DMAP: a Distributed Morphological Attention Policy for learning to locomote with a changing body

Anonymous Author(s)
Affiliation
Address
email

Abstract

Reinforcement learning typically seeks to learn control policies in stable environments. Yet, real world scenarios require continuous adaptation. In particular, learning to locomote when the length and the thickness of different body parts vary is challenging, as the policy is required to adapt to the current configuration to successfully balance and advance the agent. We study this problem in four classical continuous control environments, augmented with morphological perturbations. We show that a control policy based on the proprioceptive state performs poorly with highly variable body configurations, while an (oracle) agent with access to a learned encoding of the perturbation performs significantly better. We introduce DMAP, a biologically-inspired, attention-based policy network architecture. It combines a distributed policy, with individual controllers for each joint, and an attention mechanism, to dynamically gate sensory information from different body parts. DMAP can be trained end-to-end in all the considered environments and perturbation intensities, overall matching or surpassing the performance of an oracle agent with access to the morphology information. Thus DMAP, implementing principles of control drawn from the biological world, provides a strong inductive bias for learning challenging sensorimotor tasks. Overall, our work corroborates the power of these principles in challenging locomotion tasks.

1 Introduction

Animals and humans are highly adaptive to changes in the external environment and in their own body. For instance, animals can, after an early onset, locomote throughout development, when their body size and weight change dramatically. They can also robustly deal with irregular surfaces, and load changes [1, 2, 3, 4]. Here, we focus on the problem of learning to walk when the body is subject to morphological perturbations, consisting in the change of length and thickness of its parts. For this purpose, we adapted four different locomotion environments from OpenAI Gym [5] (Ant, Half Cheetah, Walker, Hopper), in which robots have to learn how to walk as fast as possible. In our adaptive setting, the agent's body parameters are sampled from a morphological perturbation space at the beginning of each episode and we sought to find reinforcement learning policies that could deal with these varying bodies.

We hypothesize that principles of the sensorimotor system provide a strong inductive bias to robustly learn such a policy. Specifically, we built on three well-known features: **1) Independent low-level processing:** Somatosensory inputs are first processed locally (e.g., by temporal filtering) before they are integrated across more distal body parts and represented in an orderly way [6, 7, 8]. **2)**

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

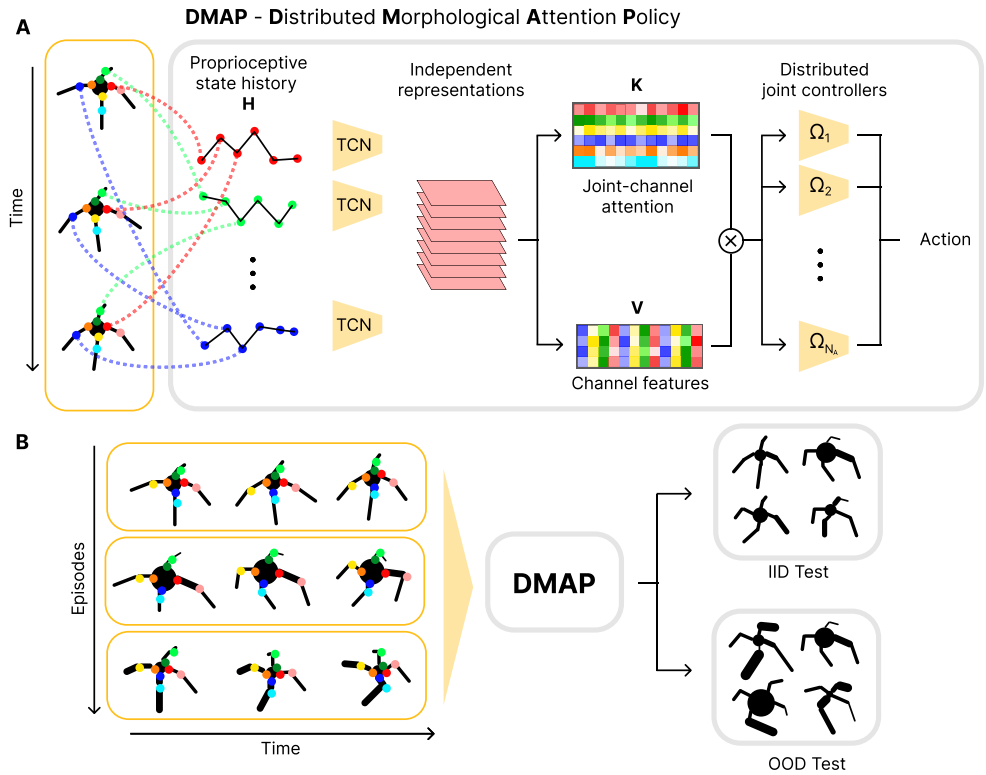

Figure 1: **A**. Overview of the architecture of DMAP. Each component of the state history is processed independently by the same temporal convolutional network. The extracted features are transformed into the value encoding matrix and the attention weights of each channel and action component. The product of these matrices results in a joint-specific morphology encoding, which is the convex combination of the value encoding vectors, weighted by their attention score. Independent fully-connected networks control a single joint, based on the current proprioceptive state and the morphology encoding **B**. DMAP is trained for several episodes with randomly sampled perturbed bodies within the training distribution. At test time, the zero-shot performance of the model is assessed on a fixed set of body configurations, sampled either within (IID) or outside (OOD) of the training distribution.

Distributed control: Different body parts are controlled in a distributed fashion [7, 8, 4]. **3) Gated connectivity:** Sensory information is dynamically gated in a task dependent manner [9], also beyond somatosensation [10], and integrated with action and value signals in the brain [11, 8, 12].

Hence, we propose the Distributed Morphological Attention Policy (DMAP) architecture (Figure 1). We show that it can robustly learn to locomote with changing bodies without access to their morphological parameters. To assess the performance of this architecture, we compare it to several other agents. Experiments reveal that an agent with access only to the proprioceptive state, consisting of its joint angles and velocities (*Simple*), fails to learn an effective locomotion strategy for a large variety of body configurations. In contrast, augmenting the observation with the current body shape leads to a strong ceiling performance (*Oracle*). Importantly, for all different environments, DMAP is as good as Oracle.

We also compare DMAP to a recently proposed two-step training technique called rapid motor adaptation (*RMA*) [13]. This algorithm trains a temporal convolutional network to infer a context encoding from a short history of proprioceptive state, imitating the Oracle's morphology encoder. While RMA can also reach Oracle performance, we show that end-to-end training of such network does not succeed for more complicated environments (Ant and Walker). In contrast, DMAP's policy can be trained end-to-end without access to the morphological parameters and reaches or surpasses Oracle performance for all four environments (Ant, Half Cheetah, Walker, Hopper). This suggests

that independent low-level processing, distributed control and dynamic gating mechanisms provide a strong inductive bias for sensorimotor control. Finally, we analyze the information gating mechanism of DMAP, which establishes pose-dependent sensorimotor pathways. The low dimensional embedding of the gating mechanism exhibits rotational dynamics, robustly across different morphologies.

2 Related work

Benchmarks for zero-shot adaptation. The setup of the experiments described in this paper can be framed as a zero-shot adaptation problem, as the agents are evaluated starting from the first moment in which they interact with an unseen context [14]. While classic RL benchmarks, such as Arcade [15] and rllab [16], assess the ability of an agent to maximize the cumulative reward in the training environment, more recent ones introduce perturbations in the dynamics [17, 18], in the observation function [19, 20] or in the reward function [21] at test time. For a more complete list, we refer to [14]. Mann et al. showed that the performance of RL agents under morphological distributional shifts correlates with the ability of internal models to generalize [22]. We took their design of environments as an inspiration for the design of morphological changes in our work.

Context encoding. Algorithms which enable learning in changing environments are important for the sim-to-real transfer problem, where domain randomization represents a successful strategy to bridge the reality gap [23, 24, 25]. A notable approach to solving the zero-shot transfer problem consists in encoding the context into a hidden state, e.g., by online system identification [26], or with a recurrent neural network (RNN) [16, 27]. However, while off-policy continuous control algorithms such as TD3 [28] and SAC [29] are the current state of the art among model-free RL algorithms, both in sample efficiency and final performance [30], training a recurrent neural network off-policy presents relevant technical challenges [31]. As the focus of our work is on the development of a network architecture to handle a changing body, we choose to follow the more straightforward approach to system identification proposed in [13], where the authors show that a context encoding is successfully extracted from a short sequence of past states and actions through temporal convolutions.

Distributed control. An agent can learn to generalize to multiple body shapes by using modular policies to control each joint independently, e.g., treating each joint [32] or groups of joints [33] as a node of the graph. For example, *Nervenet* [34] achieves state-of-the-art performance on standard RL benchmarks by leveraging different policies for each joint type. However, it cannot generalize to different designs in a zero-shot manner without fine-tuning. On the other hand, Shared Modular Policies (SMP) [35] employs the same policy for each joint, enabling an agent to deal with variable observation and action space sizes. While these works study the problem of controlling bodies with a variable number of segments, we focus on perturbations which affect some morphological parameters, but without changing the body structure.

3 A biology-inspired architecture to handle morphological perturbations

Adaptation to morphological perturbations requires an agent to rapidly identify which joints are affected and what compensatory torques are required. We propose a Distributed Morphological Attention Policy (DMAP). With DMAP (Figure 1), we seek to facilitate this process by promoting the formation of communication pathways from individual body parts to the joint controllers. The architecture is designed according to the previously mentioned principles of (biological) sensorimotor control, implemented in the following way:

Independent low-level processing. Each proprioceptive channel is first processed independently, in order to obtain a channel-specific representation. This is inspired by the processing in the proprioceptive pathway [7, 8, 9]. **Distributed control.** Independent networks control each of the agent's joint, by outputting one single action scalar (corresponding to a joint's torque). This is inspired by the architecture of the spinal cord and the motor homunculus [8, 6, 2]. **Gated connectivity.** An attention mechanism assigns the connection weight between the control policy of a joint and

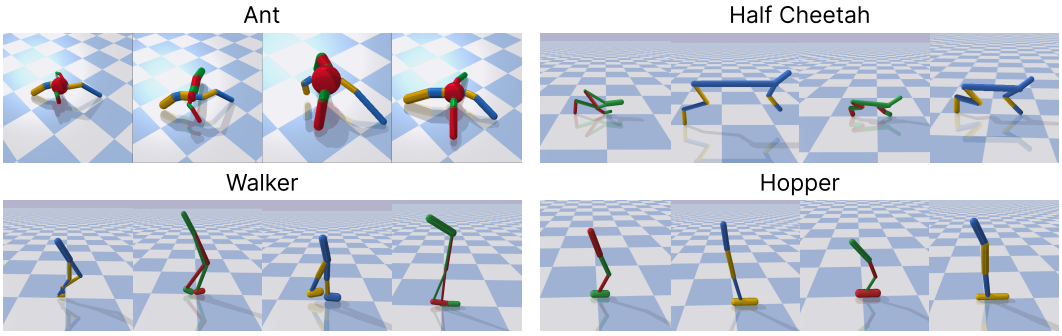

Figure 2: Example of perturbed body configurations for the four agents, sampled from the $\sigma = 0.5$ perturbation level. The segments composing the body vary in thickness and length.

the features of each component of the proprioceptive state. This is inspired by dynamic gating mechanisms in the brain [9, 10, 11].

3.1 Attention-based feature encoder

A feature encoder receives in input a history of T proprioceptive states and actions $((\mathbf{s}^{(t-T)}, \mathbf{a}^{(t-T)}), ..., (\mathbf{s}^{(t-1)}, \mathbf{a}^{(t-1)}))$, which we denote as $H^{(t)} \in \mathbb{R}^{(N_S+N_A)\times T}$, N_S and N_A being the state and action space size, respectively. Each row $\mathbf{h}_i^{(t)} \in \mathbb{R}^T$ of $H^{(t)}$ is processed independently by the same Temporal Convolutional Network (TCN), which maps the time series of each state and action component into the vectors $\mathbf{k}_i^{(t)} \in \mathbb{R}^{N_A}$, representing the attention of each joint towards that input channel i, and $\mathbf{v}_i^{(t)} \in \mathbb{R}^{N_V}$, a value encoding vector of size N_V. The vectors $\mathbf{k}_i^{(t)}$ and $\mathbf{v}_i^{(t)}$ from different channels are stacked to form the matrices $K^{(t)} \in \mathbb{R}^{(N_S+N_A)\times N_A}$ and $V^{(t)} \in \mathbb{R}^{(N_S+N_A)\times N_V}$ The context encoding vectors for each joint controller are the rows $\mathbf{e}^{(t)T}$ of the matrix $E^{(t)} = K^{(t)T}V^{(t)}$, which are convex combinations of the features extracted from each sensory modality, weighted by their attention score (Section A.1). In this way, each controller can focus on the relevant morphological information carried by specific sensory modalities, while ignoring the irrelevant ones. Furthermore, the attention matrix is conditioned to the recent transition history, making it adaptable both across episodes (and thus morphological states) and within a single episode.

3.2 Distributed joint controllers

Differently from standard policy networks, we adopt independent controllers Ω_i, $i \in 1, .., N_A$, for each joint, implemented as distinct fully-connected networks. Each network outputs the element $a_i^{(t)}$ of the action vector $\mathbf{a}^{(t)}$ based on the current state $\mathbf{s}^{(t)}$, the previous action $\mathbf{a}^{(t-1)}$ and the context encoding $\mathbf{e}_i^{(t)}$. Distributed action selection is a natural way to enable the emergence of sensorimotor pathways from the sensory data of different body parts to the controller of a joint. As one controller needs to predict one single action value, we show in our experiments that smaller fully-connected networks are sufficient to successfully solve the considered tasks, limiting the complexity overhead stemming from distributed control (Section A.2). Qualitatively, independent policy networks can more easily adapt their behavior when a morphological perturbation affects a localized area of the body, leaving the control policy of the other networks unaltered.

4 Morphological perturbation environments

We consider four locomotion environments of the PyBullet physics simulator [36]: Hopper, Walker, Ant and Half-Cheetah, which are standard benchmarks for continuous control reinforcement learning algorithms [37]. To test the ability of an agent to learn locomotion with a changing body, we introduce a set of morphological perturbations for each of them. The perturbations consist in modifying the

size and the length of one or more body parts (Section A.3). The parameter σ controls the size of the perturbation space C_σ, limiting the intensity of the perturbation between $-\sigma$ and σ. The perturbation is defined as the relative deviation of a morphological parameter from its original value. For example, a perturbation of 0.5 applied to a leg thickness will make it 1.5 times thicker. This construction results in widely varying body shapes for each of the four environments (Figure 2).

As in the unperturbed environments, the training is organized in episodes of at most 1000 time steps. At the beginning of each episode, bodily parameters are sampled uniformly at random in the space of admissible perturbations. This setup defines a Contextual Markov Decision Process (CMDP) [14] $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \gamma, C_\sigma \rangle$, determined by the state space $\mathcal{S}$, the observation function $\mathcal{O}$, the action space $\mathcal{A}$, the transition dynamics $\mathcal{T}$, the reward function $\mathcal{R}$, the discount factor γ and the context C_σ. The observation only includes proprioceptive information, in the form of angular position and velocity of the joints, besides other non-visual quantities (Section A.3). It does not contain explicit information about the current context, as identical observations can come from bodies with different parameters. On the other hand, the context influences the transition dynamics $p(\mathbf{s}_{t+1} | \, \mathbf{s}_t, \mathbf{a}_t, C_\sigma)$, modifying the effect of an action depending on the body configuration. This condition of partial observability makes the locomotion problem considerably harder. However, as the parameters defining the context are constant throughout one episode, we show that a control policy can adapt its locomotion strategy to it.

5 Experiments

We train all the agents with Soft Actor Critic (SAC) [29], using the open source library RLLib [38]. The experiments have been run on a local CPU cluster, for a total of approx. 100 000 cpu-hours. We use similar hyperparameters to [30] (Section A.3), which provide state-of-the-art results in the base versions of the four PyBullet environments considered in this work, and we run every experiment for 5 random seeds. We found that the training of the critic network benefits from augmenting the input observation with the current morphological perturbation. Therefore, we included it for all policy architectures, except for the Simple agent. This addition does not pose any limitation to the deployment of the proposed algorithms, as the policy network alone is necessary at that stage.

The experiments involve five different policy networks, which differ in architecture, required input data and training procedure (Table 1).

Table 1: The five policy networks analyzed in the experiments. MLP: multi-layer perceptron. TCN: temporal-convolutional network. 2-step: imitation of the Oracle morphology encoder. P: proprioceptive state. M: morphology information. H: transition history.

	Simple	Oracle	RMA	TCN	DMAP
Encoder	-	MLP	TCN	TCN	Attention
Policy	MLP	MLP	MLP	MLP	Distributed MLP
Input	P	P + M	P + H	P + H	P + H
Training	End-to-end	End-to-end	2-step	End-to-end	End-to-end

All the results refer to the cumulative reward obtained by zero-shot adaptation in 100 environments characterized by body configurations not observed during the training. The rewards in bold are significantly higher according to a paired t-test (p-value < 0.001). We denote the tests as IID if the 100 body configurations were extracted from the morphological space used during the training ($\sigma_{train} \geq \sigma_{test}$), otherwise we denote them as OOD ($\sigma_{train} < \sigma_{test}$).

5.1 A morphology encoding boosts learning in strongly perturbed environments

To assess whether an agent can benefit from an encoding of its morphology when learning a locomotion policy, we compare the performance achieved when providing only the proprioceptive state (Simple) and when augmenting it with the raw perturbation (Oracle). Specifically, this Oracle follows the architecture of Kumar et al. [13]. It extracts a morphology encoding from the raw perturbation

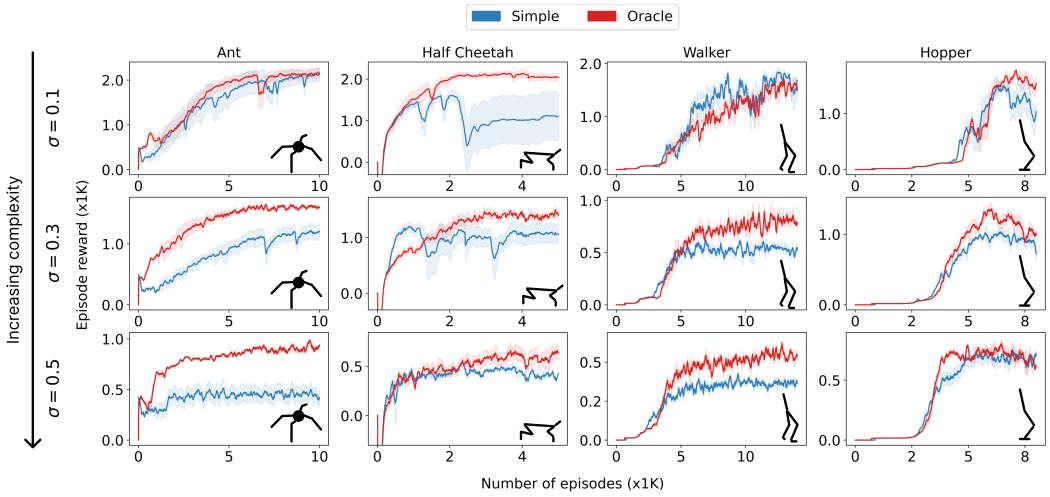

Figure 3: Learning curves of Simple and Oracle (mean ± SEM across 5 random seeds). While the Simple performs competitively in low perturbation environments ($\sigma = 0.1$), Oracle learns better at higher perturbation levels.

Table 2: IID performance of RMA and Oracle, mean episode reward $\pm$ SEM.

Env	Ant		Half Cheetah		Walker		Hopper	
Algo	RMA	Oracle	RMA	Oracle	RMA	Oracle	RMA	Oracle
$\sigma = 0.1$	2138 ± 16	2148 ± 16	2197 ± 16	2203 ± 17	1750 ± 28	1780 ± 26	1859 ± 19	1859 ± 18
$\sigma = 0.3$	1700 ± 17	1723 ± 18	1402 ± 27	$\mathbf{1469 \pm 24}$	836 ± 23	$\mathbf{908 \pm 23}$	1267 ± 27	1296 ± 26
$\sigma = 0.5$	966 ± 16	974 ± 18	595 ± 26	$\mathbf{668 \pm 24}$	579 ± 17	$\mathbf{625 \pm 17}$	730 ± 21	$\mathbf{919 \pm 18}$

vector and concatenates it to the observation vector, thus augmenting the input of the policy network and breaking the partial observability of the environment. We found that, while in mildly perturbed environments Simple performs competitively with Oracle, even outperforming it in Walker, learning a control policy solely based on the proprioceptive state becomes difficult in the larger perturbation space (Figure 3 and Section A.4). In all our experiments with $\sigma \in \{0.3, 0.5\}$ a morphology representation proved useful, which demonstrates, as expected, that learning an environment encoding is a viable option to deal with strongly perturbed environments.

5.2 The morphology encoding can be regressed from experience

The morphology parameters C influence the transition dynamics $p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t, C)$ by changing the physical properties of the body. Therefore, they are implicitly represented in a sequence of transitions, from which they might be inferred. To test this possibility, we apply the RMA algorithm [13] to our morphology-perturbed environments. It leverages the latent representation of the body generated by a trained Oracle, which has access to the contextual information, to train a TCN network to regress the context encoding from a short history of T proprioceptive states and actions $H^{(t)} = ((\mathbf{s}^{(t-T)}, \mathbf{a}^{(t-T)}), ..., (\mathbf{s}^{(t-1)}, \mathbf{a}^{(t-1)}))$. RMA achieves very similar average reward to the Oracle (Table 2), from which we can deduce that a sequence of 30 transitions (approx. 0.5 s in our experiments), only including previous proprioceptive states and actions, is sufficient to extract an embedding of the context, proving this two-phase adaptation procedure viable for morphological perturbations.

5.3 DMAP enables end-to-end training of a morphology encoding network

It is unclear whether a TCN network requires a two-step training through imitation to learn how to extract a morphology encoding from a sequence of transitions. In fact, if such encoding helps maximizing the reward, the TCN might successfully learn it end-to-end with reinforcement learning.

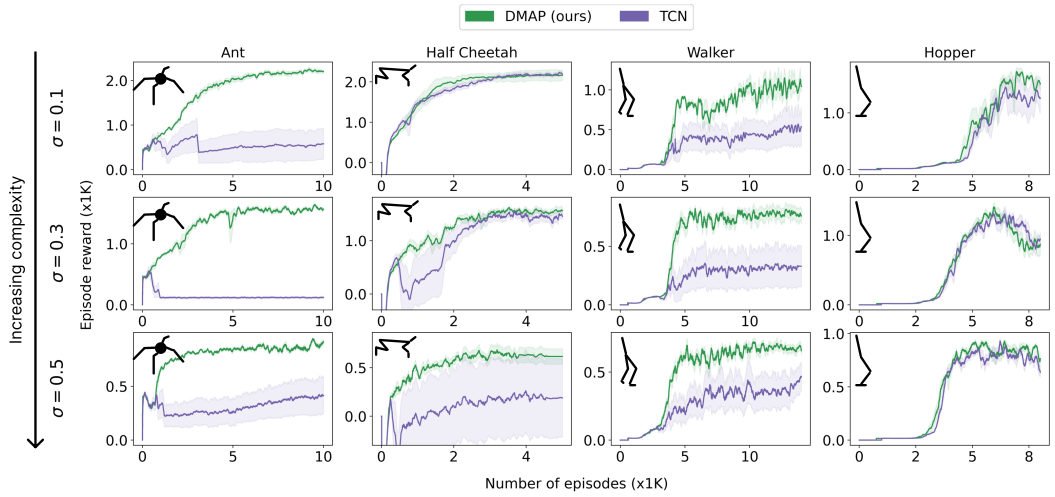

Figure 4: Learning curves of DMAP and TCN, the network architecture used in RMA, but trained end-to-end (mean ± SEM across 5 random seeds). The performance gap is particularly clear for Ant and Walker, in which the training of TCN is characterized by high variance and overall low reward.

In our experiments, however, while in Hopper and in Half Cheetah an end-to-end training of a TCN encoder proves successful, in Ant and Walker it encounters strong instability and inconsistency across the random seeds (Figure 4). Surprisingly, in these two latter environments the addition of the TCN encoder is even harmful for the overall performance, as Simple achieves stronger results in the same environments. Further analysis (Section A.4) reveals that, for certain random seeds, the training process is disrupted before the agent can learn how to walk. We hypothesize that this phenomenon is due to the difficulty of extracting a meaningful encoding to directly optimize the reward without a suitable inductive bias, with the effect of the encoder acting as a confounder for the policy network. Indeed, replacing TCN with DMAP leads to better stability during the training (Figure 4) and to a general performance improvement, particularly in Ant and Walker (Section A.4). Differently from TCN, DMAP performs on par with Oracle in all the considered environments, with the exception of Walker with $\sigma = 0.1$. Oracle and DMAP also perform similarly when tested OOD, with a slight advantage for DMAP (Figure 5 and Section A.4).

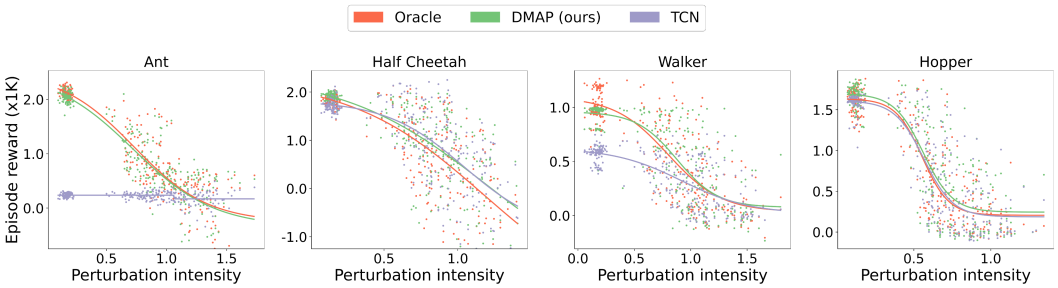

Figure 5: Reward obtained by Oracle, TCN and DMAP, trained in the $\sigma = 0.3$ environments, on a wide range of perturbations (both IID and OOD). Each point represents the average reward across 5 random seeds for one body configuration, while the perturbation intensity is the eucledian norm of the perturbation vector. The curves are obtained by fitting a sigmoid function. Despite the larger perturbations falling outside of the training distribution, DMAP slightly outperforms Oracle, which receives the exact perturbation information.

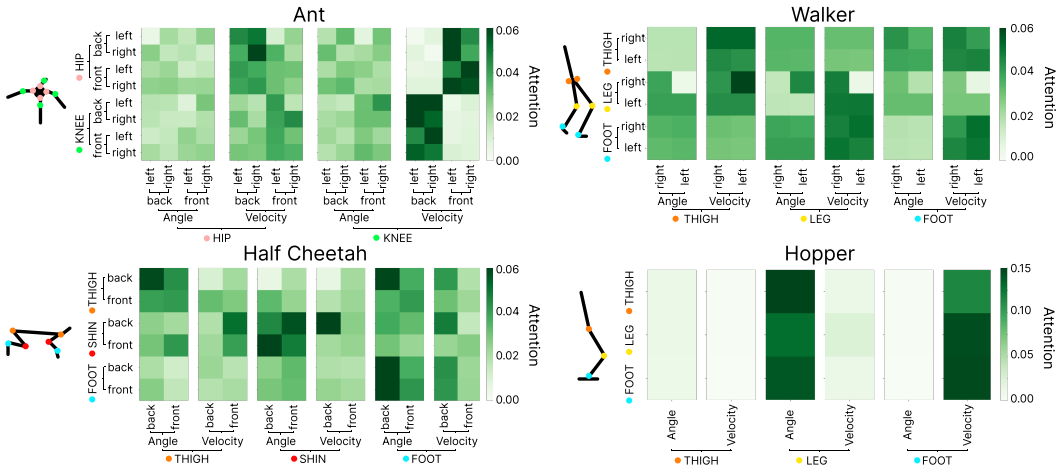

Figure 6: Attention matrices K, averaged across time and IID test body perturbations, represented as a heatmap for each agent, trained with $\sigma = 0.1$. We can observe several interpretable patterns, e.g., the bilateral symmetry in the Ant, as well as a front-back asymmetry (due to the fixed running direction). Also in the Ant, the hip controllers mainly look at the angular velocity of the front knees and of the back hips, while the ankle controllers do the opposite. In the Half Cheetah, too, the attention map develops a front-back asymmetry, with a preference for the input angular position, and each controller mainly attends to the state of the joint it controls.

5.4 Ablation: The morphology encoding is key in statically unstable environments

To isolate the contribution of the morphology encoding on the performance of DMAP, we performed test-time ablation by evaluating the trained DMAP distributed policies, while removing the morphology encoding. We found a performance drop for all the environments and perturbation levels (Table 3). It is especially severe in Hopper and Walker (approx. 50%), where the controller with ablated encoding fails to stabilize the agents, causing the early termination condition to be met sooner and thus yielding lower episode reward. This suggests that the attention mechanism is fundamental for the dynamic stability of these agents.

Table 3: IID performance of DMAP and non-encoding (ne) DMAP, mean episode reward $\pm$ SEM.

Env	Ant		Half Cheetah		Walker		Hopper	
Algo	DMAP	DMAP-ne	DMAP	DMAP-ne	DMAP	DMAP-ne	DMAP	DMAP-ne
$\sigma = 0.1$	$\mathbf{2240 \pm 11}$	1966 ± 18	$\mathbf{2261 \pm 11}$	1493 ± 15	$\mathbf{1229 \pm 24}$	337 ± 16	$\mathbf{1842 \pm 18}$	984 ± 33
$\sigma = 0.3$	$\mathbf{1623 \pm 19}$	1542 ± 19	$\mathbf{1577 \pm 22}$	1132 ± 29	$\mathbf{893 \pm 11}$	470 ± 16	$\mathbf{1316 \pm 24}$	748 ± 23
$\sigma = 0.5$	$\mathbf{960 \pm 14}$	881 ± 12	$\mathbf{669 \pm 23}$	507 ± 29	$\mathbf{743 \pm 15}$	341 ± 17	$\mathbf{953 \pm 15}$	482 ± 20

5.5 Attention gates the sensory input through information pathways

We hypothesized that DMAP enables the agent to learn which components of the input are relevant for each joint's controller. Next, we verify that such pathways emerge by analyzing the values of the attention matrix $K^{(t)}$, both in the *steady* and *dynamic* condition.

The steady attention matrix K represents the strength of each input to output (i.e., sensorimotor) connection. We found morphology-specific patterns by averaging elements across time and episodes (Figure 6). As in the spinal cord, the controllers typically attend to the sensory input of mechanically linked parts (e.g., foot controller to foot velocity, etc.). Yet, this topography is not universal: for instance in the hopper, all controllers focus on the leg angle and the foot velocity. Furthermore, the attention patterns seem to reflect the bilateral symmetry of Ant and Walker and a front-back asymmetry due to the directionality of the locomotion task.

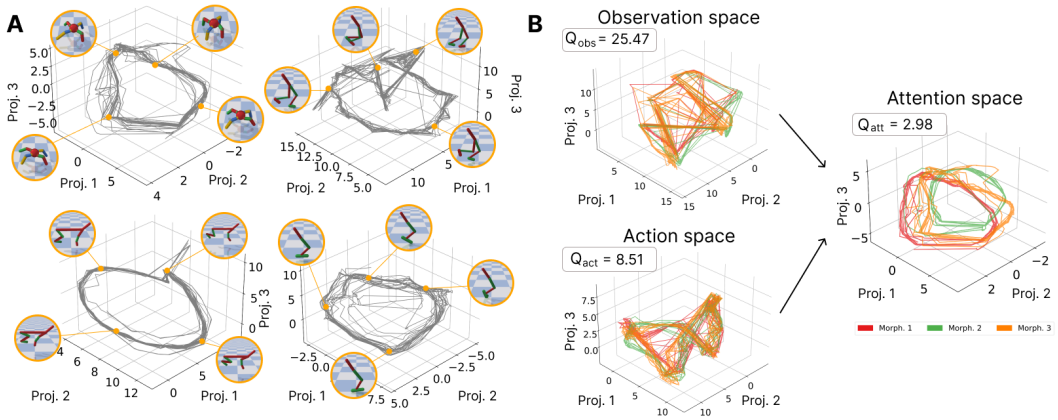

Figure 7: **A.** UMAP embedding of the attention matrix K evolving during a single episode for each agent. The attentional dynamics of K are cyclic, consistently with the periodic gait of the agents. Thus, the connectivity patterns between input channels and joint controllers change throughout the different phases of the gait. **B.** UMAP embeddings of the input observations $\mathcal{O}^{(t)} \in \mathbb{R}^{N_S \times T}$, input actions $\mathcal{A}^{(t)} \in \mathbb{R}^{N_A \times T}$ and attention matrix K for 3 different morphologies of the Ant. The change of body configuration causes a shift of the attention dynamics in the low-dimensional space. The trajectories of the attention embedding seem less tangled than the corresponding input. We quantified this with a tangling metric, Q (averaged over time and morphologies; large values correspond to highly tangled trajectories). We refer to Section A.5 for the technical details.

To analyze the temporal evolution of attention, we project the attention matrix $K^{(t)}$ onto a 3-dimensional space learned with UMAP [39] (Figure 7). Depending on the phase of the gait, the attention matrix $K^{(t)}$ strengthens or weakens specific sensorimotor pathways (Figure 7 A). The graphs reveal a rotational dynamics of the gain modulation typical of biological, legged locomotion [40, 41, 9]. Interestingly, the attention dynamics seem significantly less tangled than that of the proprioceptive and action input (Figure 7 B). We quantified this visual impression across the different environments and found a robust effect (Section A.5). We speculate that this untangling mechanism might contribute to the robustness of the policy to morphological perturbations.

6 Discussion

We proposed DMAP, a novel architecture that integrates computational principles from the sensorimotor system. We find that DMAP enables the end-to-end training of a RL policy in environments with changing body parameters, without having access to the perturbation information. It matches or even surpasses the performance of an Oracle when tested in a zero-shot manner on unseen body configurations. Interestingly, the attention mechanism of DMAP solves this challenging task by establishing gait-phase-dependent gain modulation, robust across morphologies. Notably, gains in the nervous system also change in a phase-dependent way [40, 41, 9].

Limitations and broader impact. One characteristic of our current architecture design is that each joint controller observes the full proprioceptive state. A future direction will be to limit the input to the local sensory data. Such design, separating low-latency sensorimotor connection (reflexes) and slower connections to distal body parts, would be closer to the locomotor system of the animals [2]. Furthermore, in our experiments, we have used an oracle critic to learn an estimate of the cumulative reward and guide the training of the policy. However, a suitable inductive bias, possibly inspired by the neural circuits responsible for reward prediction error, could remove this limitation.

Our research aims to facilitate the training of control policies by transferring insights from neuroscience to artificial intelligence. Powerful reinforcement learning algorithms might find application in robotics, potentially enabling, in the long term, a large-scale deployment of autonomous agents. Automation comes with great opportunities, and ethical concerns [42].

References

[1] Andrew Biewener and Sheila Patek. *Animal locomotion*. Oxford University Press, 2018.

[2] Michael H Dickinson, Claire T Farley, Robert J Full, MAR Koehl, Rodger Kram, and Steven Lehman. How animals move: an integrative view. *science*, 288(5463):100–106, 2000.

[3] Ana I Gonçalves, Jacob A Zavatone-Veth, Megan R Carey, and Damon A Clark. Parallel locomotor control strategies in mice and flies. *Current Opinion in Neurobiology*, 73:102516, 2022.

[4] Roberto Leiras, Jared M Cregg, and Ole Kiehn. Brainstem circuits for locomotion. *Annual review of neuroscience*, 45, 2022.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[6] Wilder Penfield and Edwin Boldrey. Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation. *Brain*, 60(4):389–443, 1937.

[7] Peter BC Matthews. Muscle spindles: their messages and their fusimotor supply. *Handbook of physiology: I. The nervous system. American Physiological Society.[AGF]*, 1981.

[8] Reza Shadmehr and John W Krakauer. A computational neuroanatomy for motor control. *Experimental brain research*, 185(3):359–381, 2008.

[9] Eiman Azim and Kazuhiko Seki. Gain control in the sensorimotor system. *Current opinion in physiology*, 8:177–187, 2019.

[10] Grace W Lindsay. Attention in psychology, neuroscience, and machine learning. *Frontiers in computational neuroscience*, page 29, 2020.

[11] Angelo Maravita and Atsushi Iriki. Tools for the body (schema). *Trends in cognitive sciences*, 8 (2):79–86, 2004.

[12] Daniel M Wolpert and J Randall Flanagan. Computations underlying sensorimotor learning. *Current opinion in neurobiology*, 37:7–11, 2016.

[13] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[14] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*, 2021.

[15] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[16] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[17] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[18] Chenyang Zhao, Olivier Sigaud, Freek Stulp, and Timothy M Hospedales. Investigating generalisation in continuous deep reinforcement learning. *arXiv preprint arXiv:1902.07015*, 2019.

[19] Matthew Crosby, Benjamin Beyret, Murray Shanahan, José Hernández-Orallo, Lucy Cheke, and Marta Halina. The animal-ai testbed and competition. pages 164–176, 2020.

[20] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. pages 2048–2056, 2020.

[21] Meire Fortunato, Melissa Tan, Ryan Faulkner, Steven Hansen, Adrià Puigdomènech Badia, Gavin Buttimore, Charles Deck, Joel Z Leibo, and Charles Blundell. Generalization of reinforcement learners with working and episodic memory. *Advances in Neural Information Processing Systems*, 32, 2019.

[22] Khushdeep Singh Mann, Steffen Schneider, Alberto Chiappa, Jin Hwa Lee, Matthias Bethge, Alexander Mathis, and Mackenzie W Mathis. Out-of-distribution generalization of internal models is correlated with reward. 2021.

[23] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. pages 23–30, 2017.

[24] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[25] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. pages 3803–3810, 2018.

[26] Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

[27] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[28] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. pages 1587–1596, 2018.

[29] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. pages 1861–1870, 2018.

[30] Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. pages 1634–1644, 2022.

[31] Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. 2018.

[32] Deepak Pathak, Christopher Lu, Trevor Darrell, Phillip Isola, and Alexei A Efros. Learning to control self-assembling morphologies: a study of generalization via modularity. *Advances in Neural Information Processing Systems*, 32, 2019.

[33] Julian Whitman, Matthew Travers, and Howie Choset. Learning modular robot control policies. *arXiv preprint arXiv:2105.10049*, 2021.

[34] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. 2018.

[35] Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. pages 4455–4464, 2020.

[36] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

[37] Antonin Raffin. Rl baselines zoo. https://github.com/araffin/rl-baselines-zoo, 2018.

[38] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. pages 3053–3062, 2018.

[39] Leland McInnes, John Healy, and James Melville. Umap: uniform manifold approximation and projection for dimension reduction. 2020.

[40] Philip Holmes, Robert J Full, Dan Koditschek, and John Guckenheimer. The dynamics of legged locomotion: Models, analyses, and challenges. *SIAM review*, 48(2):207–304, 2006.

[41] Wouter Hoogkamer, Frank Van Calenbergh, Stephan P Swinnen, and Jacques Duysens. Cutaneous reflex modulation and self-induced reflex attenuation in cerebellar patients. *Journal of Neurophysiology*, 113(3):915–924, 2015.

[42] Lambèr Royakkers and Rinie van Est. A literature review on new robotics: automation from love to war. *International journal of social robotics*, 7(5):549–570, 2015.

[43] Abigail A Russo, Sean R Bittner, Sean M Perkins, Jeffrey S Seely, Brian M London, Antonio H Lara, Andrew Miri, Najja J Marshall, Adam Kohn, Thomas M Jessell, et al. Motor cortex embeds muscle-like commands in an untangled population response. *Neuron*, 97(4):953–966, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6 for a discussion on the scope and limitations of this work.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In Section 6 we discuss the borader impact of our research.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] In Section 4 we introduce the experimental setup, specifying which information can be accessed by the agents and what is not available.
   (b) Did you include complete proofs of all theoretical results? [N/A] The results of this work are computational.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code is included in the supplemental material, with instructions on how to train the agents to reproduce the main results of the paper.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 for a description of the training setup, Section A.3 for the details about the environments and Section A.2 for the hypeprarameters of SAC and of the architecture of the policies.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All the tables and the graphs of the learning curves report the mean value and the standard error of the mean, obtained by running each experiment with 5 random seeds.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] In Section 5 we cite the two main libraries used in our work: PyBullet for the physics simulation and RLLib for the reinforcement learning experiments.
   (b) Did you mention the license of the assets? [Yes] In Section 4 and Section 5 we specify that the two libraries are open source.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A] We included our source code with license in the supplemental material
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] No data from external sources was used for this study
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] No data from external sources was used for this study
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] No human subject was involved in this study.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] No human subject was involved in this study.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] No human subject was involved in this study.

A Appendix

A.1 DMAP detailed architecture

Distributed Morphological Attention Policy (DMAP) consists of three main components: independent low-level proprioceptive processing, an attention-based morphology encoder and a distributed joint controller (8).

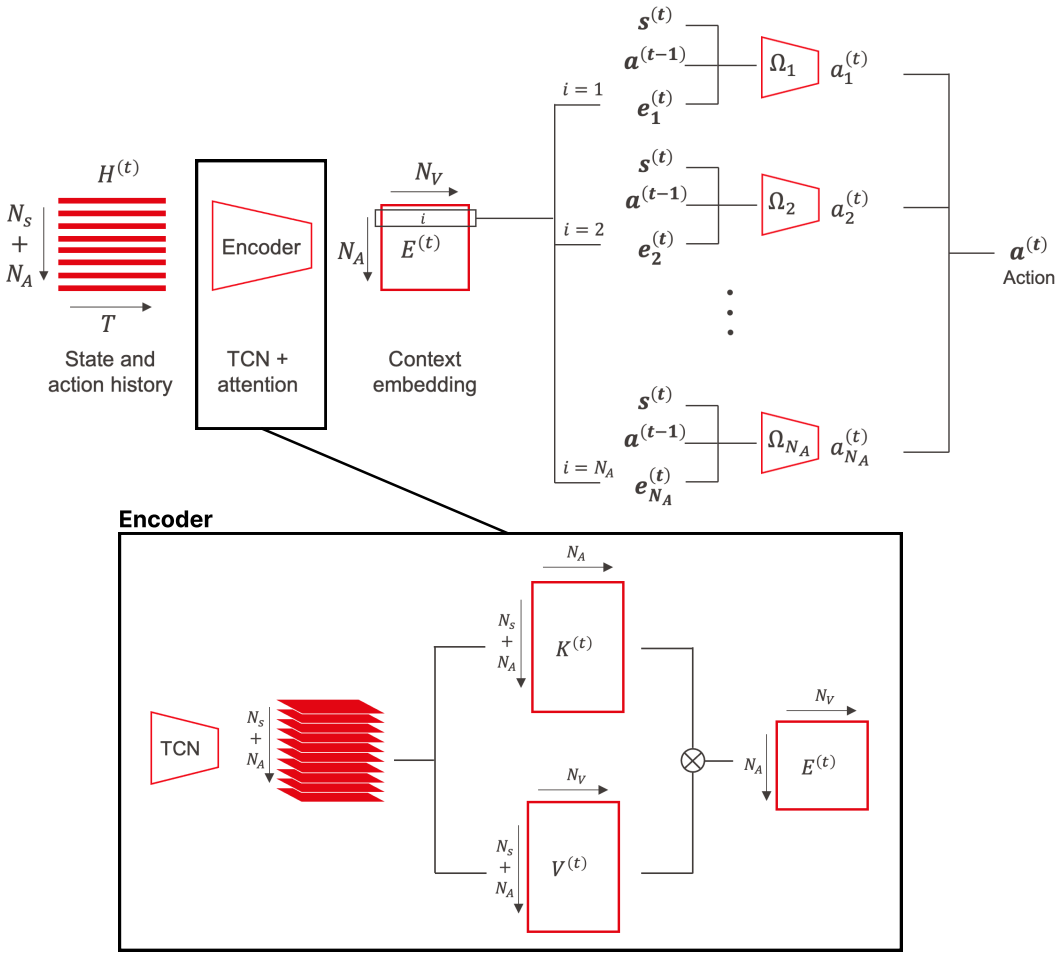

Figure 8: Detailed architecture of DMAP. After applying convolutions to the each input channel along the temporal dimension, the encoder module extracts: a key matrix $K^{(t)}$, whose elements represent the attention score of an action component towards an input channel; a value matrix $V^{(t)}$, whose rows encode the information from each input channel of the transition history. A softmax operation is applied along the columns of the $K^{(t)}$ matrix, so that the total attention of one action component sums to 1. The action component-specific encoding vectors $\mathbf{e}^{(t)}$ are obtained by matrix multiplication: $K^{(t)T}V^{(t)}$. The learned context embedding is then concatenated with the previous action $\mathbf{a}^{(t-1)}$ and the current proprioceptive observation $\mathbf{s}^{(t)}$. Together, those triples are the inputs to N_A fully-connected networks, one for each action component. They output the torque applied to one joint of the agent.

A.2 Hyperparameters

Table 4 lists the hyperparameters of SAC used for our experiments. They are the same for all network architectures, without specific fine tuning. We also list the parameters of the neural networks of the different agents.

Table 4: Parameters of SAC for each of the 4 environments

Environment Algorithms	Parameters	Ant	Half Cheetah	Walker	Hopper
All	Action normalization	Yes	Yes	Yes	Yes
	Num TD steps	1	1	1	1
	Buffer size	1 000 000	300 000	300 000	300 000
	Prioritized experience replay	No	Yes	Yes	Yes
	Actor learning rate	0.0003	0.0003	0.0003	0.0003
	Critic learning rate	0.0003	0.0003	0.0003	0.0003
	Entropy learning rate	0.0003	0.0003	0.0003	0.0003
	Discount factor γ	0.995	0.995	0.995	0.995
	Minibatch size	256	256	256	256
Simple	Policy hiddens	[256, 256]	[256, 256]	[256, 256]	[256, 256]
	Q hiddens	[256, 256]	[256, 256]	[256, 256]	[256, 256]
	Activation	ReLU	ReLU	ReLU	ReLU
Oracle	Policy encoder hiddens	[256, 128]	[256, 128]	[256, 128]	[256, 128]
	Encoding size	4	4	4	4
	Policy hiddens	[128, 128]	[128, 128]	[128, 128]	[128, 128]
	Q encoder hiddens	[256, 128]	[256, 128]	[256, 128]	[256, 128]
	Q hiddens	[128, 128]	[128, 128]	[128, 128]	[128, 128]
	Activation	ReLU	ReLU	ReLU	ReLU
TCN	Num TCN layers	3	3	3	3
	Num filters per layer	[32, 32, 32]	[32, 32, 32]	[32, 32, 32]	[32, 32, 32]
	Kernel size per layer	[5, 3, 3]	[5, 3, 3]	[5, 3, 3]	[5, 3, 3]
	Stride per layer	[4, 1, 1]	[4, 1, 1]	[4, 1, 1]	[4, 1, 1]
	Policy encoder hiddens	[128, 32]	[128, 32]	[128, 32]	[128, 32]
	Encoding size	4	4	4	4
	Policy hiddens	[128, 128]	[128, 128]	[128, 128]	[128, 128]
	Q encoder hiddens	[256, 128]	[256, 128]	[256, 128]	[256, 128]
	Q hiddens	[128, 128]	[128, 128]	[128, 128]	[128, 128]
	Activation	ReLU	ReLU	ReLU	ReLU
DMAP	Num TCN layers	3	3	3	3
	Num filters per layer	[32, 32, 32]	[32, 32, 32]	[32, 32, 32]	[32, 32, 32]
	Kernel size per layer	[5, 3, 3]	[5, 3, 3]	[5, 3, 3]	[5, 3, 3]
	Stride per layer	[4, 1, 1]	[4, 1, 1]	[4, 1, 1]	[4, 1, 1]
	Policy encoder hiddens	[128, 32]	[128, 32]	[128, 32]	[128, 32]
	Encoding size	4	4	4	4
	Policy hiddens	[32, 32]	[32, 32]	[32, 32]	[32, 32]
	Num policy networks	8	6	6	3
	Q encoder hiddens	[256, 128]	[256, 128]	[256, 128]	[256, 128]
	Q hiddens	[128, 128]	[128, 128]	[128, 128]	[128, 128]
	Activation	ReLU	ReLU	ReLU	ReLU

A.3 Perturbed environments specifications

The state and action components of the four perturbed environments (Ant, Half Cheetah, Walker, Hopper) are the same as the ones of the environment they are derived from. They are listed in Table 5, ordered as they appear in the state and action vectors. The body parameters are sampled from a uniform distribution at the beginning of each episode. The body segments are perturbed in small subgroups, e.g., the two segments forming a limb of Ant, in which the perturbation is consistent. The parameters defining a perturbation space are detailed in Table 6.

During our experiments, we experienced an unusual behavior in the Half Cheetah environment for particularly light limbs. In this case, the Cheetah could move the legs at very high speed. In this case, PyBullet allowed an unnatural locomotion, in which the agent could advance without touching the ground, seemingly ignoring gravity. For this reason, we set a lower bound to the perturbation on the length and the thickness of the limbs, corresponding to -0.3 (only for the Half Cheetah with $\sigma = 0.5$). This clipping removed the unusual behavior of the simulator.

Table 5: State and action components of each environment. α is the angle between the target direction and the current running direction. v_i, $i \in x, y, z$ are the components of the linear velocity vector.

Environment Components	Ant	Half Cheetah	Walker	Hopper
State	Elevation	Elevation	Elevation	Elevation
	$sin(\alpha)$	$sin(\alpha)$	$sin(\alpha)$	$sin(\alpha)$
	$cos(\alpha)$	$cos(\alpha)$	$cos(\alpha)$	$cos(\alpha)$
	v_x	v_x	v_x	v_x
	v_y	v_y	v_y	v_y
	v_z	v_z	v_z	v_z
	roll	roll	roll	roll
	pitch	pitch	pitch	pitch
	Back left hip angle	Back thigh angle	Right thigh angle	Thigh angle
	Back left hip velocity	Back thigh velocity	Right thigh velocity	Thigh velocity
	Back left knee angle	Back shin angle	Right leg angle	Leg angle
	Back left knee velocity	Back shin velocity	Right leg velocity	Leg velocity
	Front left hip angle	Back foot angle	Right foot angle	Foot angle
	Front left hip velocity	Back foot velocity	Right foot velocity	Foot velocity
	Front left knee angle	Front thigh angle	Left thigh angle	Foot contact
	Front left knee velocity	Front thigh velocity	Left thigh velocity	
	Back right hip angle	Front foot angle	Left leg angle	
	Back right hip velocity	Front foot velocity	Left leg velocity	
	Back right knee angle	Front foot contact	Left foot angle	
	Back right knee velocity	Front shin contact	Left foot velocity	
	Front right hip angle	Front thigh contact	Right foot contact	
	Front right hip velocity	Back foot contact	Left foot contact	
	Front right knee angle	Back shin contact		
	Front right knee velocity	Back thigh contact		
	Back left foot contact			
	Front left foot contact			
	Back right foot contact			
	Front right foot contact			
Action	Back left hip torque	Front foot torque	Right thigh torque	Thigh torque
	Back left knee torque	Front shin torque	Right leg torque	Leg torque
	Front left hip torque	Front thigh torque	Right foot torque	Foot torque
	Front left knee torque	Back foot torque	Left thigh torque	
	Back right hip torque	Back shin torque	Left leg torque	
	Back right knee torque	Back thigh torque	Left foot torque	
	Front right hip torque			
	Front right knee torque			

A.4 Performance summary

Table 7 and Table 8 compare the IID performance of Simple-Oracle and of TCN-DMAP. The numbers in bold denote a significant statistical difference between the two methods (p-value < 0.001, paired t-test). We then list the IID (9) and OOD (10 11 12) test results of all the agents trained for this work. Some negative values should not surprise the reader, as some agents, when tested way outside

Table 6: Perturbation parameters of each environment.

Environment Perturbation type	Ant	Half Cheetah	Walker	Hopper
Thickness	Torso Front left limb Front right limb Back left limb Back right limb	Head Torso Back limb Front limb	Torso Limbs Feet	Torso Limb Foot
Length	Front left limb Front right limb Back left limb Back right limb	Torso Back limb Front limb	Limbs Feet	Limb Foot
Total	9	7	5	5

of the training distribution, fail to walk, collecting more penalization (e.g., due to undesired contact force or excessive energy expenditure) than positive reward.

Table 7: Focus: IID performance of Simple and Oracle, mean episode reward $\pm$ SEM.

Env	Ant		Half Cheetah		Walker		Hopper	
Algo	Simple	Oracle	Simple	Oracle	Simple	Oracle	Simple	Oracle
$\sigma = 0.1$	2164 ± 17	2148 ± 16	1633 ± 25	$\mathbf{2203 \pm 17}$	$\mathbf{1909 \pm 17}$	1780 ± 26	1807 ± 25	1859 ± 18
$\sigma = 0.3$	1270 ± 20	$\mathbf{1723 \pm 18}$	1412 ± 22	$\mathbf{1469 \pm 24}$	691 ± 17	$\mathbf{908 \pm 23}$	1121 ± 18	$\mathbf{1296 \pm 26}$
$\sigma = 0.5$	391 ± 14	$\mathbf{974 \pm 18}$	482 ± 19	$\mathbf{668 \pm 24}$	397 ± 14	$\mathbf{625 \pm 17}$	863 ± 19	$\mathbf{919 \pm 18}$

Table 8: IID performance of TCN and DMAP, mean episode reward $\pm$ SEM.

Env	Ant		Half Cheetah		Walker		Hopper	
Algo	TCN	DMAP	TCN	DMAP	TCN	DMAP	TCN	DMAP
$\sigma = 0.1$	932 ± 45	$\mathbf{2240 \pm 11}$	2278 ± 10	2261 ± 11	1060 ± 42	$\mathbf{1229 \pm 24}$	1762 ± 16	$\mathbf{1842 \pm 18}$
$\sigma = 0.3$	251 ± 11	$\mathbf{1623 \pm 19}$	1540 ± 20	1577 ± 22	518 ± 19	$\mathbf{893 \pm 11}$	$\mathbf{1368 \pm 20}$	1316 ± 24
$\sigma = 0.5$	481 ± 20	$\mathbf{960 \pm 14}$	553 ± 27	$\mathbf{669 \pm 23}$	584 ± 18	$\mathbf{743 \pm 15}$	$\mathbf{1017 \pm 18}$	953 ± 15

Table 9: IID performance, as mean $\pm$ sem.

Algorithm σ	RMA	DMAP	DMAP-ne	TCN	Oracle	Simple
Ant $\sigma = 0.1$	2138 ± 16	2240 ± 11	1966 ± 18	932 ± 45	2148 ± 16	2164 ± 17
Ant $\sigma = 0.3$	1700 ± 17	1623 ± 19	1542 ± 19	251 ± 11	1723 ± 18	1270 ± 20
Ant $\sigma = 0.5$	966 ± 16	960 ± 14	881 ± 12	481 ± 20	974 ± 18	391 ± 14
Half Cheetah $\sigma = 0.1$	2197 ± 16	2261 ± 11	1493 ± 15	2278 ± 10	2203 ± 17	1633 ± 25
Half Cheetah $\sigma = 0.3$	1402 ± 27	1577 ± 22	1132 ± 29	1540 ± 20	1469 ± 24	1412 ± 22
Half Cheetah $\sigma = 0.5$	595 ± 26	669 ± 23	507 ± 29	553 ± 27	668 ± 24	482 ± 19
Walker $\sigma = 0.1$	1750 ± 28	1229 ± 24	337 ± 16	1060 ± 42	1780 ± 26	1909 ± 17
Walker $\sigma = 0.3$	836 ± 23	893 ± 11	470 ± 16	518 ± 19	908 ± 23	691 ± 17
Walker $\sigma = 0.5$	579 ± 17	743 ± 15	341 ± 17	584 ± 18	625 ± 17	397 ± 14
Hopper $\sigma = 0.1$	1859 ± 19	1842 ± 18	984 ± 33	1762 ± 16	1859 ± 18	1807 ± 25
Hopper $\sigma = 0.3$	1267 ± 27	1316 ± 24	748 ± 23	1368 ± 20	1296 ± 26	1121 ± 18
Hopper $\sigma = 0.5$	730 ± 21	953 ± 15	482 ± 20	1017 ± 18	919 ± 18	863 ± 19

We also show the graphs of the reward decay of Oracle, DMAP and TCN as a function of the perturbation intensity (Figure 9). Finally, we visualize the distribution of the IID and OOD episode reward of DMAP, compared to TCN, across test body configurations, normalized by the Oracle reward (Figure 10 and 11).

Table 10: OOD performance (train sigma 0.1), as mean $\pm$ sem

Algorithm σ	RMA	DMAP	DMAP-ne	TCN	Oracle	Simple
Ant $\sigma = 0.1$	2138 ± 16	2240 ± 11	1966 ± 18	932 ± 45	2148 ± 16	2164 ± 17
Ant $\sigma = 0.3$	1124 ± 26	1105 ± 26	1004 ± 22	479 ± 26	1078 ± 27	1034 ± 28
Ant $\sigma = 0.5$	604 ± 17	579 ± 18	536 ± 16	289 ± 15	598 ± 16	572 ± 19
Ant $\sigma = 0.7$	145 ± 30	129 ± 30	139 ± 28	162 ± 16	154 ± 30	169 ± 26
Half Cheetah $\sigma = 0.1$	2197 ± 16	2261 ± 11	1493 ± 15	2278 ± 10	2203 ± 17	1633 ± 25
Half Cheetah $\sigma = 0.3$	1159 ± 38	1369 ± 38	877 ± 36	1426 ± 36	966 ± 43	825 ± 36
Half Cheetah $\sigma = 0.5$	216 ± 49	374 ± 52	137 ± 46	475 ± 51	100 ± 48	208 ± 41
Half Cheetah $\sigma = 0.7$	-307 ± 45	-142 ± 48	-205 ± 44	-147 ± 48	-375 ± 44	-225 ± 42
Walker $\sigma = 0.1$	1750 ± 28	1229 ± 24	337 ± 16	1060 ± 42	1780 ± 26	1909 ± 17
Walker $\sigma = 0.3$	877 ± 37	666 ± 25	283 ± 15	504 ± 31	622 ± 34	900 ± 37
Walker $\sigma = 0.5$	318 ± 26	265 ± 19	153 ± 13	211 ± 20	226 ± 20	354 ± 27
Walker $\sigma = 0.7$	165 ± 20	140 ± 15	114 ± 12	113 ± 14	106 ± 14	165 ± 19
Hopper $\sigma = 0.1$	1859 ± 19	1842 ± 18	984 ± 33	1762 ± 16	1859 ± 18	1807 ± 25
Hopper $\sigma = 0.3$	1044 ± 34	857 ± 33	536 ± 27	960 ± 36	947 ± 33	760 ± 35
Hopper $\sigma = 0.5$	356 ± 24	308 ± 23	216 ± 20	310 ± 29	417 ± 25	303 ± 22
Hopper $\sigma = 0.7$	217 ± 19	208 ± 19	160 ± 15	206 ± 24	219 ± 18	188 ± 18

Table 11: OOD performance train sigma 0.3, as mean $\pm$ sem

Algorithm σ	RMA	DMAP	DMAP-ne	TCN	Oracle	Simple
Ant $\sigma = 0.1$	2103 ± 6	2051 ± 7	2015 ± 6	238 ± 12	2124 ± 5	1671 ± 20
Ant $\sigma = 0.3$	1700 ± 17	1623 ± 19	1542 ± 19	251 ± 11	1723 ± 18	1270 ± 20
Ant $\sigma = 0.5$	924 ± 22	856 ± 22	833 ± 20	227 ± 9	900 ± 24	796 ± 18
Ant $\sigma = 0.7$	306 ± 32	228 ± 33	230 ± 32	185 ± 12	261 ± 30	283 ± 28
Half Cheetah $\sigma = 0.1$	1795 ± 11	1900 ± 9	1578 ± 15	1719 ± 8	1824 ± 9	1704 ± 11
Half Cheetah $\sigma = 0.3$	1402 ± 27	1577 ± 22	1132 ± 29	1540 ± 20	1469 ± 24	1412 ± 22
Half Cheetah $\sigma = 0.5$	822 ± 42	1033 ± 38	552 ± 42	1117 ± 35	892 ± 43	787 ± 40
Half Cheetah $\sigma = 0.7$	350 ± 48	453 ± 47	151 ± 45	462 ± 45	250 ± 50	267 ± 45
Walker $\sigma = 0.1$	906 ± 22	943 ± 9	482 ± 15	574 ± 19	1028 ± 21	846 ± 12
Walker $\sigma = 0.3$	836 ± 23	893 ± 11	470 ± 16	518 ± 19	908 ± 23	691 ± 17
Walker $\sigma = 0.5$	487 ± 23	518 ± 20	239 ± 14	342 ± 18	491 ± 24	388 ± 18
Walker $\sigma = 0.7$	248 ± 18	289 ± 19	171 ± 14	219 ± 16	267 ± 19	221 ± 16
Hopper $\sigma = 0.1$	1683 ± 16	1676 ± 14	729 ± 25	1591 ± 12	1618 ± 16	1351 ± 9
Hopper $\sigma = 0.3$	1267 ± 27	1316 ± 24	748 ± 23	1368 ± 20	1296 ± 26	1121 ± 18
Hopper $\sigma = 0.5$	532 ± 28	729 ± 30	456 ± 23	672 ± 29	666 ± 29	562 ± 23
Hopper $\sigma = 0.7$	329 ± 24	443 ± 27	334 ± 21	382 ± 26	371 ± 26	342 ± 21

A.5 Analysis of the attention dynamics

We quantify the tangling of a trajectory with the measure introduced by Russo et al. [43]:

$$Q(t) = \max_{t'} \frac{||\dot{x_t} - \dot{x_{t'}}||^2}{||x_t - x_{t'}||^2 + \epsilon},$$

where x_t represents the trajectory at a specific time and $x_{t'}$ denotes the corresponding temporal derivative, and ϵ is a constant. We use this measure to study the the difference in trajectory tangling between attention and action/observation (Figure 12). We find that the attention dynamics are less tangled than the inputs. This suggests that DMAP achieves control across morphological parameters by regularizing.

To gain further insights into this mechanism, we verified if the attentional dynamics become less entangled over learning. Thus, we visualized how the attention dynamics evolves throughout the training (Figure 13). While at an early stage the characteristic rotational dynamics is absent, it emerges as the training proceeds, progressively untangling the trajectories.

Table 12: OOD performance train sigma 0.5, as mean $\pm$ sem

Algorithm σ	RMA	DMAP	DMAP-ne	TCN	Oracle	Simple
Ant $\sigma = 0.1$	1667 ± 5	1477 ± 5	1347 ± 7	692 ± 30	1665 ± 6	358 ± 13
Ant $\sigma = 0.3$	1428 ± 11	1280 ± 11	1133 ± 11	620 ± 26	1454 ± 11	444 ± 14
Ant $\sigma = 0.5$	966 ± 16	960 ± 14	881 ± 12	481 ± 20	974 ± 18	391 ± 14
Ant $\sigma = 0.7$	409 ± 27	504 ± 22	364 ± 31	273 ± 21	479 ± 21	236 ± 20
Half Cheetah $\sigma = 0.1$	748 ± 23	816 ± 12	718 ± 18	466 ± 33	834 ± 17	601 ± 7
Half Cheetah $\sigma = 0.3$	697 ± 23	788 ± 15	660 ± 22	603 ± 27	766 ± 21	585 ± 9
Half Cheetah $\sigma = 0.5$	595 ± 26	669 ± 23	507 ± 29	553 ± 27	668 ± 24	482 ± 19
Half Cheetah $\sigma = 0.7$	399 ± 30	443 ± 30	303 ± 33	373 ± 31	364 ± 33	312 ± 27
Walker $\sigma = 0.1$	746 ± 13	949 ± 7	533 ± 17	702 ± 16	783 ± 14	468 ± 12
Walker $\sigma = 0.3$	722 ± 14	887 ± 11	490 ± 17	716 ± 16	787 ± 13	484 ± 13
Walker $\sigma = 0.5$	579 ± 17	743 ± 15	341 ± 17	584 ± 18	625 ± 17	397 ± 14
Walker $\sigma = 0.7$	374 ± 17	504 ± 19	207 ± 15	418 ± 19	424 ± 18	318 ± 14
Hopper $\sigma = 0.1$	1169 ± 13	1109 ± 5	612 ± 19	1272 ± 8	1224 ± 13	1002 ± 7
Hopper $\sigma = 0.3$	961 ± 18	1102 ± 8	675 ± 19	1161 ± 14	1070 ± 16	1036 ± 12
Hopper $\sigma = 0.5$	730 ± 21	953 ± 15	482 ± 20	1017 ± 18	919 ± 18	863 ± 19
Hopper $\sigma = 0.7$	428 ± 21	591 ± 23	342 ± 19	643 ± 25	558 ± 22	576 ± 22

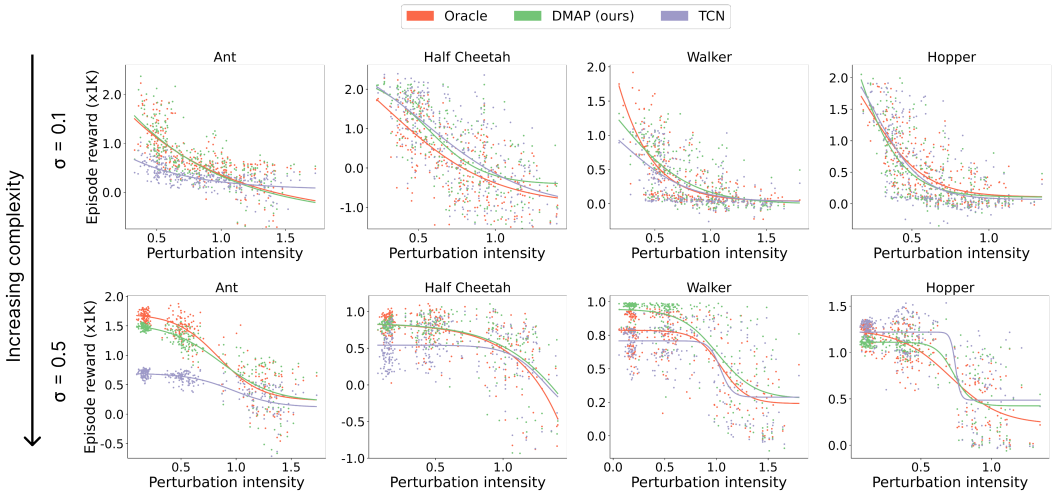

Figure 9: Reward obtained by Oracle, TCN and DMAP, trained in the $\sigma = 0.1$ and $\sigma = 0.5$ environments, on a wide range of perturbations (both IID and OOD). Each point represents the average reward across 5 random seeds for one body configuration, while the perturbation intensity is the eucledian norm of the perturbation vector. The curves are obtained by fitting a sigmoid function. DMAP match or even surpass the Oracle performance and outperform TCN (except for Hopper with $\sigma = 0.5$) for all training environment.

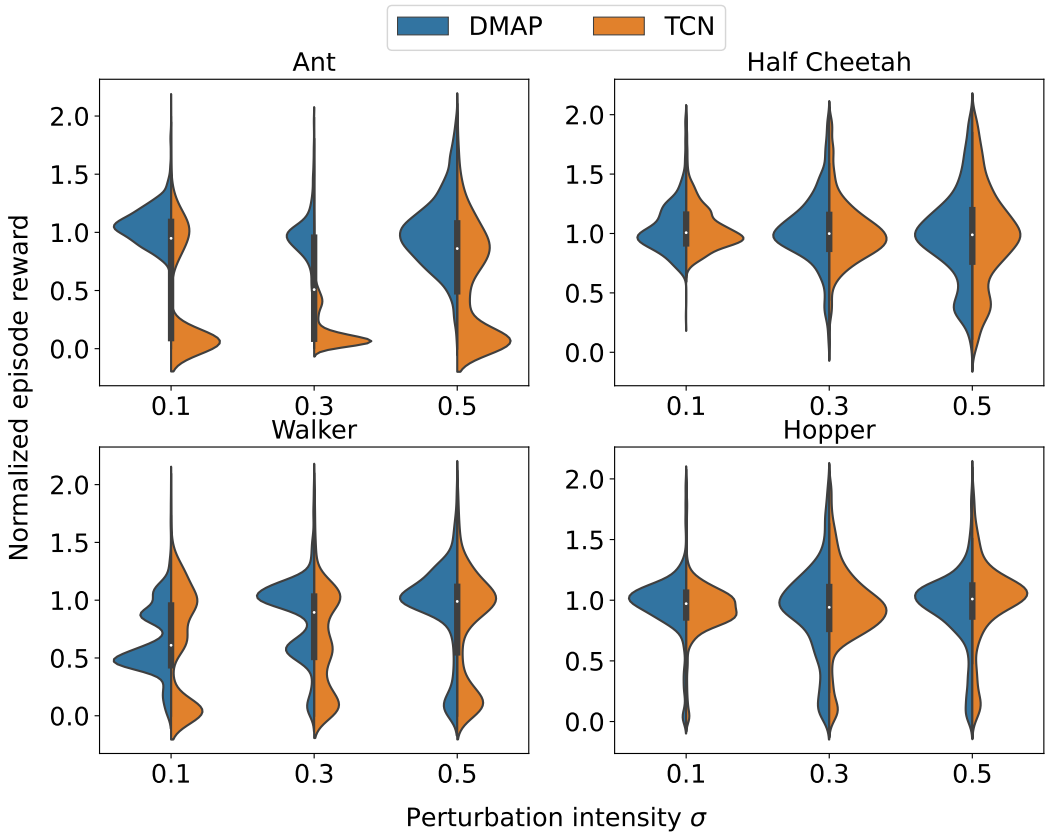

Figure 10: The normalized cumulative episode reward collected in each of the 100 test body configurations (IID test) by TCN and DMAP. Normalization was done based on Oracle rewards per morphology. We can observe that the low reward obtained by TCN in Ant and Walker is due to the density peaks in the lower part of the graphs, corresponding to agents trained with random seeds which did not lead to a successful outcome.

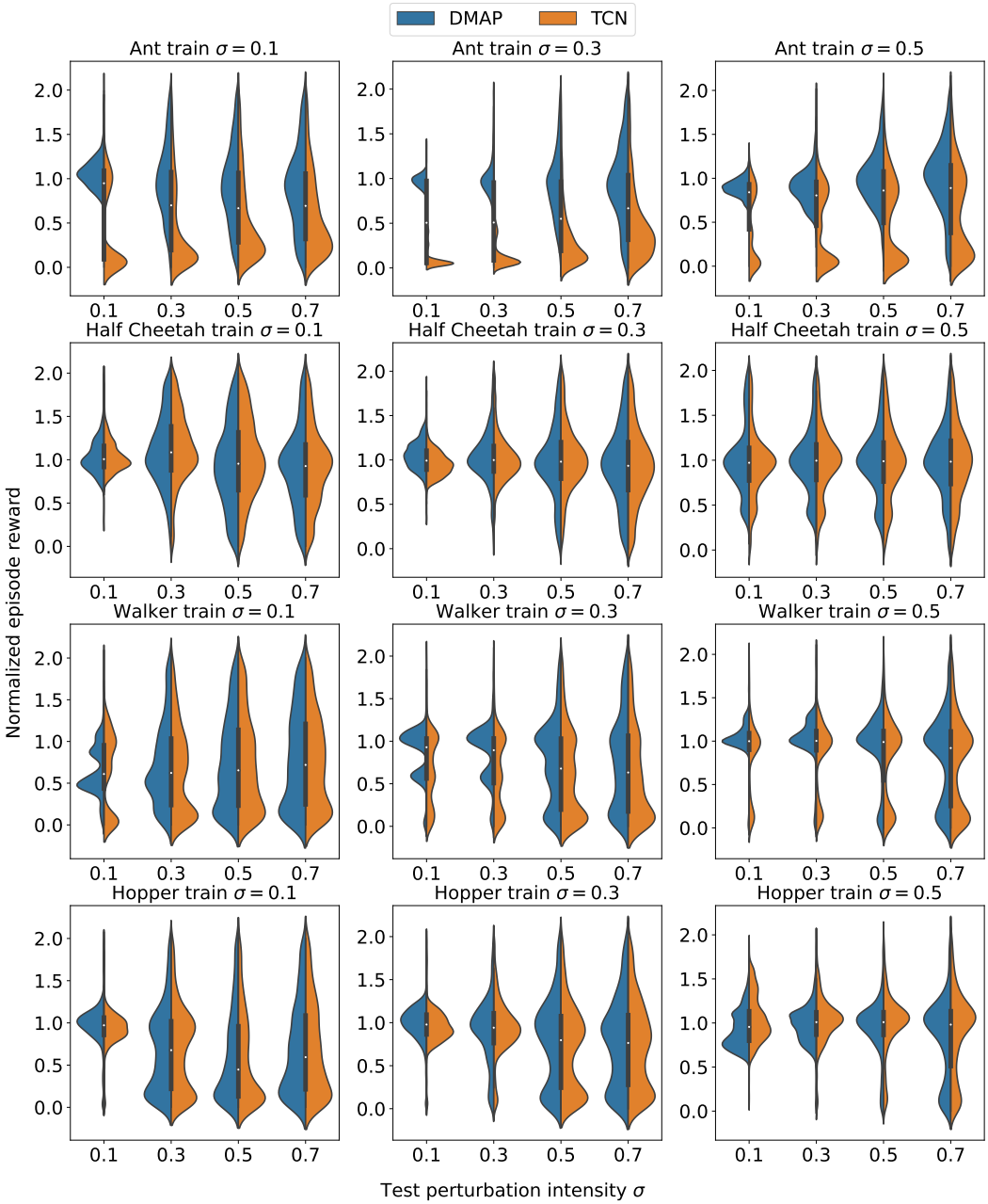

Figure 11: As in Figure 10, but keeping the training σ fixed, while changing the test σ. In general, as the perturbation level increases, the distributions become less concentrated, because the variance of the relative performance increases. This suggests that, when tested OOD, TCN and DMAP do not necessarily perform well or badly across body shapes proportionally to Oracle's performance.

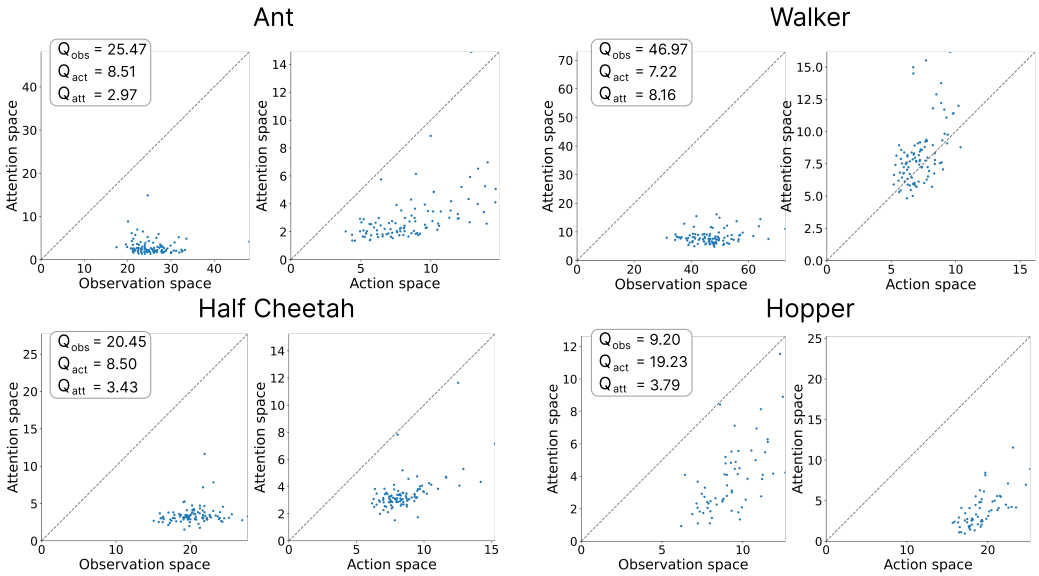

Figure 12: Scatter plot of the tangling measure for the attention embedding trajectories versus the observation and action ones. Each point represents the tangling measure for one of the 100 test body configurations averaged over time. The data refer to all the environments with $\sigma = 0.1$ and to the corresponding IID test body configurations. The large majority of the points lie below the threshold line, strongly suggesting that the attention trajectories are less tangled than those of the input actions and observations.

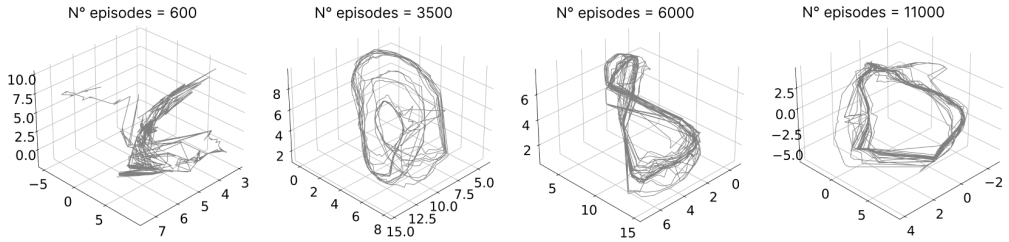

Figure 13: UMAP embedding of the attention matrix K of a single episode for different checkpoints during the training of an agent in the Ant environment with $\sigma = 0.1$. The characteristic rotational dynamics emerges as the agent learns how to walk.