of discrete skills. Depending on the selection of the prior, the parameterization of the discriminator has to be adjusted. Importantly, the support of the posterior has to contain the support of the prior. In Table 4 we list different combinations of priors and posteriors we tested. We found that Dirichlet-distributed skills offer a good trade-off between continuous skill expressiveness and discriminability accuracy.

Table 4: **Possible choices for skill distributions.** Depending on the choice of the prior distribution for the skills, we choose the posterior according to this table.

Prior $p(\mathbf{z})$	Posterior $q(\mathbf{z} \mid \mathbf{s})$
Uniform categorical	Categorical
Uniform continuous	Gaussian
Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$	Gaussian
Uniform on sphere	Von Mises-Fisher
Symmetric Dirichlet	Dirichlet

**METRA: Metric-Aware Abstraction** METRA Park et al. [5] (as well as LSD [19] and CSD [8]) aims to learn a skill-conditioned policy by learning an encoder  $\phi$  that maps states into a latent space of the same dimensionality as the skill space. The objective is to maximize the alignment of latent transitions  $\phi(\mathbf{s}') - \phi(\mathbf{s})$  with the skill  $\mathbf{z}$  under a constraint  $\|\phi(\mathbf{s}) - \phi(\mathbf{s}')\|_2 \leq d(\mathbf{s}', \mathbf{s})$ , with distance metric  $d$ . LSD proposes to use the Euclidian distance between the states  $d(\mathbf{s}', \mathbf{s}) = \|\mathbf{s}' - \mathbf{s}\|$ , CSD [20] proposes to use controllability aware distance metric, while METRA proposes to use temporal distance, i.e., the minimum number of environment steps to reach  $\mathbf{s}'$  from  $\mathbf{s}$ , which in this setup is simply  $d(\mathbf{s}', \mathbf{s}) = 1$ .

The objective for the encoder becomes:

$$\mathcal{J}_{\text{METRA}}(\theta, \phi) = \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}), \mathbf{s} \sim \pi_\theta(\mathbf{z})} [(\phi(\mathbf{s}') - \phi(\mathbf{s}))^\top \mathbf{z}] \text{ s.t. } \|\phi(\mathbf{s}) - \phi(\mathbf{s}')\|_2 \leq d(\mathbf{s}', \mathbf{s}), \forall (\mathbf{s}, \mathbf{s}') \in \mathcal{S}_{\text{adj}} \quad (4)$$

In practice, this is trained via a dual variable.

440 The agent is rewarded if its actions result in state transitions that align with the skill in the latent state  
441 space.

$$r_{\text{METRA}}(\mathbf{s}, \mathbf{z}, \mathbf{s}') = (\phi(\mathbf{s}') - \phi(\mathbf{s}))^\top \mathbf{z} \quad (5)$$

442 This objective leads to skills that move maximally fast through the state space, which might not be  
443 desirable. [8] proposed a norm matching objective to also control the execution speed of a skill:

$$\mathcal{J}_{\text{NM}}(\theta, \phi) = \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}), \mathbf{s} \sim \pi_\theta(\mathbf{z})} [\|\phi(\mathbf{s}') - \phi(\mathbf{s}) - \mathbf{z}\|^2] \text{ s.t. } \|\phi(\mathbf{s}) - \phi(\mathbf{s}')\| \leq d(\mathbf{s}', \mathbf{s}), \forall (\mathbf{s}, \mathbf{s}') \in \mathcal{S}_{\text{adj}} \quad (6)$$

444 The reward becomes a function of the error between the skill and the latent transition.

$$r_{\text{METRA,nm}}(\mathbf{s}, \mathbf{z}, \mathbf{s}') = (1 + \sigma \|(\phi(\mathbf{s}_{t+1}) - \phi(\mathbf{s}_t)) - \mathbf{z}\|_2^2)^{-1} \quad (7)$$

445 with a scaling factor  $\sigma \in \mathbb{R}$

446 This objective however loses the alignment component completely, making it harder to train. To  
447 combine the strengths of both objectives, start the learning process with the original alignment  
448 objective and, depending on current performance, we smoothly transition to the norm matching  
449 objective. We define an interpolation parameter  $\alpha_{\text{mix}} \in [0, 1]$  that depends on cosine similarity  
450 between  $(\phi(\mathbf{s}') - \phi(\mathbf{s}))$  and  $\mathbf{z}$ . We then calculate the objective as:

$$\mathcal{J}_{\text{METRA mix}}(\theta, \phi) = (1 - \alpha_{\text{mix}})\mathcal{J}_{\text{METRA}}(\theta, \phi) + \alpha_{\text{mix}}\mathcal{J}_{\text{NM}}(\theta, \phi) \quad (8)$$

451 This weight is directly analogously used for the reward calculation and encoder loss.

452 **DUSDi: Disentangled Unsupervised Skill Discovery** To learn disentangle skills, Hu et al. [6]  
453 proposes to learn two discriminators per factor, based on DIAYN. The first discriminator predicts the  
454 skill factor from the respective state factor  $q_i(\mathbf{z}_i | \mathbf{s}_i)$  while the second discriminator predicts the skill  
455 from every other state factor:  $q_{\neg i}(\mathbf{z}_i | \mathbf{s}_{\neg i})$ , where  $\mathbf{s}_{\neg i} \in \mathcal{S}_{\neg i} = \mathcal{S}_1 \times \dots \times \mathcal{S}_{i-1} \times \mathcal{S}_{i+1} \times \dots \times \mathcal{S}_N$ . The  
456 first reward is the standard DIAYN reward, while the second one has the same formulation but is used  
457 as a penalty. The harder it is to infer a skill factor given other state factors, the more disentangled  
458 they are.

$$r_{\text{DUSDI}}(\mathbf{s}, \mathbf{a}, \mathbf{z}) \triangleq \sum_{i=1}^N q_i(\mathbf{z}_i | \mathbf{s}_i) - \gamma q_{\neg i}(\mathbf{z}_i | \mathbf{s}_{\neg i}),$$

459 where  $\gamma < 1$  is a hyperparameter that controls the importance of the entanglement penalty relative to  
460 the skill-factor association (typically  $\gamma = 0.1$ ).

## 461 A.2 Symmetry Augmentation

462 Symmetry augmentation can help boosting sample efficiency and learning smoother behaviors. Mittal  
463 et al. [30] proposes to simply augment the collected data instead of introduction an extra symmetry  
464 objective, or enforcing symmetry in the network architecture.

465 The quadruped ANYmal-D, is left-right and front-back symmetric, resulting in four symmetry  
466 transformations: identity, left-right reflection, front-back reflection, 180° rotation about the z-axis.

467 So far, symmetry biases have not been used as part of unsupervised skill discovery. It might however  
468 be useful to learn symmetric skills and to boost exploration. To utilize symmetry augmentation, the  
469 MDP needs to have symmetries, which requires the reward to be invariant to symmetry transforma-  
470 tions. In general, this is not the case in skill discovery, it can however, be enforced by averaging over  
471 all symmetries:

$$r_{\text{sym}}(\mathbf{s}, \mathbf{a}, \mathbf{z}) = \frac{1}{K} \sum_{i=1}^K r(M_s^i(\mathbf{s}), M_a^i(\mathbf{a}), M_z^i(\mathbf{z})), \quad (9)$$

472 where  $r_{\text{sym}}$  is a reward that is guaranteed to be invariant over all symmetries.

473 **Skill Mirroring Composition** The function that mirrors skills,  $M_z$ , can be chosen freely so long  
 474 as it preserves (i) the invariance of the skill prior and (ii) the group composition of the underlying  
 475 symmetries, for instance, mirroring *left-right* followed by *front-back* yields the same physical  
 476 transformation as a  $180^\circ$  rotation about the  $z$ -axis; the same composition must hold in skill space.  
 477 Otherwise symmetry augmentation introduces contradictory training signals. Concretely, if

$$M_s^1(M_s^2(s_i)) = M_s^3(s_i) \quad \forall s_i \in \mathcal{S}_i,$$

478 but there exists a skill

$$z_i \in \mathcal{Z}_i \quad \text{s.t.} \quad M_z^1(M_z^2(z_i)) \neq M_z^3(z_i),$$

479 then symmetry augmentation can produce tuples with the *same* state but *different* mirrored skills:

$$(M_s^3(s_i), \dots, M_z^1(M_z^2(z_i))) \quad \text{and} \quad (M_s^3(s_i), \dots, M_z^3(z_i)).$$

480 Because states are paired with ambiguous skills, any skill-conditioned reward or discriminator cannot  
 481 remain consistent, yielding irreconcilable gradients and hindering learning. Therefore  $M_z$  must  
 482 satisfy  $M_z^1 \circ M_z^2 = M_z^3$ , to ensure coherent symmetry-augmented training.

483 **Skill Mirroring Implementation** For **DIAYN**, we mirror skills such that subskills form a Latin  
 484 square. We use the following skill permutations:

$$\begin{aligned} M_z^1(z_i) &= [z_i^1; z_i^2; z_i^3; z_i^4] \\ M_z^2(z_i) &= [z_i^3; z_i^4; z_i^1; z_i^2] \\ M_z^3(z_i) &= [z_i^2; z_i^1; z_i^4; z_i^3] \\ M_z^4(z_i) &= [z_i^4; z_i^3; z_i^2; z_i^1] \end{aligned}$$

485 Permuting sub-skills gives the latent space room for states that are invariant to certain symmetries.  
 486 Let  $\mathcal{S}_{\text{sym}(i,j)} = \{s \in \mathcal{S} \mid M_s^i(s) = M_s^j(s)\}$  be such states (e.g., forward/backward velocity is  
 487 invariant to a left-right flip). Whenever  $\|\mathcal{S}_{\text{sym}(i,j)}\| > 1$  is, we require matching skills  $\mathcal{Z}_{\text{sym}(i,j)} =$   
 488  $\{z \in \mathcal{Z} \mid M_z^i(z) = M_z^j(z)\}$ , which our permutation-based mirroring provides automatically:

$\mathcal{Z}_{\text{sym}(i,j)}$	$j = 1$	$j = 2$	$j = 3$	$j = 4$
$i = 1$	$\mathcal{Z}$	$[a; b; a; b]$	$[a; a; b; b]$	$[a; b; b; a]$
$i = 2$	$[a; b; a; b]$	$\mathcal{Z}$	$[a; b; b; a]$	$[a; a; b; b]$
$i = 3$	$[a; a; b; b]$	$[a; b; b; a]$	$\mathcal{Z}$	$[a; b; a; b]$
$i = 4$	$[b; a; a; b]$	$[a; a; b; b]$	$[a; b; a; b]$	$\mathcal{Z}$

489 where  $a, b \in \mathbb{R}^n$  denote subskills. This pattern guarantees that a state symmetric under  $i$  and  $j$  can  
 490 always be paired with a unique skill lying in  $\mathcal{Z}_{\text{sym}(i,j)}$ .

491 If  $\mathcal{S}_{\text{sym}(i,j)} = \mathcal{S}$ , i.e., every state of a factor is invariant under a pair  $(i, j)$ , then, the skill map must  
 492 satisfy  $\mathcal{Z}_{\text{sym}(i,j)} = \mathcal{Z}$  as well. For the heading-rate factor, a scalar that flips sign under left-right  
 493 or front-back reflections but is unchanged by their composition, both reflections are merged into  
 494 a single “flip”, resulting in the symmetries  $\{1, 2\} = (\text{identity}, \text{flip})$ . With two subskills we set  
 495  $M_z^1(z_i) = [z_i^1; z_i^2]$  and  $M_z^2(z_i) = [z_i^2; z_i^1]$ . For states with no symmetries, e.g., the base height, we  
 496 omit symmetry augmentation for the skills completely.

497 For **METRA**, we treat the skill vector  $z_i$  as a directional proxy for the associated state factor and  
 498 mirror it accordingly. While the skill does not lie in state space, applying the same symmetry  
 499 transformations introduces an inductive bias that aligns skill behavior with the symmetries of the  
 500 underlying physical system.

### 501 A.3 Implementation Details

502 For the policy, we use a 3-layer MLP [512, 256, 128] with elu activations. The action space is  
 503 12-dimensional. Per action, the policy predicts the mean and log std of a Gaussian distribution, of  
 504 which we clamp the standard deviation to the range  $[e^{-5}, e^2]$ . During training, actions are sampled  
 505 from the predicted distribution. During deployment, the action is the predicted mean. Training is  
 506 done in Isaac Lab [32] with 2048 parallel environments on a RTX 3090 GPU.

We list hyperparameters for PPO in Table 5, for METRA in Table 7 and for DIAYN in Table 8. The rewards for the style factor are listed in Table 9 and the regularization penalties in Table 10. The policy observations can be found in Table 6

Table 5: PPO Hyperparameters

Hyperparameter	Value
PPO clip ratio	0.2
Value clip ratio	0.2
Num env steps before update	24
Num learning epochs	5
Num mini batches	4
Learning rate	1.0e-3
Discount factor	0.99
GAE lambda	0.95
KL target	0.01
Max grad norm	1.0

Table 6: Policy Observations

Name	Dim
Base xy-position in world frame	2
Height scan	231
Base linear velocity	3
Base angular velocity	3
Projected gravity	3
Previous action	12
Joint position	12
Joint velocity	12

Table 7: METRA Hyperparameters

Hyperparameter	Value
Learning rate	1.0e-4
Initial Lagrange multiplier	30.0
Lagrange multiplier lr	1e-4
Lagrange multiplier slack	1e-5
Objective switching range	(0.5, 0.7)
Network	MLP: [256, 256]
Norm matching $\sigma$	10.0

Table 8: DIAYN Hyperparameters

Hyperparameter	Value
Skill distribution	dirichlet
Disentanglement $\lambda$	0.1
Network	MLP: [256, 256]
Learning rate	1e-4
Dirichlet param range	(0.05, 1.0)

Table 9: Style Factor Rewards

Name	Objective	Weight
Joint torques	$\ \tau\ _2^2$	-1.0e-3
Joint acceleration	$\ \ddot{\mathbf{q}}\ _2^2$	-1.0e-5
Action rate	$\ \mathbf{a}_t - \mathbf{a}_{t-1}\ _2^2$	-0.2
Action norm	$\ \mathbf{a}\ _2^2$	-0.4
Undesired Contacts	$\sum_{b \in \{\text{Thighs, Shanks, Base}\}} \mathbf{1}[\text{contact}(b)]$	-30.0
Base height	$\ p_z - 0.55\ ^2$	-10.0
Flat orientation	$\ \mathbf{g}_{b,xy}\ _2^2$	-10.0

**Critic Decomposition** In DUSDi [6], the authors also propose to decompose the Q function as a sum of Q values over the individual factors. We do the same but with value functions. Additionally, instead of one value function per factor, we may have an ensemble of value functions per factor due to UCB exploration [14]. To do so, we need to store the individual factor rewards separately as the aggregated rewards are only required for the policy update. As a result, we do weighted aggregation over the advantage estimates:

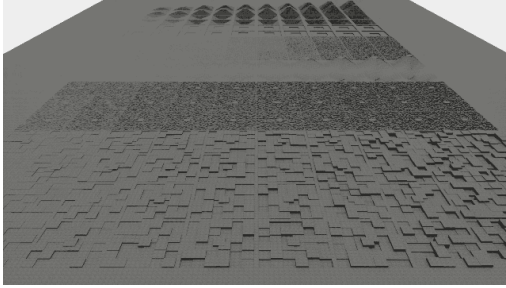
$$A = \lambda_{N+1} A_{\text{style}} + \sum_{i=1}^N \lambda_i (A_{i,\mu} + \lambda_{\text{UCB}} A_{i,\sigma}). \quad (10)$$

where  $A_{\text{style}}$  is the advantage of the style factor, and  $A_{i,\mu}, A_{i,\sigma}$  are the mean and standard deviation of advantage ensembles per factor. The policy is updated with aggregated advantage  $A$ .

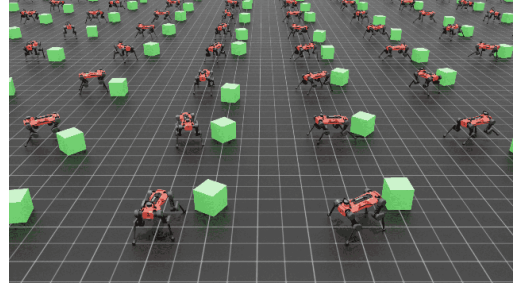
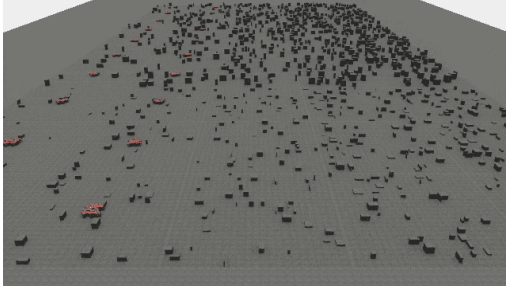
**Environments** In Fig. 6 we visualize all types of terrains we used during our experiments.

Table 10: Regularization Rewards

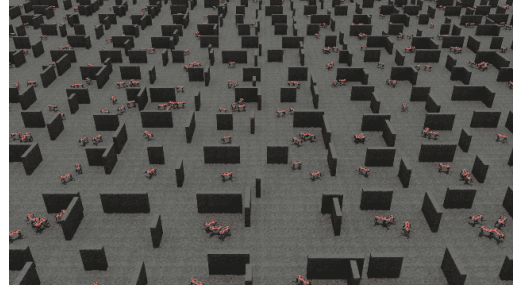
Name	Objective	Weight
Joint torques	$\ \boldsymbol{\tau}\ _2^2$	-1.0e-3
Joint acceleration	$\ \ddot{\mathbf{q}}\ _2^2$	-2.5e-7
Action rate	$\ \mathbf{a}_t - \mathbf{a}_{t-1}\ _2^2$	-0.05
Torque limits	$\ \max(\boldsymbol{\tau} - \tau_{\max}, 0)\ _1 + \ \min(\boldsymbol{\tau} - \tau_{\min}, 0)\ _1$	-15.0
Torque ratio limits	$\ \max(\boldsymbol{\tau} - 0.75\tau_{\max}, 0)\ _1 + \ \min(\boldsymbol{\tau} - 0.75\tau_{\min}, 0)\ _1$	-15.0
Joint vel limits	$\ \min(\max( \dot{\mathbf{q}}  - \dot{\mathbf{q}}_{\lim}, 0), 1.0)\ _1$	-10.0
Joint pos limits	$\ \max(\mathbf{q} - \mathbf{q}_{\lim}^{\text{upper,soft}}, 0) + \min(\mathbf{q} - \mathbf{q}_{\lim}^{\text{lower,soft}}, 0)\ _1$	-10.0
Upside down termination	1 [flipped termination]	-5000



(a) Rough curriculum terrain, similar to the one used by Rudin et al. [35].

(b) Flat terrain with a movable box ( $0.5 \times 0.5 \times 0.5 \text{ m}^3$ , 10 kg).

(c) Curriculum terrain with increasing density of obstacles.



(d) Terrain with walls.

Figure 6: Terrains used in our experiments

## Factor Weighting

### A.4 Experiment Details

In this section we provide further details for the experiments mentioned in the main paper. We expected factor weighing to alleviate the issue in factorized skill learning when certain factors cannot be interacted with simultaneously. To evaluate this, we factorized the position factor further into four quadrants (NE ( $x, y > 0$ ), SE ( $x < 0, y > 0$ ), SW ( $x < 0, y < 0$ ), NE ( $x > 0, y < 0$ )). The robot cannot be in multiple quadrants simultaneously which conflicts with skills commanded for each factor in each quadrant. We trained all factors with METRA

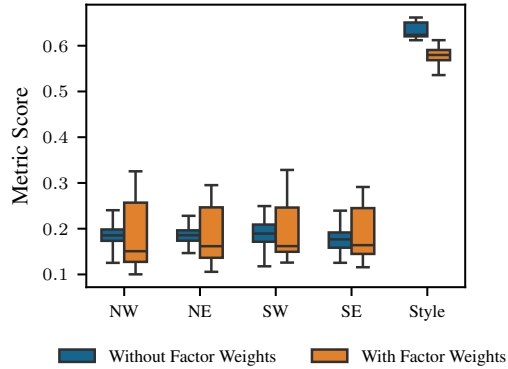


Figure 7: Effect of factor weighting on conflicting factors. Weighting does not improve the metrics

and without symmetry augmentation. In Fig. 7 we visualize the cosine similarity between latent state transition and commanded skills, and the scaled style reward for setups with and without weighting. Weighing does not change the performance significantly.

**Symmetry Augmentation** In Fig. 8, we show how symmetry augmentation affects skill discovery performance across different state factors. METRA is used for the position factor, while DIAYN is used for the others. Notably, symmetry augmentation reduces the encoder accuracy in METRA, while it has little effect on the discriminator accuracy in DIAYN. We hypothesize that this is because METRA relies on a more direct, directional interpretation of the skill vector, making it more sensitive to the constraints introduced by symmetry augmentation.

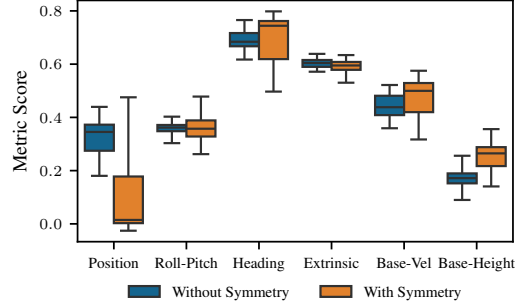


Figure 8: Effect of symmetry augmentation on skill learning metrics.

## A.5 Additional Tasks

**Loco-manipulation** We attempted to learn loco-manipulation skills by placing a movable box in the environment and adding the box pose as an additional state factor. However, this setup alone failed to produce meaningful skills, as interactions with the box were rare and the resulting intrinsic rewards from the box factor were weak. To address this, we incorporated exploration guidance using RND [13] and UCB [14]. Neither method led to significant improvements. With RND, the prediction error rapidly decreased before the agent could discover interactions with the box, providing little incentive to explore it. With UCB, the agent received persistently high intrinsic rewards due to high ensemble disagreement, caused by low-quality value estimates, without corresponding learning progress, leading to unstructured and unproductive behavior.

Strouse et al. [12] showed that DIAYN suffers from poor exploration. To still encourage high diversity, we can add an exploration bonus on top of the pure skill discovery reward. A simple form of intrinsic motivation is random network distillation (RND) Burda et al. [13], which encourages exploration by rewarding states that are rarely visited. Another method to encourage exploration is to use ensemble disagreement as an exploration reward. One way to implement this, proposed by Strouse et al. [12], is by defining multiple discriminators  $q_{\phi_i}(z | s)$  and then reward the agent for high entropy of the mixture compared to the mean entropy.

$$r_{\text{DISDAIN}} = \mathcal{H} \left( \frac{1}{N} \sum_{i=0}^N q_{\phi_i}(z | s) \right) - \frac{1}{N} \sum_{i=0}^N \mathcal{H}(q_{\phi_i}(z | s)) \quad (11)$$

Depending on the distribution, this may not be easy to implement. A simpler approach based on ensemble disagreement uses the variance of the rewards as an exploration bonus:

$$r_{\text{EXPLORE}} = \text{Var}([\log q_{\phi_1}(z | s), \dots, \log q_{\phi_N}(z | s)]) \quad (12)$$

This is similar to the method proposed by Chen et al. [14], which used a Bayesian learning approach by updating the policy based on an upper confidence bound (UCB) for the value estimate by defining an ensemble of value functions and adding a disagreement bonus. However, this method also did not help to discover meaningful loco-manipulation skills.

## A.6 High-level Skill Learning

We investigated whether our pretrained skills could be reused to learn higher-level navigation behaviors. In this setup, the high-level policy outputs skill vectors directly, using the same pretrained low-level policies. We evaluated this in environments with randomly placed obstacles (Figs. 6c and 6d), training only METRA on the base position factor. While the agent learned to cover the space,

576 it struggled to avoid obstacles, even with access to a 2D planar distance scan. We hypothesize this is  
 577 due to the lack of an extrinsic signal encouraging obstacle avoidance.

578 We also explored using this setup to learn box-pushing behaviors. To simplify the task, the box could  
 579 be moved by simple collisions. Using METRA on the box position factor, the agent learned to push  
 580 the box effectively. However, since the low-level policy was not trained in the presence of the box, it  
 581 lacked any meaningful manipulation capabilities. As a result, the agent relied on forceful collisions  
 582 to move the box, an approach that is unsafe for real-world deployment.

583 **DIAYN Distribution type** We evaluate the impact of different Dirichlet priors on skill diversity  
 584 for DIAYN-trained factors. We compare three setups: a fixed high-concentration prior ( $\alpha = 0.05$ ),  
 585 a fixed low-concentration prior ( $\alpha = 1.0$ ), and  
 586 a curriculum that linearly increases  $\alpha$  from 0.05  
 587 to 1.0 based on discriminator accuracy. As  
 588 shown in Fig. 9, the curriculum setup results  
 589 in greater variability and often higher diversity  
 590 scores for the Base Height and Heading Rate  
 591 factors. This suggests that starting with sparse  
 592 skill sampling helps early specialization, while  
 593 gradually broadening the support encourages  
 594 later diversity. In contrast, the Roll-Pitch fac-  
 595 tor appears less sensitive to the choice of prior,  
 596 likely due to its lower inherent diversity. Over-  
 597 all, the curriculum provides a trade-off between  
 598 diversity and training stability.

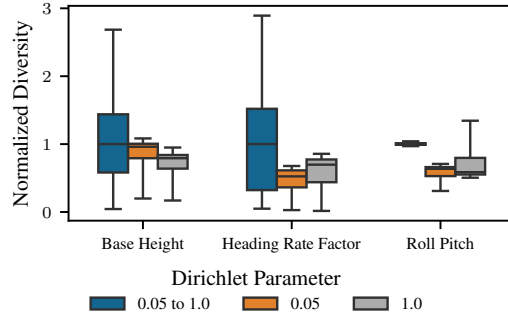


Figure 9: Effect of dirichlet parameter on diversity on three factors