A latent control model for realistic rodent motion

Aidan Sirbu[1,2,3,✉] **Charles Y. Zhang**[4] **Yuanjia Yang**[6,7]
Bence P. Ölveczky[5] **Talmo D. Pereira**[6] **Blake A. Richards**[1,2,3,8,9,10]

[1]Google, Paradigms of Intelligence Team [2]Mila – Quebec AI Institute
[3]School of Computer Science, McGill University
[4]Harvard University [5]Department of Organismal and Evolutionary Biology, Harvard University
[6]Salk Institute for Biological Studies [7]University of California San Diego
[8]Department of Neurology and Neurosurgery, McGill University
[9]Montreal Neurological Institute, McGill University
[10]Learning in Machines and Brains Program, CIFAR

[✉]aidan.sirbu@mail.mcgill.ca
https://aidansirbu.com/SCAMPER-site/

Abstract

The development of embodied models of animal behavior is a major goal of the neuro-AI community. Embodied models will enhance our ability to make direct comparisons between biological and artificial agents. To make such comparisons effective, though, we need embodied models that actually move like animals. Previous work has developed virtual animal bodies and techniques for training realistic motion through imitation. However, to date, there is no system for embodied animal modeling that produces natural movements both during trained goal-directed behavior, and during exploration driven by random generation. Here, we introduce **S**imulated **C**ontrol via **A**ugmented **M**otor **P**riors for **E**mbodied **R**einforcement learning (SCAMPER), a system for training an embodied virtual rat model, enabling natural movements during both goal-directed behavior and random exploration. SCAMPER uses a variational architecture with two distinct priors, a motion prior for control and a behavior prior for exploration. We show that combining both of these priors allows SCAMPER to generate realistic behavior during exploration and exploit the task structure effectively. SCAMPER can also learn from sparse rewards as a result of the diverse movements induced by the behavior prior, allowing the system to model realistic sparse-reward tasks from experimental neuroscience. Altogether, SCAMPER provides the neuro-AI community with an effective new system for training embodied models of rodents on realistic tasks.

1 Introduction

One of the ultimate goals of neuro-AI research is to be able to fully capture the richness of animal learning and behavior with embodied computational models [1, 2]. Previous work has developed realistic virtual bodies for a variety of model organisms commonly used in neuroscience experiments [3]. However, controlling a realistic virtual body to consistently produce natural motion is non-trivial. For example, when an artificial neural network was trained to control a virtual rat on a variety of tasks using deep reinforcement learning (RL), the model could solve the tasks, but often did so with unnatural movements that did not resemble the motion of real rats [4]. While researchers have since used imitation learning to improve the realism of virtual animal motion, these models are either limited to pure imitation [5], or struggle to transfer to downstream RL tasks [6].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Within robotics, physics-based controllers trained with RL to produce realistic humanoid motion can generalize to new tasks [7–12]. However, most approaches have no mechanism for random generation of realistic motion. One exception is the recent Physics-based Universal motion Latent SpacE (PULSE) framework that uses residual control of a variational motion prior [13]. PULSE can sample from its motion prior to produce random, yet realistic movements. However, this sampling exhibits limited diversity: depending on initialization, the system typically settles into a single behavioral mode and stays there. This is especially problematic when PULSE is used as a low-level controller for sparse-reward tasks, which demand substantial structured exploration [14, 15]. Because most experimental tasks in neuroscience offer only sparse extrinsic reward [16–18], training embodied animal models to solve them while maintaining naturalistic motion remains an open problem.

Here, we present **S**imulated **C**ontrol via **A**ugmented **M**otor **P**riors for **E**mbodied **R**einforcement learning (SCAMPER), a system for training an embodied virtual rat model, enabling natural movements during both goal-directed behavior and random exploration. SCAMPER builds on the PULSE framework [13], but adds an additional behavior prior that greatly increases the diversity of randomly generated movements while preserving the realism of the motion. We provide a formal analysis demonstrating why the behavior prior increases the diversity of generated movements. We then show empirically that SCAMPER produces a diverse array of realistic behavior during random rollouts, covering the behavioral repertoire present in the training data and beyond. We also demonstrate that SCAMPER transfers effectively to a range of downstream tasks, from dense-reward locomotion and rearing to sparse-reward foraging tasks, all while maintaining naturalistic motion throughout training. Altogether, SCAMPER provides the neuro-AI community with a powerful new tool for embodied modeling of animal behavior.

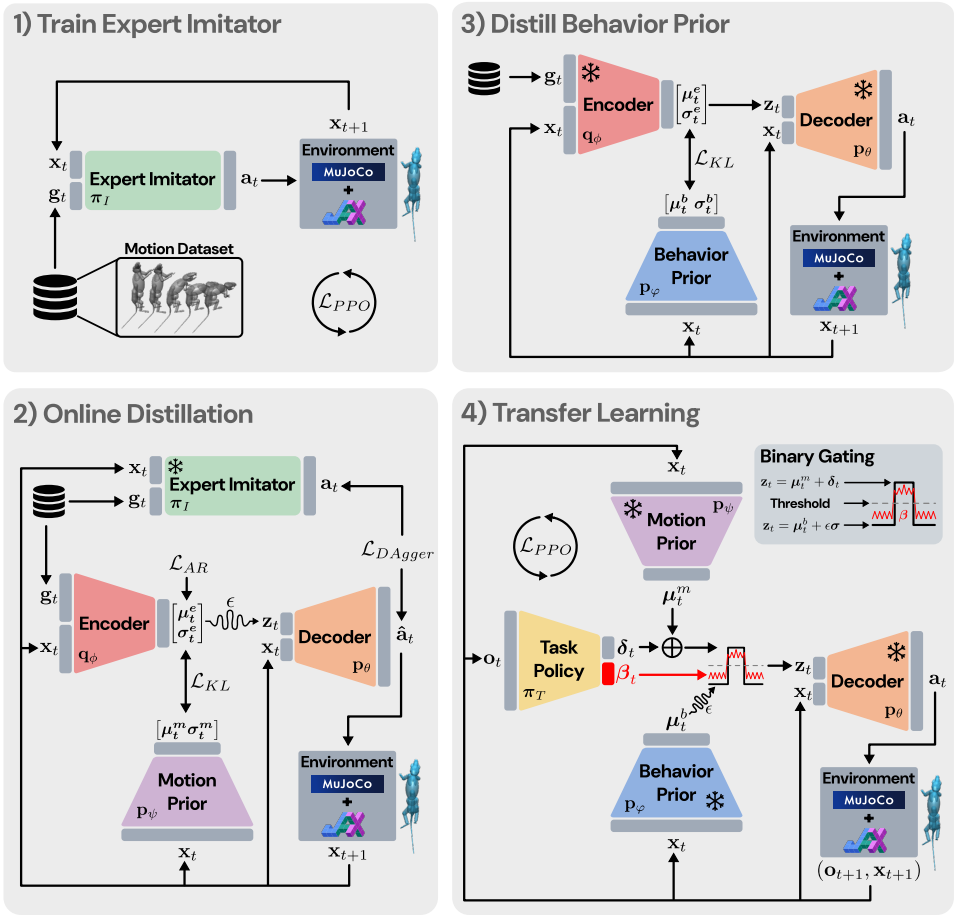


Figure 1: **Training stages of SCAMPER.** (1) The expert imitator is trained with mocap data. (2) The latent controller is trained via online distillation. (3) The behavior prior is distilled from the frozen encoder. Finally, (4) the model can be trained with RL to perform different tasks.

2 Related work

Motor priors and latent skill spaces for physics-based control. Training physics-based controllers from scratch for different tasks often yields unnatural motion [19], motivating imitation-based controllers that track reference motion data [7]. Merel et al. [8] extended this approach by distilling a tracking expert into a latent skill space through an information bottleneck, and subsequent work explored adversarial alternatives [9, 10, 12] and variational formulations [20–22]. PULSE [13] distills a universal motion imitator [23] into a variational model regularized with a learnable motion prior, achieving broad coverage of human motor skills. These frameworks target humanoid characters trained on human motion capture (mocap) data; their applicability to non-human morphologies driven by species-specific behavioral data has not yet been demonstrated. Moreover, the learned motion prior in PULSE supports local motion continuation, but random rollouts tend to remain within the behavioral mode in which they are initialized, producing locally plausible but behaviorally narrow motion. Kinematic motion models, including autoregressive CVAE-based priors [24] and diffusion-based generators [25], can produce diverse long-horizon motion but do not yield a physics-based controller. Furthermore, autoregressive policies have been applied to physics-based control [26, 27], but they provide no mechanism for sampling diverse motion that respects a species-specific behavioral repertoire. Here, we extend the PULSE framework to a biomechanically realistic rodent trained on 3D kinematic recordings of freely behaving rats, and complement the motion prior with a behavior prior that induces realistic transitions across behavioral modes.

Physics-based control of virtual animals. Biomechanically realistic virtual animals have emerged as a tool for neuro-AI. Merel et al. [4] introduced a virtual rodent trained with RL, but without imitation pretraining the resulting motion was unnatural. Aldarondo et al. [5] extended this work by training a controller for the virtual rat to imitate freely moving rats and found that its internal representations matched those in the motor cortex and dorsolateral striatum of the real rats; the controller was, however, purely imitative and could not generate diverse behavior. MIMIC-MJX [6] generalizes this tracking-imitation approach across species via an open-source JAX/MJX pipeline, but supports only limited transfer learning. Parallel efforts in *Drosophila* [28, 29] and *C. elegans* [30] further extend this paradigm beyond rodents. In zebrafish, Keller et al. [31] drive naturalistic behavioral transitions via model-based intrinsic motivation against an ethological prior. Across this line of work, no existing model provides the ability to both mimic real motion and support sampling appropriate for exploration. Our work fills this gap, yielding a controller that can be both reused across tasks and sampled for diverse, naturalistic behavior.

3 Overview of SCAMPER

3.1 Preliminaries

We define the proprioception of the agent at a given timestep as

$$\mathbf{x}_t = (\boldsymbol{\theta}_t, \dot{\boldsymbol{\theta}}_t, \boldsymbol{\tau}_t, h_t, \boldsymbol{\phi}_t, \mathbf{p}_t^{ee}, \mathbf{s}_t^{kin}, \mathbf{c}_t, \mathbf{a}_{t-1}), \tag{1}$$

containing the joint angles ($\boldsymbol{\theta}_t$), angular velocities ($\dot{\boldsymbol{\theta}}_t$), joint torques ($\boldsymbol{\tau}_t$), body height ($h_t$), orientation relative to the world z-axis ($\boldsymbol{\phi}_t$, akin to a vestibular signal), egocentric end-effector positions ($\mathbf{p}_t^{ee}$), kinematic sensors (accelerometer, velocimeter, gyro; $\mathbf{s}_t^{kin}$), end-effector touch sensors ($\mathbf{c}_t$), and the previous action ($\mathbf{a}_{t-1}$). During imitation tasks, the goal of the agent at a given timestep is defined as

$$\mathbf{g}_t = (\mathbf{r}_{t+1} - \mathbf{r}_t, \mathbf{q}_{t+1} \ominus \mathbf{q}_t, \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t, \mathbf{p}_{t+1}^{seg} - \mathbf{p}_t^{seg}), \tag{2}$$

and specifies the displacement the agent must achieve, computed as differences between its current state and the target state at the next timestep in the root position ($\mathbf{r}_t$), root orientation ($\mathbf{q}_t$; where $\ominus$ is egocentric quaternion difference), joint angles ($\boldsymbol{\theta}_t$), and egocentric Cartesian body segment positions ($\mathbf{p}_t^{seg}$). Target states are provided by a mocap dataset registered to the biomechanical rat model via inverse kinematics using STAC-MJX [6]. During imitation the policy network is provided the next five goal states ($g_{t:t+4}$). During transfer learning, the agent no longer receives a goal state, $\mathbf{g}_t$, and instead observes task-specific environmental observations $\mathbf{o}_t$ (e.g., scent gradient). In this work we use the terms 'rat', 'rodent', and 'agent' interchangeably.

We frame each task (aside from the distillation objectives) as goal-conditioned RL, formulated as a Markov Decision Process $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$. The states $s_t \in \mathcal{S}$ contain $\mathbf{x}_t$ and either $\mathbf{g}_t$ or

$\mathbf{o}_t$; all values are normalized by their running mean and variance. Actions $\mathbf{a}_t \in \mathcal{A}$ are torques applied to the rodent's actuated joints. $\mathcal{T}$ and $\mathcal{R}$ are the transition dynamics (determined by the physics environment) and the reward function, respectively. We train policies with proximal policy optimization (PPO) [32] to maximize the discounted return $\mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t\right]$.

3.2 Training pipeline

SCAMPER uses a four-stage training pipeline (Fig. 1). All networks are multi-layer perceptrons (MLPs), but in principle other architectures could be used.

(1) Expert imitator. We train an expert imitator $\pi_I(\mathbf{a}_t|\mathbf{x}_t, \mathbf{g}_t)$ with PPO to reproduce the mocap trajectories (SuppVid1; App. C).

(2) Online distillation. Following the PULSE framework [13], we distill the expert imitator into a latent skill controller. Similar to PULSE, we find that training the latent skill controller using supervised distillation of a pre-trained expert leads to better training stability than training the latent skill controller from scratch using RL, hence we adopt the same approach. The controller consists of three jointly trained networks: an encoder $q_\phi(\mathbf{z}_t|\mathbf{x}_t, \mathbf{g}_t)$, a decoder $p_\theta(\mathbf{a}_t|\mathbf{x}_t, \mathbf{z}_t)$, and a proprioception-conditioned motion prior $p_\psi(\mathbf{z}_t|\mathbf{x}_t)$ over latent codes. Training proceeds by rolling out the student controller on the imitation task in simulation, while querying the frozen expert, π_I, for target actions on visited states, following DAgger [33]. The networks are jointly trained by minimizing:

$$\mathcal{L}_{\text{distill}} = \mathcal{L}_{\text{DAgger}} + \lambda \mathcal{L}_{\text{KL}} + \alpha \mathcal{L}_{\text{AR}}, \tag{3}$$

where $\mathcal{L}_{\text{DAgger}} = ||\mathbf{a}_t - \hat{\mathbf{a}}_t||_2^2$ matches the expert's target action $\mathbf{a}_t$ to the decoder's prediction $\hat{\mathbf{a}}_t$, $\mathcal{L}_{\text{KL}} = D_{\text{KL}}(q_\phi||p_\psi)$ regularizes the encoder toward the motion prior, and $\mathcal{L}_{\text{AR}} = ||\boldsymbol{\mu}_t^e - \boldsymbol{\mu}_{t-1}^e||_2^2$ encourages temporal smoothness in the latent space. The decoder and motion prior are retained for downstream use; the encoder is kept only for the next training stage. λ and α are hyperparameters (Table 3).

(3) Behavior prior distillation. After online distillation, we train a second proprioception-conditioned prior $p_\varphi(\mathbf{z}_t|\mathbf{x}_t)$ with all other networks frozen. Rolling out the latent skill controller on the imitation task in simulation, we optimize p_φ by minimizing

$$\mathcal{L}_{\text{behavior}} = D_{\text{KL}}(q_\phi(\mathbf{z}_t|\mathbf{x}_t, \mathbf{g}_t) \,\|\, p_\varphi(\mathbf{z}_t|\mathbf{x}_t)). \tag{4}$$

By Proposition 2, this ensures p_φ captures the full goal-conditional variation in the encoder's outputs.

(4) Transfer learning. For downstream tasks, a task policy $\pi_T(\boldsymbol{\delta}_t, \beta_t|\mathbf{x}_t, \mathbf{o}_t)$ is trained with PPO to produce a residual correction $\boldsymbol{\delta}_t$ to the motion prior's mean $\boldsymbol{\mu}_t^m$. Importantly, we adjudicate between the two priors (motion and behavior) using a continuous gating variable $\beta_t \in [-1, 1]$ that is binarized to $b_t \in \{0, 1\}$ via hysteresis thresholding. Thus, at each timestep, b_t selects between two latent codes:

$$\mathbf{z}_t = \begin{cases} \boldsymbol{\mu}_t^m + \boldsymbol{\delta}_t & \text{if } b_t = 1, \\ \boldsymbol{\mu}_t^b + \boldsymbol{\epsilon}\boldsymbol{\sigma}, \quad \boldsymbol{\epsilon} \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{1}) & \text{if } b_t = 0, \end{cases} \tag{5}$$

where $\boldsymbol{\mu}_t^m$ and $\boldsymbol{\mu}_t^b$ are the means of $p_\psi(\mathbf{z}_t|\mathbf{x}_t)$ and $p_\varphi(\mathbf{z}_t|\mathbf{x}_t)$, respectively, and $\boldsymbol{\sigma}$ is a fixed noise scale. The resulting latent code is passed through the frozen decoder p_θ to produce the action. Intuitively, $b_t = 1$ engages goal-directed residual control of the motion prior (exploitation), while $b_t = 0$ samples naturalistic motion from the behavior prior (exploration). Because π_T outputs β_t, the system learns when to explore and when to exploit.

3.3 Formal analysis of the priors

Why does SCAMPER use both a motion prior, p_ψ, for goal-directed motion and a behavior prior, p_φ, for exploration? Although both p_ψ and p_φ are proprioception-conditioned distributions over the same latent space, their training objectives differ. We formalize this distinction in two propositions.

Proposition 1 (Information bottleneck under joint training)**.** *When the encoder $q_\phi(\mathbf{z}|\mathbf{g}, \mathbf{x})$ and the state-dependent prior $p_\psi(\mathbf{z}|\mathbf{x})$ are trained jointly, minimizing their expected KL divergence is equivalent to the following optimization:*

$$\min_{\phi,\psi} \mathbb{E}_{p(\mathbf{x},\mathbf{g})} \left[D_{\text{KL}}(q_\phi \parallel p_\psi)\right] = \min_{\phi,\psi} \left[I_\phi(Z; G \mid X) + \mathbb{E}_{p(\mathbf{x})} \left[D_{\text{KL}}(m_\phi \parallel p_\psi)\right]\right], \tag{6}$$

where $m_\phi(\mathbf{z}|\mathbf{x}) = \mathbb{E}_{p(\mathbf{g}|\mathbf{x})}[q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x})]$ *is the marginal posterior induced by the encoder, which evolved during joint training.*

Joint training of the encoder and the motion prior thus not only forces the motion prior to approximate the marginal posterior, but also minimizes the encoder's conditional mutual information between the latent states and the goals. However, from equation (3) both of these terms which the KL loss decomposes into are weighted by the same hyperparameter (λ). As a result, in practice, it is difficult to obtain a favorable balance between regularization via the bottleneck and the distortion of movement imitation. We find that setting λ in order to obtain the right balance limits how closely the motion prior can approximate the marginal posterior, yielding a motion prior concentrated on local motion continuation. We also find that stochastic sampling of the motion prior with high noise can occasionally push the system to switch behavioral modes, but as we show below (Fig. 2), the motion prior alone does not reproduce the rich pattern of behavioral transitions observed in the training data. Despite all this, a motion prior focused on local motion continuation yields a network favorable for residual control. The marginal posterior, by contrast, contains the more fine-grained structure of the behavioral transitions. Therefore, in order to have the practical ability to generate diverse behavior during exploration we need another prior that *only approximates the marginal posterior*, allowing us to decouple the training dynamics from minimizing the conditional mutual information. We can achieve this by training a new prior, the behavior prior, with the frozen encoder.

Proposition 2 (Marginal posterior approximation under frozen encoder)**.** *When the encoder* $q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x})$ *is frozen, minimizing the expected KL divergence between the encoder and a state-dependent prior* $p_\varphi(\mathbf{z}|\mathbf{x})$ *is equivalent to*

$$\min_\varphi \mathbb{E}_{p(\mathbf{x},\mathbf{g})}\left[D_{\mathrm{KL}}(q_\phi \parallel p_\varphi)\right] = I_\phi(Z;G \mid X) + \min_\varphi \mathbb{E}_{p(\mathbf{x})}\left[D_{\mathrm{KL}}(m_\phi \parallel p_\varphi)\right], \tag{7}$$

where $m_\phi(\mathbf{z}|\mathbf{x}) = \mathbb{E}_{p(\mathbf{g}|\mathbf{x})}[q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x})]$ *is the marginal posterior induced by the frozen encoder, held fixed during optimization. The conditional mutual information,* $I_\phi(Z;G \mid X)$*, does not depend on the optimization variable* φ*; it is therefore an additive constant that sits outside the minimization. Hence, the optimal behavior prior is the closest possible state-conditional approximation to the marginal posterior* m_ϕ*.*

Therefore, by directly distilling the behavior prior from the frozen pre-trained encoder, we are able to lower the KL divergence between it and the goal-rich encoder to a greater extent than we could for the motion prior which was limited by the rate-distortion tradeoff. Specifically, we achieve a KL divergence of 5.88 nats between the behavior prior and the encoder, vs 14.35 nats for the motion prior and the encoder. This allows the behavior prior to better capture the behavioral transitions in the training data. However, since the behavior prior is more inclined to sample from transitions in behavior in its control signal, it is less stable when residually controlled than the motion prior (Fig. 13, Table 11). This motivates the use of the dual priors for a sparse-reward exploration task in Section 4.5. We provide proofs for these two propositions in App. A and empirical justification in App. H.1. By training the behavior prior in this way, we achieve a good approximation for the marginal posterior, $m_\phi(\mathbf{z}|\mathbf{x})$, which can then be used for effective exploration with natural movements, as we show in the subsequent experiments.

4 Experiments

4.1 The behavior prior expands behavioral diversity beyond the motion prior

We first verify Propositions 1 and 2: that sampling the behavior prior produces greater behavioral diversity than sampling the motion prior (SuppVid[2-4]; App. C). We initialized the rodent in a realistic pose, then iteratively sampled a latent from one prior, decoded it to an action, and stepped the simulation for 5 s of simulated time per rollout (Fig. 2a). We collected 842 rollouts per noise level, comparing pure behavior-prior sampling ($b_t = 0$) to pure motion-prior sampling ($b_t = 1$); see App. F.1.

We measured the Vendi score [34], which captures the effective number of distinct behavioral modes. The Vendi score has been used to quantify trajectory and skill diversity in RL [35, 36]. Across all noise levels, the Vendi score was significantly higher for the behavior prior than the motion prior (Fig. 2b). Motion-prior sampling (similar to PULSE) produced lower Vendi scores than the training data for all $\sigma < 1$. At very high noise, motion-prior diversity matched the training data but at the cost

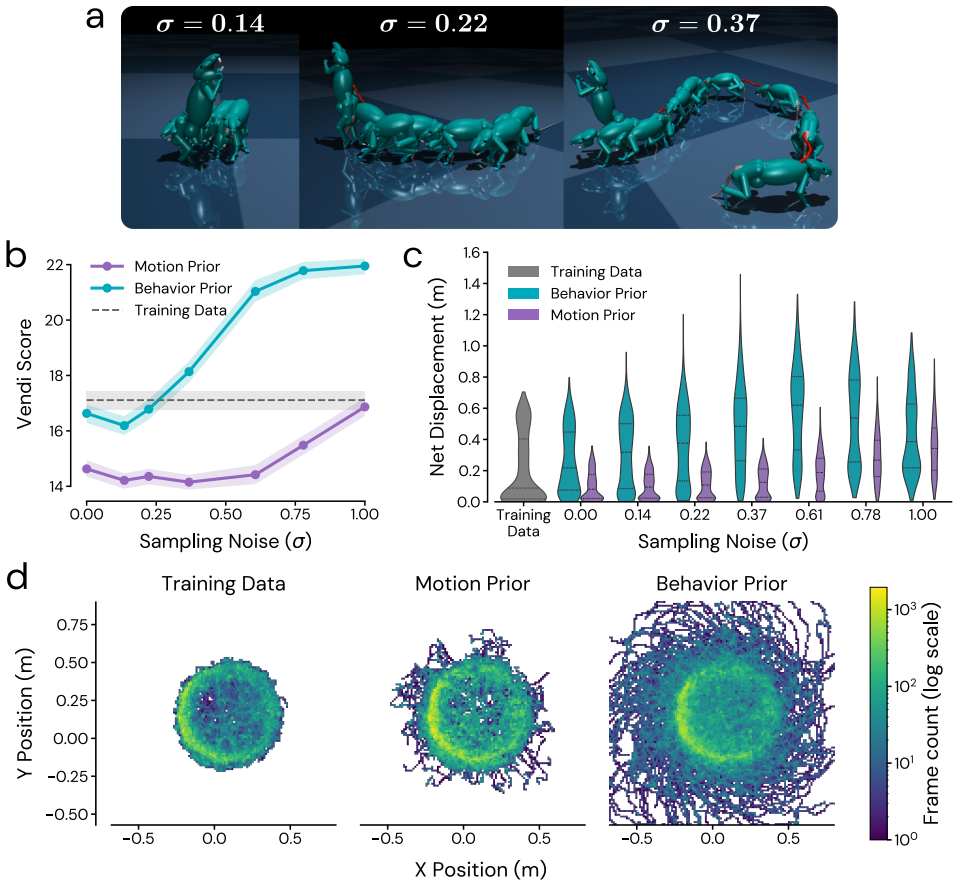


Figure 2: **SCAMPER is behaviorally diverse. (a)** Renderings of behavior-prior rollouts at increasing sampling noise. **(b)** Vendi score vs. latent sampling noise $\sigma = \exp(\text{logvar}/2)$, for the behavior prior (*blue*), motion prior (*purple*), and the training data (*dashed gray*); data shown is bootstrapped means with 95% CIs. **(c)** Per-rollout net displacement across σ; violins with interquartile lines, gray violin marks training data. **(d)** Spatial occupancy at $\sigma = 0.22$ shown as 2D histograms of body location (log-scaled frame counts) for the training data, motion prior, and behavior prior (see Fig. 9 for all noise levels).

of motion realism (Section 4.2). In contrast, the behavior prior matched the training data diversity at low noise and exceeded it at high noise. This provides a tunable knob for controlling the behavioral diversity of the agent.

Next, to assess whether this diversity aids exploration, we measured the net distance traveled across sampling noise levels. Mirroring the Vendi score results, motion-prior sampling produced shorter trajectories than the training data at almost all noise levels (Fig. 2c). Behavior-prior sampling roughly matched the training data distance at low noise and exceeded it at high noise. We also examined the spatial occupancy in the virtual arena. In the training data, the rat was in a circular arena, and thus, the occupancy was restricted to a circular area (Fig. 2d, *left*). Motion-prior rollouts at low noise remained largely within this same area despite no physical barrier (Fig. 2d, *middle*), demonstrating that the motion prior samples from a restricted distribution of movements seen in the training data. The behavior prior, in contrast, explored beyond this region (Fig. 2d, *right*).

4.2 The behavior prior preserves motion realism under stochastic sampling

To verify that the movements induced by the behavior prior during random exploration were natural we used the Kernel Inception Distance (KID), which measures the dissimilarity between real and generated probability distributions [37] and has been used to measure the realism of generative human motion [38]. We trained a variational autoencoder (VAE) on the joint angle trajectories from the

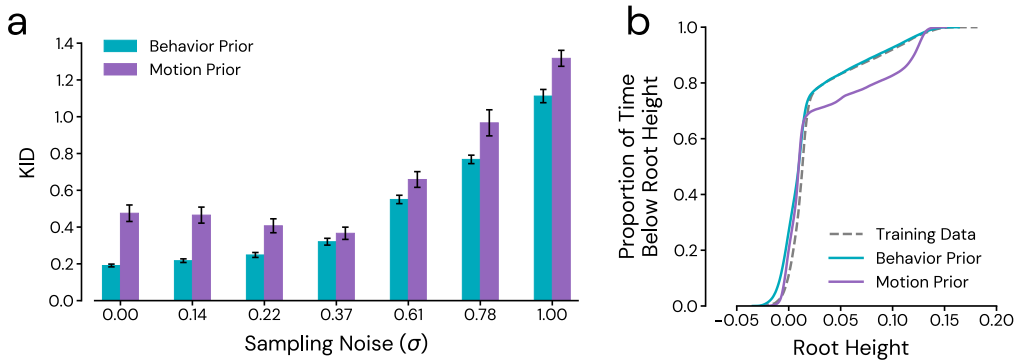


Figure 3: **SCAMPER preserves motion realism during random behavior. (a)** Kernel Inception Distance (KID) between random rollouts and the training data behavior distributions across sampling noise $\sigma = \exp(\text{logvar}/2)$, for the behavior prior (*blue*) and motion prior (*purple*). Data shown is means ± 1 s.d. **(b)** Empirical cumulative distribution function (ECDF) of root height at $\sigma = 0.22$, compared against the training data (*dashed gray*).

training data as the inception model, then computed the KID between the VAE features of the training and generated trajectories. Across all noise levels, the KID was lower for the behavior prior than the motion prior (Fig. 3a). This confirms that the behavior prior produces more natural movements during random generation than the motion prior.

For a more concrete measure of motion realism, we examined the rearing behavior of the virtual rat by measuring the proportion of time spent at various body heights. The behavior prior more closely matched the training data's rearing distribution, while the motion prior diverged from it (Fig. 3b). This held across different levels of noise (Fig. 10), and was quantitatively confirmed using the Wasserstein distance of the rearing curves between the generated data and the training data (Table 9). Together with Section 4.1, these results show that the behavior prior can generate diverse and realistic rat motion during random sampling.

4.3 The motion prior helps produce realistic gaits at different speeds during transfer

Having confirmed that the behavior prior enables diverse and realistic random generation, we next tested whether residual control of the motion prior enables effective transfer to RL tasks. We began with a dense-reward locomotion task: at every timestep, the agent received a reward that decreased linearly with the difference between the target and actual speed. We trained separate policy networks for each target speed. SCAMPER closely matched the target speed across a range of realistic rat locomotion speeds (Fig. 4a, *blue line*) whereas MIMIC-MJX failed to generalize beyond speeds seen in the training data (Fig. 4a, *red line*). A model trained from scratch (as in Merel et al. [4]) also matched target speeds but with deeply unnatural motion and higher variance in its performance (Fig. 4a, *gray line*).

To better understand the resulting movement we examined the gaits at different speeds. Qualitatively, SCAMPER produced distinct gaits resembling walking, trotting, and galloping in real rats (Fig. 4b). Notably, the training data—collected from freely behaving rats in a circular arena—contained no sustained or high-speed locomotion, and thus no trot or gallop. The emergence of these gaits in SCAMPER suggests it generalizes out-of-distribution while maintaining natural locomotion.

To verify this, we performed a quantitative gait analysis, as is standard in behavioral experiments [40, 41], comparing SCAMPER, MIMIC-MJX, and the from-scratch model against real rat data [39]. For each gait (walk, trot, gallop), SCAMPER matched real rat profiles both qualitatively (Fig. 4c) and quantitatively (Table 1). Gait profiles of MIMIC-MJX and the from-scratch model significantly diverged from the real rats (Fig. 13). Notably, MIMIC-MJX could not match target speeds above 0.5 m/s, so the higher-speed gaits (trot, gallop) never emerged. Together, these results show that residual control of the motion prior enables effective task transfer with natural movement

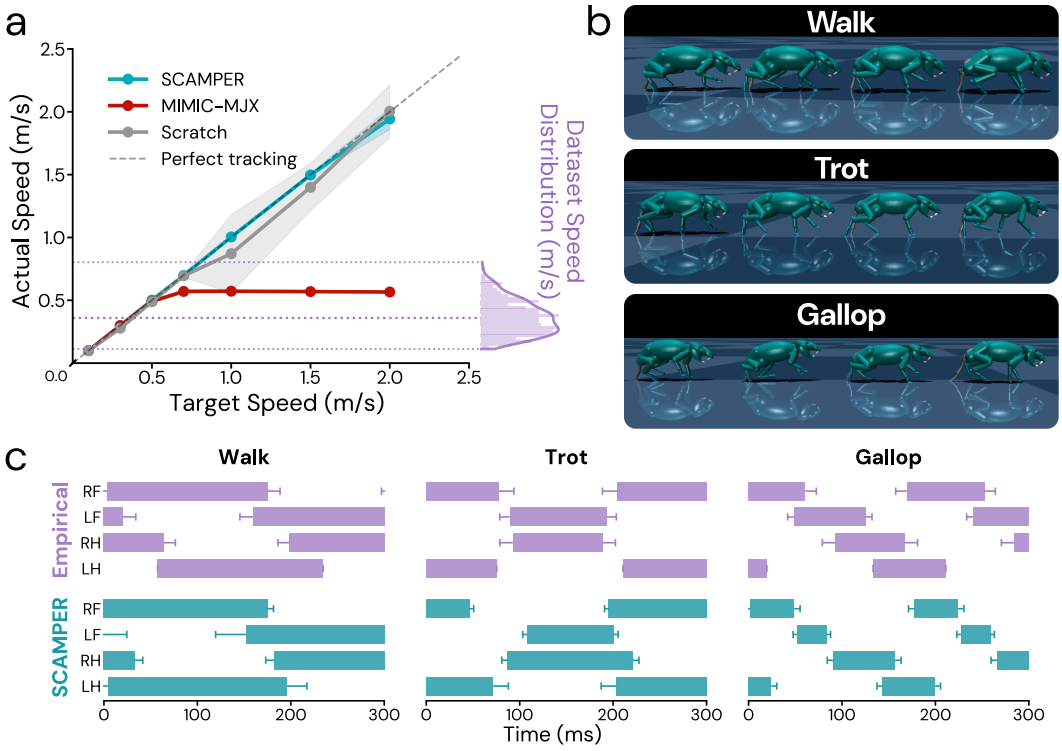


Figure 4: **SCAMPER produces realistic gaits at varying speeds. (a)** Actual vs. target speed for SCAMPER (*blue*), MIMIC-MJX (*red*), and the from-scratch model (*gray*); lines show means, shaded bands show ± 1 s.d. across seeds, dashed line indicates perfect tracking. The training-data speed distribution is shown in purple on the right. **(b)** Renderings of SCAMPER walking, trotting, and galloping at increasing target speeds. **(c)** Stance-phase plots for each limb (RF, LF, RH, LH) across walk, trot, and gallop gaits, comparing empirical rat data (*purple*; data from [39]) to SCAMPER (*blue*); bars show stance duration, error bars show variability across strides.

Table 1: Per-metric distance to empirical rat data, computed as the absolute difference in means divided by empirical standard deviation, $|\mu_{emp} - \mu_{virtual}|/\sigma_{\text{emp}}$ ($\downarrow$). Values shown are the average across metrics in each category: Coordination (temporal symmetry, inter-limb phase couplings), Pattern (phase couplings, duty factors), and All (18 gait metrics). (See Table 11 for all metrics)

	Walk			Trot			Gallop		
Method	Coord.	Pattern	All	Coord.	Pattern	All	Coord.	Pattern	All
SCAMPER	**1.49**	**0.95**	**0.59**	**1.27**	**1.40**	**1.19**	**1.35**	**2.29**	**1.91**
MIMIC-MJX	1.83	1.68	1.28	5.15	3.62	2.72	1.96	2.51	2.74
Scratch	4.83	6.55	3.44	7.84	14.69	11.01	5.93	8.28	5.91

and out-of-distribution gait generalization (see SuppVid[5-14] for videos demonstrating these results; App. C).

4.4 The motion prior expands the stability and precision of rearing behaviors

To verify that these results extend beyond locomotion, we tested a dense-reward rearing task in which the virtual rat must lift its head to a target location (Fig. 5a). The reward decreases linearly with the between the skull and the target, and a one-time bonus reward is obtained for holding at the target for a full second. We set the target at the maximum rearing height observed in the training data to test each model's ability to use the full range of motion of the virtual body.

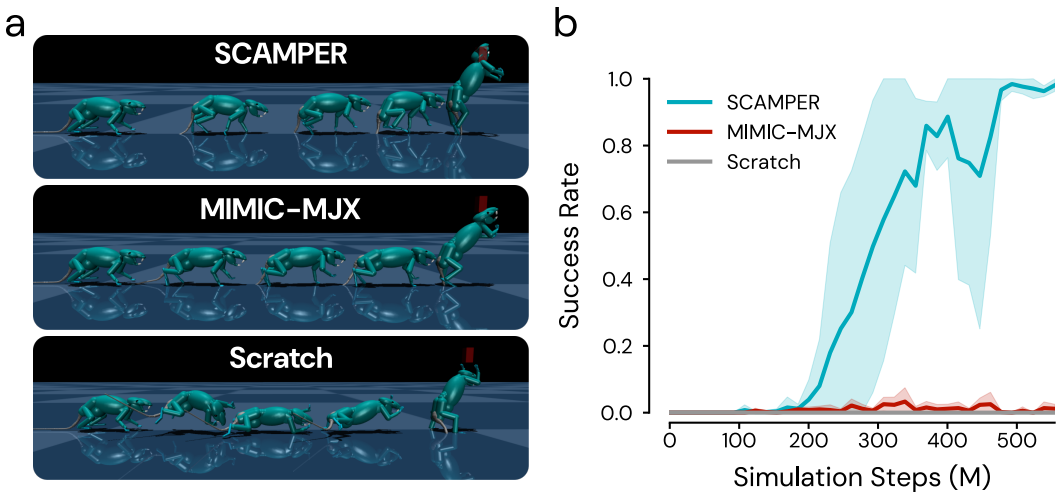


Figure 5: **SCAMPER expands the precision and stability of learnable rearing behaviors. (a)** Example rollouts of a walk-then-rear policy, with the target rearing height indicated by the red box, for policies trained using SCAMPER (*top*), MIMIC-MJX (*middle*), and trained from scratch (*bottom*). **(b)** Task success rate over simulation steps for each method; lines show the mean across seeds and bands show ± 1 s.d. Only the SCAMPER policy solves the task consistently within the training budget.

SCAMPER eventually achieved a high success rate (Fig. 5b, *blue line*). In contrast, MIMIC-MJX rarely held the target height long enough (Fig. 5b, *red line*), and the from-scratch model never reached the target, unable to use the full dynamic range of the rat body in a stable manner (Fig. 5b, *gray line*). Thus, SCAMPER enables effective task transfer with realistic motion for both locomotion and rearing (see SuppVid[15-17] for videos demonstrating these results; App. C).

4.5 Adaptive prior gating balances exploration and exploitation in a sparse-reward foraging task

Finally, we tested whether SCAMPER's gating mechanism could switch between the behavior and motion priors to balance exploration and exploitation in a sparse-reward task. We chose a foraging task relevant to behavioral neuroscience: the virtual rodent must explore a circular arena and navigate to goal locations to collect treats (Fig. 6a). In each trial, the rodent spawned at the center of the arena. To force exploration, treats spawned only outside a minimum radius. Each treat emitted a Gaussian "scent" field, with the rodent receiving the local odor gradient as a state variable. Collecting a treat (head overlapping its location) yielded reward and removed both the treat and its scent.

We compared the full SCAMPER model (Fig. 6b, *blue line*) against a motion-prior-only variant (Fig. 6b, *purple line*), MIMIC-MJX (Fig. 6b, *red line*), and the from-scratch model (Fig. 6b, *gray line*). All models found treats at small spawn radii, where little exploration was required. However, only the full SCAMPER model with the ability to switch to random generation via the behavior prior (Fig. 12) was capable of solving the task when the minimum spawn radius was large and extensive exploration was necessary. This demonstrates that combining the behavior and motion priors enables SCAMPER to train on realistic sparse-reward tasks that require effective exploration (see SuppVid[18-22] for videos demonstrating these results; App. C).

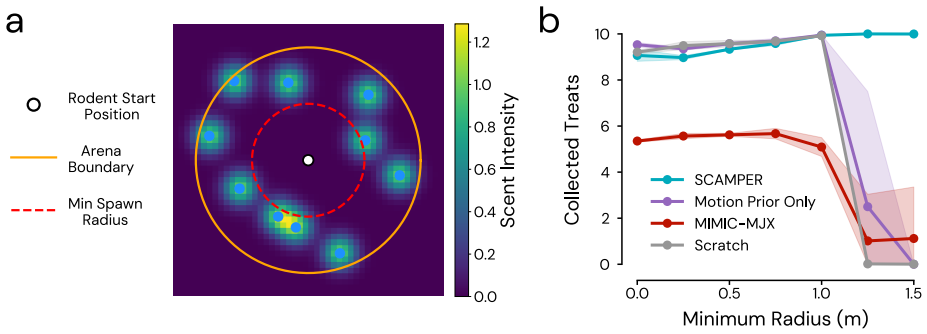


Figure 6: **SCAMPER enables efficient foraging under increasingly sparse reward. (a)** Arena forage task: treats (*blue circles*) are spawned uniformly within an annulus bounded by the minimum spawn radius (*red dashed*) and the arena boundary (*orange*), with a scent field (colorbar) guiding the rodent from its start position (*white*). **(b)** Number of treats collected per episode as a function of the minimum spawn radius, for policies trained using SCAMPER (*blue*), a motion-prior-only variant (*purple*), MIMIC-MJX (*red*), and trained from scratch (*gray*). Lines show the mean across seeds and bands show ± 1 s.d.

5 Discussion

Summary. In this paper, we introduced SCAMPER, a system for training a realistic rat body on realistic behavioral tasks. SCAMPER enables natural movement generation during both goal-directed behavior and random exploration, thanks to its dual-prior architecture. We demonstrated that the behavior prior expands the diversity of randomly sampled motion, while preserving motion realism. We also showed that through residual control of the motion prior, SCAMPER can produce realistic gaits at different speeds and fully leverage its rearing capabilities. Furthermore, we demonstrated that by combining both priors the virtual rodent can solve more difficult sparse-reward tasks that demand exploration.

Significance for neuro-AI. SCAMPER is designed primarily as a tool for neuro-AI. It builds on established techniques for imitation, latent skill learning, and residual control; its additions, a behavior prior distilled from the frozen encoder and a learned gate between the two priors, extend naturalistic motion from goal-directed control to exploration. Naturalistic exploration matters for neuroscience: most experimental tasks offer only sparse reward, so models must explore to learn them, and unnatural movement would confound comparisons with neural data. Aldarondo et al. [5] found that the internal representations of a purely imitative virtual rat matched neural activity in rat motor cortex and dorsolateral striatum. SCAMPER makes it possible to extend such comparisons to goal-directed tasks, with models and animals trained on the same task and compared throughout learning.

Limitations & next steps. There are several limitations to this work that should be recognized. First and foremost, the dataset that we used for training the model was very limited. It consisted of only 1 hour of a rodent freely behaving in an open arena. As such, it contained only simple behaviors, namely walking, rearing, sitting still, and grooming. A more extensive dataset with rats engaging in more diverse behaviors, exhibiting longer bouts of consistent locomotion at various speeds, and movement in more diverse landscapes, would be of great benefit. Another key, but missing, component in the system we presented here is the presence of rich, multi-modal sensory inputs. The current version of SCAMPER uses only proprioceptive inputs, and a simple proxy for scent, effectively making the rat blind and deaf. As the field moves towards more sophisticated embodied models and experiments, this limitation will become more pressing to address. Future work can expand SCAMPER with a larger array of realistic sensory inputs. Finally, one key difference between SCAMPER and real motion control in animals is the use of torque actuators rather than muscles. Real animals must learn to control their bodies through activation of muscle fibers, and a good embodied model would likely take this facet of animal behavior into consideration. In principle, SCAMPER is agnostic to the actuator type, so as new virtual bodies get developed, SCAMPER could be applied to muscle-based models.

Acknowledgments and Disclosure of Funding

The authors would like to thank Nanda Krishna, João Sacramento and Guillaume Lajoie for discussions, support, and feedback. AS was supported by a Fonds de recherche du Québec – Nature et technologies (FRQNT) Master's Research Scholarship, a Natural Sciences and Engineering Research Council of Canada (NSERC) Canada Graduate Research Scholarship – Master's (CGRS M), and a UNIQUE Excellence Scholarship. CYZ was supported by the NSF CSGrad4US Fellowship. This work was also supported by a L.I.F.E. Foundation grant to TDP, a Salk Innovation Award to TDP, an NIH BRAIN Initiative grant (U01NS136507) to BPÖ and TDP, a SFARI Investigator Award to BPÖ, and an ARNI award to BPÖ and BAR. This research was enabled in part by compute resources provided by Mila (https://mila.quebec), Calcul Québec (https://www.calculquebec.ca), and the Digital Research Alliance of Canada (https://alliancecan.ca).

Competing interests

BAR is a Research Scientist at Google and AS is a Student Researcher at Google. The authors declare no other competing interests.

References

[1] Melanie Segado, Felipe Parodi, Jordan K. Matelsky, Michael L. Platt, Eva B. Dyer, and Konrad P. Kording. Grounding Intelligence in Movement, July 2025. URL http://arxiv.org/abs/2507.02771. arXiv:2507.02771 [cs].

[2] Anthony Zador, Sean Escola, Blake Richards, Bence Ölveczky, Yoshua Bengio, Kwabena Boahen, Matthew Botvinick, Dmitri Chklovskii, Anne Churchland, Claudia Clopath, James DiCarlo, Surya Ganguli, Jeff Hawkins, Konrad Körding, Alexei Koulakov, Yann LeCun, Timothy Lillicrap, Adam Marblestone, Bruno Olshausen, Alexandre Pouget, Cristina Savin, Terrence Sejnowski, Eero Simoncelli, Sara Solla, David Sussillo, Andreas S. Tolias, and Doris Tsao. Catalyzing next-generation Artificial Intelligence through NeuroAI. *Nature Communications*, 14(1):1597, March 2023. ISSN 2041-1723. doi: 10.1038/s41467-023-37180-x. URL https://www.nature.com/articles/s41467-023-37180-x.

[3] Sibo Wang-Chen and Pavan Ramdya. The embodied brain: Bridging the brain, body, and behavior with neuromechanical digital twins, January 2026. URL http://arxiv.org/abs/2601.08056. arXiv:2601.08056 [q-bio].

[4] Josh Merel, Diego Aldarondo, Jesse Marshall, Yuval Tassa, Greg Wayne, and Bence Ölveczky. Deep neuroethology of a virtual rodent, November 2019. URL http://arxiv.org/abs/1911.09451. Issue: arXiv:1911.09451 arXiv:1911.09451 [q-bio].

[5] Diego Aldarondo, Josh Merel, Jesse D. Marshall, Leonard Hasenclever, Ugne Klibaite, Amanda Gellis, Yuval Tassa, Greg Wayne, Matthew Botvinick, and Bence P. Ölveczky. A virtual rodent predicts the structure of neural activity across behaviours. *Nature*, 632(8025):594–602, August 2024. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-024-07633-4. URL https://www.nature.com/articles/s41586-024-07633-4. Number: 8025.

[6] Charles Y. Zhang, Yuanjia Yang, Aidan Sirbu, Elliott T. T. Abe, Emil Wärnberg, Eric J. Leonardis, Diego E. Aldarondo, Adam Lee, Aaditya Prasad, Jason Foat, Kaiwen Bian, Joshua Park, Rusham Bhatt, Hutton Saunders, Akira Nagamori, Ayesha R. Thanawalla, Kee Wui Huang, Fabian Plum, Hendrik K. Beck, Steven W. Flavell, David Labonte, Blake A. Richards, Bingni W. Brunton, Eiman Azim, Bence P. Ölveczky, and Talmo D. Pereira. MIMIC-MJX: Neuromechanical Emulation of Animal Behavior, November 2025. URL http://arxiv.org/abs/2511.20532. arXiv:2511.20532 [q-bio].

[7] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Transactions on Graphics*, 37(4):1–14, August 2018. ISSN 0730-0301, 1557-7368. doi: 10.1145/3197517.3201311. URL http://arxiv.org/abs/1804.02717. arXiv:1804.02717 [cs].

[8] Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control, January 2019. URL http://arxiv.org/abs/1811.11711. arXiv:1811.11711 [cs].

[9] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control. *ACM Transactions on Graphics*, 40(4):1–20, August 2021. ISSN 0730-0301, 1557-7368. doi: 10.1145/3450626.3459670. URL http://arxiv.org/abs/2104.02180. arXiv:2104.02180 [cs].

[10] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. ASE: Large-Scale Reusable Adversarial Skill Embeddings for Physically Simulated Characters. *ACM Transactions on Graphics*, 41(4):1–17, July 2022. ISSN 0730-0301, 1557-7368. doi: 10.1145/3528223.3530110. URL http://arxiv.org/abs/2205.01906. arXiv:2205.01906 [cs].

[11] Leonard Hasenclever, Fabio Pardo, Raia Hadsell, Nicolas Heess, and Josh Merel. CoMic: Complementary Task Learning & Mimicry for Reusable Skills. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4105–4115. PMLR, November 2020. URL https://proceedings.mlr.press/v119/hasenclever20a.html.

[12] Chen Tessler, Yoni Kasten, Yunrong Guo, Shie Mannor, Gal Chechik, and Xue Bin Peng. CALM: Conditional Adversarial Latent Models for Directable Virtual Characters. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, pages 1–9, July 2023. doi: 10.1145/3588432.3591541. URL http://arxiv.org/abs/2305.02195. arXiv:2305.02195 [cs].

[13] Zhengyi Luo, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris Kitani, and Weipeng Xu. Universal Humanoid Motion Representations for Physics-Based Control, April 2024. URL http://arxiv.org/abs/2310.04582. arXiv:2310.04582 [cs].

[14] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven Exploration by Self-supervised Prediction, May 2017. URL http://arxiv.org/abs/1705.05363. arXiv:1705.05363 [cs].

[15] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, February 2021. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-020-03157-9. URL http://arxiv.org/abs/2004.12919. arXiv:2004.12919 [cs].

[16] Matteo Carandini and Anne K. Churchland. Probing perceptual decisions in rodents. *Nature Neuroscience*, 16(7):824–831, July 2013. ISSN 1546-1726. doi: 10.1038/nn.3410. URL https://www.nature.com/articles/nn.3410.

[17] The International Brain Laboratory, Valeria Aguillon-Rodriguez, Dora Angelaki, Hannah Bayer, Niccolo Bonacchi, Matteo Carandini, Fanny Cazettes, Gaelle Chapuis, Anne K Churchland, Yang Dan, Eric Dewitt, Mayo Faulkner, Hamish Forrest, Laura Haetzel, Michael Häusser, Sonja B Hofer, Fei Hu, Anup Khanal, Christopher Krasniak, Ines Laranjeira, Zachary F Mainen, Guido Meijer, Nathaniel J Miska, Thomas D Mrsic-Flogel, Masayoshi Murakami, Jean-Paul Noel, Alejandro Pan-Vazquez, Cyrille Rossant, Joshua Sanders, Karolina Socha, Rebecca Terry, Anne E Urai, Hernando Vergara, Miles Wells, Christian J Wilson, Ilana B Witten, Lauren E Wool, and Anthony M Zador. Standardized and reproducible measurement of decision-making in mice. *eLife*, 10:e63711, May 2021. ISSN 2050-084X. doi: 10.7554/eLife.63711. URL https://doi.org/10.7554/eLife.63711.

[18] Simon Musall, Matthew T. Kaufman, Ashley L. Juavinett, Steven Gluf, and Anne K. Churchland. Single-trial neural dynamics are dominated by richly varied movements. *Nature Neuroscience*, 22(10):1677–1686, October 2019. ISSN 1546-1726. doi: 10.1038/s41593-019-0502-4. URL https://www.nature.com/articles/s41593-019-0502-4.

[19] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin Riedmiller, and David Silver. Emergence of Locomotion Behaviours in Rich Environments, July 2017. URL http://arxiv.org/abs/1707.02286. arXiv:1707.02286 [cs].

[20] Heyuan Yao, Zhenhua Song, Baoquan Chen, and Libin Liu. ControlVAE: Model-Based Learning of Generative Controllers for Physics-Based Characters. *ACM Transactions on Graphics*, 41(6):1–16, December 2022. ISSN 0730-0301, 1557-7368. doi: 10.1145/3550454.3555434. URL http://arxiv.org/abs/2210.06063. arXiv:2210.06063 [cs].

[21] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. Physics-based character controllers using conditional VAEs. *ACM Trans. Graph.*, 41(4):96:1–96:12, July 2022. ISSN 0730-0301. doi: 10.1145/3528223.3530067. URL https://dl.acm.org/doi/10.1145/3528223.3530067.

[22] Heyuan Yao, Zhenhua Song, Yuyang Zhou, Tenglong Ao, Baoquan Chen, and Libin Liu. MoConVQ: Unified Physics-Based Motion Control via Scalable Discrete Representations, December 2023. URL http://arxiv.org/abs/2310.10198. arXiv:2310.10198 [cs].

[23] Zhengyi Luo, Jinkun Cao, Alexander Winkler, Kris Kitani, and Weipeng Xu. Perpetual Humanoid Control for Real-time Simulated Avatars, September 2023. URL http://arxiv.org/abs/2305.06456. arXiv:2305.06456 [cs].

[24] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J. Guibas. HuMoR: 3D Human Motion Model for Robust Pose Estimation, August 2021. URL http://arxiv.org/abs/2105.04668. arXiv:2105.04668 [cs].

[25] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H. Bermano. Human Motion Diffusion Model, October 2022. URL http://arxiv.org/abs/2209.14916. arXiv:2209.14916 [cs].

[26] Nolan Wagener, Andrey Kolobov, Felipe Vieira Frujeri, Ricky Loynd, Ching-An Cheng, and Matthew Hausknecht. MoCapAct: A Multi-Task Dataset for Simulated Humanoid Control, January 2023. URL http://arxiv.org/abs/2208.07363. arXiv:2208.07363 [cs].

[27] Ilija Radosavovic, Bike Zhang, Baifeng Shi, Jathushan Rajasegaran, Sarthak Kamat, Trevor Darrell, Koushil Sreenath, and Jitendra Malik. Humanoid Locomotion as Next Token Prediction, February 2024. URL http://arxiv.org/abs/2402.19469. arXiv:2402.19469 [cs].

[28] Roman Vaxenburg, Igor Siwanowicz, Josh Merel, Alice A. Robie, Carmen Morrow, Guido Novati, Zinovia Stefanidi, Gert-Jan Both, Gwyneth M. Card, Michael B. Reiser, Matthew M. Botvinick, Kristin M. Branson, Yuval Tassa, and Srinivas C. Turaga. Whole-body physics simulation of fruit fly locomotion. *Nature*, pages 1–3, April 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09029-4. URL https://www.nature.com/articles/s41586-025-09029-4.

[29] Sibo Wang-Chen, Victor Alfred Stimpfling, Thomas Ka Chung Lam, Pembe Gizem Özdil, Louise Genoud, Femke Hurtak, and Pavan Ramdya. NeuroMechFly v2: simulating embodied sensorimotor control in adult Drosophila. *Nature Methods*, 21(12):2353–2362, December 2024. ISSN 1548-7105. doi: 10.1038/s41592-024-02497-y. URL https://www.nature.com/articles/s41592-024-02497-y.

[30] Mengdi Zhao, Ning Wang, Xinrui Jiang, Xiaoyang Ma, Haixin Ma, Gan He, Kai Du, Lei Ma, and Tiejun Huang. An integrative data-driven model simulating C. elegans brain, body and environment interactions. *Nature Computational Science*, 4(12):978–990, December 2024. ISSN 2662-8457. doi: 10.1038/s43588-024-00738-w. URL https://www.nature.com/articles/s43588-024-00738-w.

[31] Reece Keller, Alyn Tornell, Felix Pei, Xaq Pitkow, Leo Kozachkov, and Aran Nayebi. Autonomous Behavior and Whole-Brain Dynamics Emerge in Embodied Zebrafish Agents with Model-based Intrinsic Motivation, May 2025. URL http://arxiv.org/abs/2506.00138. arXiv:2506.00138 [q-bio].

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. URL http://arxiv.org/abs/1707.06347. arXiv:1707.06347 [cs].

[33] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, March 2011. URL http://arxiv.org/abs/1011.0686. arXiv:1011.0686 [cs].

[34] Dan Friedman and Adji Bousso Dieng. The Vendi Score: A Diversity Evaluation Metric for Machine Learning, July 2023. URL http://arxiv.org/abs/2210.02410. arXiv:2210.02410 [cs].

[35] Tom Stanic, Marco Jiralerspong, Xiaofeng Zhang, Danilo Vucetic, and Gauthier Gidel. Beyond reward maximization: Evaluating the diversity of trajectories in reinforcement learning with temporal vendi score. 2026.

[36] Erik M. Lintunen. VendiRL: A Framework for Self-Supervised Reinforcement Learning of Diversely Diverse Skills, October 2025. URL http://arxiv.org/abs/2509.02930. arXiv:2509.02930 [cs].

[37] Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs, January 2021. URL http://arxiv.org/abs/1801.01401. arXiv:1801.01401 [stat].

[38] Sigal Raab, Inbal Leibovitch, Peizhuo Li, Kfir Aberman, Olga Sorkine-Hornung, and Daniel Cohen-Or. MoDi: Unconditional Motion Synthesis from Diverse Data, December 2022. URL http://arxiv.org/abs/2206.08010. arXiv:2206.08010 [cs].

[39] Simon Danner, Courtney Shepard, Casey Hainline, Natalia Shevtsova, Ilya Rybak, and David Magnuson. *Spinal control of locomotion before and after spinal cord injury*. June 2023. doi: 10.1101/2023.03.22.533794. Journal Abbreviation: bioRxiv : the preprint server for biology Publication Title: bioRxiv : the preprint server for biology.

[40] Frank P.T. Hamers, Alex J. Lankhorst, Teus Jan van Laar, Wouter B. Veldhuis, and Willem Hendrik Gispen. Automated Quantitative Gait Analysis During Overground Locomotion in the Rat: Its Application to Spinal Cord Contusion and Transection Injuries. *Journal of Neurotrauma*, 18(2):187–201, February 2001. ISSN 0897-7151. doi: 10.1089/08977150150502613. URL https://journals.sagepub.com/action/showAbstract.

[41] César S. Mendes, Imre Bartos, Zsuzsanna Márka, Turgay Akay, Szabolcs Márka, and Richard S. Mann. Quantification of gait parameters in freely walking rodents. *BMC Biology*, 13(1):50, July 2015. ISSN 1741-7007. doi: 10.1186/s12915-015-0154-0. URL https://doi.org/10.1186/s12915-015-0154-0.

[42] Timothy W Dunn, Jesse D Marshall, Kyle S Severson, Diego E Aldarondo, David G C Hildebrand, Selmaan N Chettih, William L Wang, Amanda J Gellis, David E Carlson, Dmitriy Aronov, Winrich A Freiwald, Fan Wang, and Bence P Ölveczky. Geometric deep learning enables 3D kinematic profiling across species and environments. *Nat. Methods*, 18(5):564–573, May 2021.

[43] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, October 2012.

[44] MuJoCo XLA Authors. MuJoCo XLA (MJX). https://mujoco.readthedocs.io/en/stable/mjx.html.

[45] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[46] Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. MuJoCo playground. *arXiv [cs.RO]*, February 2025.

[47] C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – a differentiable physics engine for large scale rigid body simulation. *arXiv [cs.RO]*, June 2021.

[48] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, May 2018. doi: 10.1109/ICRA.2018.8460528. URL http://arxiv.org/abs/1710.06537. arXiv:1710.06537 [cs].

[49] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, January 2018. URL http://arxiv.org/abs/1706.08500. arXiv:1706.08500 [cs].

[50] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the Continuity of Rotation Representations in Neural Networks, June 2020. URL http://arxiv.org/abs/1812.07035. arXiv:1812.07035 [cs].

[51] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. ISSN 1533-7928. URL http://jmlr.org/papers/v13/gretton12a.html.

A Proofs

We restate each proposition from Section 3.3 and provide the full proof.

Proposition 1 (Information bottleneck under joint training)**.** *When the encoder $q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x})$ and the state-dependent prior $p_\psi(\mathbf{z}|\mathbf{x})$ are trained jointly, minimizing their expected KL divergence decomposes as*

$$\min_{\phi,\psi} \mathbb{E}_{p(\mathbf{x},\mathbf{g})} \left[D_{\mathrm{KL}}(q_\phi \parallel p_\psi) \right] = \min_{\phi,\psi} \left[I_\phi(Z; G \mid X) + \mathbb{E}_{p(\mathbf{x})} \left[D_{\mathrm{KL}}(m_\phi \parallel p_\psi) \right] \right], \tag{8}$$

where $m_\phi(\mathbf{z}|\mathbf{x}) = \mathbb{E}_{p(\mathbf{g}|\mathbf{x})}[q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x})]$ is the marginal posterior induced by the encoder, which evolved during joint training.

Proof. Let the joint optimization objective be the expected KL divergence over the data distribution:

$$\min_{\phi,\psi} \mathcal{L}(\phi,\psi) = \min_{\phi,\psi} \mathbb{E}_{p(\mathbf{x},\mathbf{g})} \left[D_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x}) \parallel p_\psi(\mathbf{z}|\mathbf{x})\big) \right]. \tag{9}$$

Expanding the expected KL divergence into its integral form:

$$\min_{\phi,\psi} \mathcal{L}(\phi,\psi) = \min_{\phi,\psi} \int p(\mathbf{x}) \int p(\mathbf{g}|\mathbf{x}) \int q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x})}{p_\psi(\mathbf{z}|\mathbf{x})} \, d\mathbf{z} \, d\mathbf{g} \, d\mathbf{x}. \tag{10}$$

Define the marginal posterior (which depends on the encoder parameters ϕ):

$$m_\phi(\mathbf{z}|\mathbf{x}) \triangleq \int q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x}) p(\mathbf{g}|\mathbf{x}) \, d\mathbf{g}. \tag{11}$$

Multiplying numerator and denominator inside the logarithm by $m_\phi(\mathbf{z}|\mathbf{x})$:

$$\log \frac{q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x})}{p_\psi(\mathbf{z}|\mathbf{x})} = \log \frac{q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x})}{p_\psi(\mathbf{z}|\mathbf{x})} \frac{m_\phi(\mathbf{z}|\mathbf{x})}{m_\phi(\mathbf{z}|\mathbf{x})} = \log \frac{q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x})}{m_\phi(\mathbf{z}|\mathbf{x})} + \log \frac{m_\phi(\mathbf{z}|\mathbf{x})}{p_\psi(\mathbf{z}|\mathbf{x})}. \tag{12}$$

Substituting this decomposition back into the integral yields two terms:

$$\begin{aligned} \min_{\phi,\psi} \mathcal{L}(\phi,\psi) = \min_{\phi,\psi} \Bigg[& \underbrace{\int p(\mathbf{x}) \int p(\mathbf{g}|\mathbf{x}) \int q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x})}{m_\phi(\mathbf{z}|\mathbf{x})} \, d\mathbf{z} \, d\mathbf{g} \, d\mathbf{x}}_{\text{Term 1}} \\ & + \underbrace{\int p(\mathbf{x}) \int p(\mathbf{g}|\mathbf{x}) \int q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x}) \log \frac{m_\phi(\mathbf{z}|\mathbf{x})}{p_\psi(\mathbf{z}|\mathbf{x})} \, d\mathbf{z} \, d\mathbf{g} \, d\mathbf{x}}_{\text{Term 2}} \Bigg]. \end{aligned} \tag{13}$$

Term 1 is, by definition, the conditional mutual information between the latent $\mathbf{z}$ and the goal $\mathbf{g}$ given the state $\mathbf{x}$:

$$\text{Term 1} = I_\phi(Z; G \mid X). \tag{14}$$

Term 2: Since $\log \frac{m_\phi(\mathbf{z}|\mathbf{x})}{p_\psi(\mathbf{z}|\mathbf{x})}$ is independent of $\mathbf{g}$, and the integrand is absolutely integrable[1], we apply Fubini's theorem to evaluate the integral over $\mathbf{g}$ first:

$$\begin{aligned} \text{Term 2} &= \int p(\mathbf{x}) \int \underbrace{\left(\int q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x}) p(\mathbf{g}|\mathbf{x}) \, d\mathbf{g} \right)}_{m_\phi(\mathbf{z}|\mathbf{x})} \log \frac{m_\phi(\mathbf{z}|\mathbf{x})}{p_\psi(\mathbf{z}|\mathbf{x})} \, d\mathbf{z} \, d\mathbf{x} \\ &= \mathbb{E}_{p(\mathbf{x})} \left[D_{\mathrm{KL}}\big(m_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\psi(\mathbf{z}|\mathbf{x})\big) \right]. \end{aligned} \tag{15}$$

Combining the two terms gives the desired decomposition:

$$\min_{\phi,\psi} \mathcal{L}(\phi,\psi) = \min_{\phi,\psi} \left[I_\phi(Z; G \mid X) + \mathbb{E}_{p(\mathbf{x})} \left[D_{\mathrm{KL}}\big(m_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\psi(\mathbf{z}|\mathbf{x})\big) \right] \right]. \tag{16}$$

□

[1] Absolute integrability follows from the assumption that $D_{\mathrm{KL}}(q_\phi || p_\psi) < \infty$ on the support of $p(\mathbf{x},\mathbf{g})$.

Proposition 2 (Marginal posterior approximation under frozen encoder)**.** *When the encoder $q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x})$ is frozen, minimizing the expected KL divergence between the encoder and a state-dependent prior $p_\varphi(\mathbf{z}|\mathbf{x})$ is equivalent to*

$$\min_\varphi \mathbb{E}_{p(\mathbf{x},\mathbf{g})}\left[D_{\mathrm{KL}}(q_\phi \parallel p_\varphi)\right] = I_\phi(Z;G \mid X) + \min_\varphi \mathbb{E}_{p(\mathbf{x})}\left[D_{\mathrm{KL}}(m_\phi \parallel p_\varphi)\right], \tag{17}$$

where $m_\phi(\mathbf{z}|\mathbf{x}) = \mathbb{E}_{p(\mathbf{g}|\mathbf{x})}[q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x})]$ is the marginal posterior induced by the frozen encoder, held fixed during optimization. The conditional mutual information, $I_\phi(Z;G \mid X)$, does not depend on the optimization variable φ; it is therefore an additive constant that sits outside the minimization. Hence, the optimal behavior prior is the closest state-conditional approximation to the marginal posterior m_ϕ.

Proof. Let the distillation objective be the expected KL divergence over the data distribution:

$$\min_\varphi \mathcal{L}(\varphi) = \min_\varphi \mathbb{E}_{p(\mathbf{x},\mathbf{g})}\left[D_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x}) \parallel p_\varphi(\mathbf{z}|\mathbf{x})\big)\right]. \tag{18}$$

Expanding the inner KL divergence as the difference between cross-entropy and entropy:

$$D_{\mathrm{KL}}(q_\phi \parallel p_\varphi) = \underbrace{-\int q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x}) \log p_\varphi(\mathbf{z}|\mathbf{x})\, d\mathbf{z}}_{H(q_\phi,p_\varphi)} + \underbrace{\int q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x})\, d\mathbf{z}}_{-H(q_\phi)}. \tag{19}$$

Since the encoder parameters ϕ are frozen, the entropy $H(q_\phi)$ is constant with respect to φ and can be pulled outside the minimization:

$$\min_\varphi \mathcal{L}(\varphi) = \min_\varphi \left[\mathbb{E}_{p(\mathbf{x},\mathbf{g})}[H(q_\phi,p_\varphi)]\right] - \mathbb{E}_{p(\mathbf{x},\mathbf{g})}[H(q_\phi)]. \tag{20}$$

First we focus on the first term inside the minimization and expand it out:

$$\mathbb{E}_{p(\mathbf{x},\mathbf{g})}[H(q_\phi,p_\varphi)] = -\int p(\mathbf{x}) \int p(\mathbf{g}|\mathbf{x}) \int q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x}) \log p_\varphi(\mathbf{z}|\mathbf{x})\, d\mathbf{z}\, d\mathbf{g}\, d\mathbf{x}. \tag{21}$$

Assuming absolute integrability, we apply Fubini's theorem to swap the order of integration between $\mathbf{g}$ and $\mathbf{z}$, and factor out $\log p_\varphi(\mathbf{z}|\mathbf{x})$ since it is independent of $\mathbf{g}$:

$$\begin{aligned}
\mathbb{E}_{p(\mathbf{x},\mathbf{g})}[H(q_\phi,p_\varphi)] &= -\int p(\mathbf{x}) \int \log p_\varphi(\mathbf{z}|\mathbf{x}) \underbrace{\left(\int q_\phi(\mathbf{z}|\mathbf{g},\mathbf{x}) p(\mathbf{g}|\mathbf{x})\, d\mathbf{g}\right)}_{m_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z}\, d\mathbf{x} \\
&= -\int p(\mathbf{x}) \int m_\phi(\mathbf{z}|\mathbf{x}) \log p_\varphi(\mathbf{z}|\mathbf{x})\, d\mathbf{z}\, d\mathbf{x} \\
&= \mathbb{E}_{p(\mathbf{x})}[H(m_\phi,p_\varphi)].
\end{aligned} \tag{22}$$

This is the expectation of the cross-entropy between m_ϕ and p_φ. Substituting back into equation (20) yields:

$$\min_\varphi \mathcal{L}(\varphi) = \min_\varphi \left[\mathbb{E}_{p(\mathbf{x})}[H(m_\phi,p_\varphi)]\right] - \mathbb{E}_{p(\mathbf{x},\mathbf{g})}[H(q_\phi)]. \tag{23}$$

To relate it to the KL divergence, we add and subtract the expectation of the entropy of the marginal posterior:

$$\min_\varphi \mathcal{L}(\varphi) = \min_\varphi \left[\mathbb{E}_{p(\mathbf{x})}[H(m_\phi,p_\varphi)]\right] - \mathbb{E}_{p(\mathbf{x})}[H(m_\phi)] + \mathbb{E}_{p(\mathbf{x})}[H(m_\phi)] - \mathbb{E}_{p(\mathbf{x},\mathbf{g})}[H(q_\phi)]. \tag{24}$$

Pulling $\mathbb{E}_{p(\mathbf{x})}[H(m_\phi)]$ into the minimization operation and using the identity $H(m_\phi,p_\varphi) = D_{\mathrm{KL}}(m_\phi||p_\varphi) + H(m_\phi)$:

$$\min_\varphi \mathcal{L}(\varphi) = \min_\varphi \mathbb{E}_{p(\mathbf{x})}\left[D_{\mathrm{KL}}(m_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\varphi(\mathbf{z}|\mathbf{x}))\right] + \Big(\mathbb{E}_{p(\mathbf{x})}[H(m_\phi)] - \mathbb{E}_{p(\mathbf{x},\mathbf{g})}[H(q_\phi)]\Big). \tag{25}$$

The bracketed difference of entropies is, by definition, the conditional mutual information between the latent and the goal given the state under the frozen encoder, therefore:

$$\min_\varphi \mathcal{L}(\varphi) = \min_\varphi \mathbb{E}_{p(\mathbf{x})}\left[D_{\mathrm{KL}}(m_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\varphi(\mathbf{z}|\mathbf{x}))\right] + I_\phi(Z;G \mid X). \tag{26}$$

□

B Dataset

Motion capture. We use a rat motion dataset collected by Aldarondo et al. [5]. The dataset is composed of 842 5 s clips of rats freely behaving in a circular enclosure. The mocap data is characterized by 23 keypoints captured at 50 Hz using DANNCE v1.3 [42] from multicamera video. The dataset contains various kinds of walking, rearing, grooming and prone behaviors. The mocap keypoints are aligned to the virtual rodent body model through STAC-MJX [6].

Empirical rat locomotion data. The empirical data used in the gait analysis (phase differences, duty factors, and gait classifications) were obtained from Danner et al. [39], publicly available at https://github.com/dannerlab/rat-sci-locomotion under a Creative Commons Attribution–ShareAlike 4.0 International license (CC BY–SA 4.0, https://creativecommons.org/licenses/by-sa/4.0/). The data was used unmodified.

C Supplementary videos

We host the supplementary videos demonstrating the rodent performing the various tasks described in this work on the project's page (https://aidansirbu.com/SCAMPER-site/). Specific videos are also referenced throughout the text: "SuppVid<x>" refers to the corresponding video on the website and the .mp4 file in the supplementary video package.

D Implementation & compute

Implementation. SCAMPER is implemented on top of the MuJoCo physics engine [43]. We accelerate simulation using MuJoCo XLA (MJX) [44], a JAX-based [45] reimplementation of MuJoCo, which enables GPU-vectorized physics rollouts and domain randomization to be applied directly to the per-environment parameters. Environments are built on top of MuJoCo Playground's [46] base class and wrappers, which expose MJX simulation through a JAX-compatible interface and adapt our environments to the Brax [47] training API. RL is performed with Brax, whose PPO implementation and distributed training utilities we extend to support our networks and distillation objectives. Pretraining (Stages 1-3) was performed using the JAX backend of MJX and task transfer was performed using the Warp backend of MJX. MIMIC-MJX baselines were implemented using the training setup and hyperparameters described in Zhang et al. [6]. The from-scratch baseline uses position (PD) actuators, whereas all other models use torque actuators. Torque actuators are highly unstable and difficult to learn from scratch, so we give the from-scratch model PD actuators to give it a fair chance to compete.

Compute. All experiments were run on a single GPU with 4 CPU cores and 48 GB of host RAM. Pretraining used NVIDIA L40S (48 GB) GPUs: the Stage 1 expert imitator trains for 3×10^9 environment steps over 4096 parallel envs in ≈ 14 h, the Stage 2 distillation of the encoder/decoder and motion prior trains for 1×10^9 steps over 4096 envs in ≈ 12 h, and the Stage 3 behavior prior trains for 2×10^9 steps over 4096 envs in ≈ 30 h, for a total of roughly 56 L40S-hours per end-to-end pretraining run. The three task-transfer benchmarks were trained with PPO under MuJoCo Warp on NVIDIA A100 (80 GB) GPUs, with four seeds per condition. Walk-then-rear was run for the SCAMPER, MIMIC-MJX, and from-scratch models, with each job training for 1×10^9 steps over 512 envs in ≈ 11 h (12 jobs total, ≈ 132 A100-hours). Locomotion was run for the same three models swept over 7 target speeds, with each job training for 4×10^8 steps over 1024 envs in ≈ 10 h (84 jobs total, ≈ 840 A100-hours). Arena foraging was run for SCAMPER, a motion-prior-only variant, MIMIC-MJX, and the from-scratch model swept over 7 minimum spawn radii, with each job training for 2×10^9 steps over 1024 envs in ≈ 6 h (112 jobs total, ≈ 672 A100-hours). One full reproduction of the paper's training pipeline therefore costs roughly 56 L40S-GPU-hours and $1{,}650$ A100-GPU-hours.

E Tasks

E.1 Stage 1: Expert imitator

Task. The expert imitator $\pi_I(\mathbf{a}_t|\mathbf{x}_t, \mathbf{g}_t)$ tracks 3D motion-capture trajectories of freely behaving rats using the biomechanical rat model of Merel et al. [4]. We use a $\sim$1 h subset of the open-source rat mocap dataset released by Aldarondo et al. [5] (842 clips of 5 s mocap), registered to the virtual rat via STAC-MJX [6]. At the start of each episode we sample a clip uniformly from the training partition (80% of the dataset; the remaining 20% is held out for evaluation rollouts) and a starting frame uniformly in $[0, 50]$, with zero initial joint velocity. Episodes last up to 250 reference frames. At each control step the agent observes the proprioception vector $\mathbf{x}_t$ and the egocentric goal $\mathbf{g}_t$ defined in Section 3.1, where $\mathbf{g}_t$ is computed from a 5-frame reference horizon, and emits torques $\mathbf{a}_t \in \mathbb{R}^{38}$ applied to the rat's actuated joints. Control runs at 100 Hz on top of a 500 Hz MuJoCo MJX physics simulation, with a 50 Hz mocap reference. The body model uses torque actuators and a $0.9\times$ rescaling factor.

Reward. Each imitation tracking term takes the form $r_i = w_i \exp\big(-\frac{1}{2}(e_i/\sigma_i)^2\big)$, a Gaussian kernel of the per-term error e_i with characteristic scale σ_i; values of (w_i, σ_i) are tabulated in Table 2. The four active tracking terms penalize errors in root position, root orientation (quaternion geodesic), joint angles, and egocentric Cartesian end-effector positions. Three regularization costs penalize quadratic control magnitude, quadratic control differences (jerk), and clipped torque–velocity product, and a torso-z-range term adds an in-range bonus that rewards keeping the torso height within the healthy range $[0.0325, 0.5]$ m.

Termination. An episode terminates early if (i) the root position deviates from the reference by more than 0.1 m, (ii) the root orientation deviates by more than 60°, (iii) the pose L_2 error exceeds 4.5, or (iv) any state value becomes `NaN`. Episodes also terminate at the end of the reference clip.

Domain randomization. To encourage robust control across variations in the biomechanical model, at the start of each episode we sample physics parameters uniformly within fixed ranges and hold them fixed for the duration of the episode [48]. Randomized parameters include floor friction, static friction scale, joint armature scale, link mass scale, torso mass jitter, center-of-mass jitter, and an additive jitter on the initial joint configuration $\boldsymbol{\theta}_0$; full ranges are listed in Table 2.

Network architecture. The expert imitator is a multi-layer perceptron with no latent bottleneck: the concatenation of $\mathbf{x}_t$ and $\mathbf{g}_t$, normalized by their running mean and variance, is fed directly to a $[2048, 1024, 1024, 512]$ policy MLP that outputs the mean and log-standard-deviation of a tanh-squashed diagonal Gaussian over the 38-dimensional action, so that $\mathbf{a}_t \in [-1, 1]^{38}$. The critic is a separate $[2048, 1024, 1024, 1024, 1024, 512]$ MLP estimating the state-value function. Both networks use SiLU activations with LayerNorm between hidden layers, and are trained from random initialization using LeCun initialization with zero bias.

Training. We train π_I with PPO [32] for 3×10^9 environment steps across $4{,}096$ parallel environments. Each PPO update uses a 20-step unroll, batch size $1{,}024$, and 4 epochs of 16 minibatches per batch, with Adam at learning rate 10^{-4} and gradient clipping at 20.0. We use discount $\gamma = 0.95$, PPO clip range $\varepsilon = 0.2$, entropy coefficient 10^{-2}, value-loss coefficient 0.25, and reward scaling 1. Evaluation rollouts on the held-out clip partition are performed every 2.5×10^7 environment steps with deterministic (mean) actions. Training is implemented in JAX [45] with MuJoCo MJX [44] and runs on a single GPU; the full list of training hyperparameters is given in Table 2.

E.2 Stage 2: Online distillation

Task. Online distillation reuses the imitation environment from Stage 1 (App. E.1): identical clip set and train/eval split, control rate, physics solver, reward terms, termination criteria, and domain randomization (Table 2). The Stage 1 expert imitator π_I is loaded as a frozen teacher and queried for target actions $\mathbf{a}_t$ on every visited state, while a student latent skill controller—the encoder $q_\phi(\mathbf{z}_t|\mathbf{x}_t, \mathbf{g}_t)$, decoder $p_\theta(\mathbf{a}_t|\mathbf{x}_t, \mathbf{z}_t)$, and proprioception-conditioned motion prior $p_\psi(\mathbf{z}_t|\mathbf{x}_t)$—is rolled out in simulation and trained jointly to match the teacher, using a similar DAgger [33] schedule as PULSE [13]. Episode terminations and rewards are inherited from the underlying environment but are not used as a learning signal: the student is supervised against the teacher's actions, not the imitation reward.

Loss. The student minimizes the composite distillation objective

$$\mathcal{L}_{\text{distill}} = \mathcal{L}_{\text{DAgger}} + \lambda \mathcal{L}_{\text{KL}} + \alpha \mathcal{L}_{\text{AR}},$$

where $\mathcal{L}_{\text{KL}} = D_{\text{KL}}(q_\phi || p_\psi)$ regularizes the encoder toward the motion prior and $\mathcal{L}_{\text{AR}} = ||\boldsymbol{\mu}_t^e - \boldsymbol{\mu}_{t-1}^e||_2^2$ promotes temporal smoothness in the encoder mean. In practice $\mathcal{L}_{\text{DAgger}}$ is computed on the policy's action-distribution parameters rather than on sampled actions: both the teacher MLP π_I and the student decoder p_θ output the mean and log-standard-deviation of a tanh-squashed diagonal Gaussian over the action, and we take the MSE between these parameter vectors. This reduces to the action-matching form $\mathcal{L}_{\text{DAgger}} = ||\mathbf{a}_t - \hat{\mathbf{a}}_t||_2^2$ given in the main text when both policies are evaluated deterministically (i.e., at their distribution means), and is a policy-distribution distillation in the spirit of DAgger [33] more generally. The KL coefficient λ is annealed linearly from 10^{-3} to 10^{-4} over the first 25% of training and held fixed thereafter; the AR(1) coefficient is fixed at $\alpha = 5 \times 10^{-3}$. To prevent posterior collapse and runaway sampling noise, the encoder and prior log-variances are clipped to $[-5, 0]$ (matching PULSE [13]).

Network architecture. All three networks are MLPs with SiLU activations and LayerNorm between hidden layers (matching the Stage 1 policy). The encoder maps the concatenation of $\mathbf{x}_t$ and $\mathbf{g}_t$ through hidden layers $[2048, 1024, 1024]$ to per-dimension means and log-variances of a 16-dimensional Gaussian latent $\mathbf{z}_t$; we sample $\mathbf{z}_t$ via the standard reparameterization trick. The decoder maps the concatenation of $\mathbf{z}_t$ and $\mathbf{x}_t$ through $[1024, 1024, 512]$ to an action prediction $\hat{\mathbf{a}}_t$. The proprioception-conditioned motion prior maps $\mathbf{x}_t$ through $[1536, 1024, 512]$ to the mean and log-variance of $p_\psi(\mathbf{z}_t|\mathbf{x}_t)$. Encoder, decoder, and motion prior are randomly initialized using LeCun initialization with zero bias and trained end-to-end; the teacher π_I remains frozen throughout.

Training. We collect $4{,}096$ parallel student rollouts in MuJoCo MJX with the same 20-step unroll length used in Stage 1, train for 10^9 environment steps with batch size $1{,}024$ split across 16 minibatches and 4 epochs per batch, and update (ϕ, θ, ψ) jointly with Adam at learning rate 10^{-4} and global gradient-norm clipping at 10.0. Observations are normalized by their running mean and variance, with statistics inherited from the Stage 1 normalizer to ensure consistency with the frozen teacher. Evaluation rollouts are run every 2×10^7 environment steps with the deterministic decoder mean. The complete list of distillation hyperparameters is given in Table 3.

E.3 Stage 3: Behavior prior distillation

Task. Behavior prior distillation reuses the imitation environment from Stage 1 (App. E.1) for rollout collection, with domain randomization disabled. The Stage 2 latent skill controller—encoder $q_\phi(\mathbf{z}_t|\mathbf{x}_t, \mathbf{g}_t)$, decoder $p_\theta(\mathbf{a}_t|\mathbf{x}_t, \mathbf{z}_t)$, and motion prior $p_\psi(\mathbf{z}_t|\mathbf{x}_t)$—is loaded frozen and used to drive simulation: at each step we feed the encoder's mean $\boldsymbol{\mu}_t^e$ as the latent into the frozen decoder and use the resulting deterministic action mean to advance the environment. The trainable behavior prior $p_\varphi(\mathbf{z}_t|\mathbf{x}_t)$ is optimized only on the proprioceptive states visited along these trajectories; loss evaluation queries the full encoder distribution $(\boldsymbol{\mu}_t^e, \boldsymbol{\sigma}_t^e)$ at each visited state, not a sample.

Loss. With the encoder q_ϕ held fixed, the behavior prior is trained by minimizing the expected KL divergence

$$\mathcal{L}_{\text{behavior}} = D_{\text{KL}}\big(q_\phi(\mathbf{z}_t|\mathbf{x}_t, \mathbf{g}_t) \,\|\, p_\varphi(\mathbf{z}_t|\mathbf{x}_t)\big),$$

which by Proposition 2 is equivalent to matching the marginal posterior $m_\phi(\mathbf{z}_t|\mathbf{x}_t) = \mathbb{E}_{p(\mathbf{g}_t|\mathbf{x}_t)}[q_\phi(\mathbf{z}_t|\mathbf{g}_t, \mathbf{x}_t)]$. Crucially, the encoder is not modified at this stage; it provides a fixed target distribution, so the behavior prior absorbs the goal-conditional variation in q_ϕ rather than collapsing onto local motion continuation as the motion prior does (Section 3.3).

Network architecture. The behavior prior is an MLP with SiLU activations and LayerNorm between hidden layers, mirroring the motion prior. It maps $\mathbf{x}_t$ through hidden sizes $[2048, 2048, 1024, 512]$ to the mean and log-variance of a 16-dimensional Gaussian over $\mathbf{z}_t$, matching the latent dimensionality of the frozen encoder. The behavior prior is the only trainable network in this stage; encoder, decoder, and motion prior all remain frozen. The behavior prior is initialized using LeCun initialization with zero bias.

Training. We collect $4{,}096$ parallel rollouts in MuJoCo MJX with the same 20-step unroll length used in earlier stages, train for 2×10^9 environment steps with batch size $1{,}024$ split across 16 minibatches and 4 epochs per batch, and update φ with Adam at learning rate 10^{-4} and global gradient-norm clipping at 10.0. Observation statistics are inherited from the Stage 2 normalizer and held fixed,

so that p_φ sees the same input distribution as the frozen encoder. Evaluation rollouts are run every 2×10^7 environment steps using the deterministic decoder mean and a deterministic behavior-prior mean (with stochastic-sampling evaluations reported separately in the experiments). The complete list of behavior-prior hyperparameters is given in Table 4.

E.4 Stage 4 (transfer learning): Arena foraging

Task. Stage 4 fine-tunes a task-specific policy on top of the frozen Stage 3 controller. The decoder p_θ, motion prior p_ψ, and behavior prior p_φ are loaded frozen from the Stage 3 checkpoint (Table 4); only the task policy $\pi_T(\boldsymbol{\delta}_t, \beta_t | \mathbf{x}_t, \mathbf{o}_t)$ and its critic are trained. In the arena foraging task, the rat is placed at the center of a flat circular arena (Fig. 11) and must navigate to and collect $N = 10$ treats (blue spheres of radius 0.06 m at height 0.03 m). At the start of each episode the rat's initial yaw is sampled uniformly in $[0, 2\pi)$, and the N treats are placed one per angular sector with sub-sector jitter, at uniformly random radii within an annulus of inner radius of variable size depending on the trial and outer radius $r_{\max} = 2.0$ m. Each treat emits an isotropic Gaussian scent field with bandwidth $\sigma = 0.2$ m. In addition to the proprioception vector $\mathbf{x}_t$ from Section 3.1, the agent observes a task-specific $\mathbf{o}_t$ comprising (i) the scalar total scent at the rat's skull (sum of Gaussians from currently active treats) and (ii) the egocentric gradient $\nabla s_t \in \mathbb{R}^3$ of the same scent field, which points toward the scent-weighted treat center.

Reward. The task uses a sparse reward: collecting a treat (skull entering its collision sphere) yields a large, one-time reward and the treat with its scent contribution is removed for the remainder of the episode.

Termination. Each episode runs for up to $2{,}000$ control steps (20 s at the 100 Hz control rate inherited from Stage 1), or terminates early once all N treats have been collected.

Policy gating. The task policy outputs a 16-dimensional residual $\boldsymbol{\delta}_t$ to the motion prior's mean and a scalar gating logit β_t. The binary gate b_t is thresholded with bias 0.5 and hysteresis margin 0.2: switching b_t to 1 requires $\beta_t > 0.7$, and switching to 0 requires $\beta_t < 0.3$. When $b_t = 1$, the latent is $\mathbf{z}_t = \boldsymbol{\mu}_t^m + \boldsymbol{\delta}_t$ (motion-prior residual control). When $b_t = 0$, the latent is sampled stochastically from the behavior prior, $\mathbf{z}_t = \boldsymbol{\mu}_t^b + \epsilon\boldsymbol{\sigma}$ with $\epsilon \sim \mathcal{N}(0, 1)$ and a fixed log-variance of -2.25 (i.e., $\sigma = 0.32$), which yields random motions that fall within the range of the natural behavioral diversity of the training data (Fig. 2b). The latent is passed through the frozen decoder p_θ to produce the action torques.

Network architecture. The task policy is an MLP with hidden sizes $[2048, 1024, 512]$ that maps the concatenation of $\mathbf{x}_t$ and $\mathbf{o}_t$ (normalized by running mean and variance) to the mean and log-standard-deviation of a tanh-squashed diagonal Gaussian over the joint 17-dimensional output $(\boldsymbol{\delta}_t, \beta_t)$, so that $\boldsymbol{\delta}_t \in [-1, 1]^{16}$ and $\beta_t \in [-1, 1]$. The critic is a separate $[2048, 1024, 1024, 512]$ MLP estimating the state-value function. Both networks use SiLU activations with LayerNorm between hidden layers and are randomly initialized using LeCun initialization with zero bias. The decoder, motion prior, and behavior prior all remain frozen throughout this stage.

Training. We train π_T with PPO [32] for 2×10^9 environment steps across $1{,}024$ parallel environments. Each PPO update uses a 20-step unroll, batch size $1{,}024$, and 4 epochs of 16 minibatches per batch, with Adam at learning rate 10^{-4} and gradient clipping at maximum norm 50.0. We use discount $\gamma = 0.99$, GAE $\lambda = 0.95$, PPO clip range $\varepsilon = 0.3$, entropy coefficient 10^{-2}, value-loss coefficient 0.5, and reward scaling 1.0. Evaluation rollouts are performed every 1.5×10^7 environment steps. The complete list of hyperparameters is given in Table 5.

E.5 Locomotion

Task. The locomotion task assesses how effectively the motion prior p_ψ supports goal-directed control via residual fine-tuning, without recourse to the behavior prior. The motion prior p_ψ and decoder p_θ are loaded frozen from the Stage 2 checkpoint (Table 3); only the task policy and critic are trained. We do not use the behavior prior in this task: we use $b_t = 1$ at every step, so that the latent is always $\mathbf{z}_t = \boldsymbol{\mu}_t^m + \boldsymbol{\delta}_t$, isolating the quality of residual control of the motion prior. At the start of each episode the rat is placed at the world origin in a neutral standing pose facing the world $+x$ direction (identity body quaternion, $z = 0$), on a flat arena. The task is to maintain a fixed target forward speed $v^\star$ along the world $+x$ axis. In addition to the proprioception vector $\mathbf{x}_t$ from Section 3.1, the

agent receives a task observation $\mathbf{o}_t$ comprising the previous action, and the rat's kinematic and touch sensors. The target speed $v^\star$ itself is a fixed task parameter and is not part of the observation.

Reward. The reward at each step is a piecewise-linear forward-velocity tracking signal,

$$r_t = \max\Big(0,\ 1 - |v_t^x - v^\star|/v^\star\Big),$$

where v_t^x is the world-frame $+x$ component of the torso's linear velocity (subtree center-of-mass velocity of the torso body). The reward equals 1 when $v_t^x = v^\star$ and decays linearly to 0 at $v_t^x = 0$ and at $v_t^x = 2v^\star$.

Termination. An episode terminates early if (i) the rat has *fallen*—defined as torso z below $0.03\,\mathrm{m}$ or torso tilt from vertical exceeding $60°$—or (ii) any state value becomes `NaN`. Episodes are otherwise truncated at a fixed length of $1{,}000$ control steps ($10\,\mathrm{s}$ at the $100\,\mathrm{Hz}$ control rate inherited from Stage 1).

Network architecture. The task policy is an MLP with hidden sizes $[2048, 1024, 512]$ that maps the concatenation of $\mathbf{x}_t$ and $\mathbf{o}_t$ (normalized by running mean and variance) to the mean and log-standard-deviation of a tanh-squashed diagonal Gaussian over the residual, so that $\boldsymbol{\delta}_t \in [-1, 1]^{16}$. The critic is a separate $[2048, 1024, 1024, 512]$ MLP estimating the state-value function. Both networks use SiLU activations with LayerNorm between hidden layers and are randomly initialized using LeCun initialization with zero bias. The motion prior p_ψ and decoder p_θ remain frozen throughout this task; the encoder q_ϕ is not used.

Training. We train the task policy with PPO [32] for 4×10^8 environment steps across $1{,}024$ parallel environments. Each PPO update uses a 20-step unroll, batch size $1{,}024$, and 4 epochs of 16 minibatches per batch, with Adam at learning rate 10^{-4} and gradient clipping at maximum norm 50.0. We use discount $\gamma = 0.99$, GAE $\lambda = 0.95$, PPO clip range $\varepsilon = 0.3$, entropy coefficient 10^{-2}, value-loss coefficient 0.5, and reward scaling 1.0. Evaluation rollouts are performed every 1.5×10^7 environment steps. The complete list of hyperparameters is given in Table 6.

E.6 Rearing

Task. The walk-then-rear task tests whether residual control of the motion prior can drive a navigate-and-rear sequence terminated by a sparse success bonus. As in the locomotion task, the motion prior p_ψ and decoder p_θ are loaded frozen from the Stage 2 checkpoint (Table 3); only the task policy and critic are trained, and we use $b_t = 1$ at every step so that the latent is always $\mathbf{z}_t = \boldsymbol{\mu}_t^m + \boldsymbol{\delta}_t$. At the start of each episode the rat is placed at the world origin in a neutral standing pose facing the world $+x$ direction (identity body quaternion, $z = 0$), on a flat arena; a translucent target box is rendered at the goal location for visualization. The agent must walk to the horizontal target $(x^\star, y^\star) = (1.0,\ 0.0)\,\mathrm{m}$ and rear up so that its skull reaches a target height $z^\star = 0.2\,\mathrm{m}$. A target pose is achieved when the skull is simultaneously within $0.01\,\mathrm{m}$ of $(x^\star, y^\star)$ horizontally and within $0.02\,\mathrm{m}$ of $z^\star$ vertically; success requires holding the target pose continuously for $1.0\,\mathrm{s}$ (100 control steps at $100\,\mathrm{Hz}$). In addition to the proprioception vector $\mathbf{x}_t$ from Section 3.1, the agent receives a task observation $\mathbf{o}_t$ comprising the previous action, the rat's kinematic and touch sensors, the egocentric position of the world origin in the torso frame, and the egocentric position of the target in the torso frame.

Reward. The reward at each step combines a dense piecewise-linear shaping term and a sparse terminal bonus. Letting $d_t = \|\mathbf{p}_t^{\text{skull}} - \mathbf{p}^\star\|_2$ denote the Euclidean distance from the skull to the target, the dense term is

$$r_t^{\text{dist}} = \max\Big(0,\ 1 - d_t/m\Big),$$

with shaping margin $m = 1.0\,\mathrm{m}$, so it equals 1 when the skull is at the target and decays linearly to 0 at $d_t = m$. The sparse bonus $r_t^{\text{succ}} = 500$ fires on the single step at which the running hold counter first reaches 100 steps, and is 0 otherwise.

Termination. An episode terminates if (i) the hold counter reaches 100 steps (success), (ii) the rat has *fallen*—defined as torso z below $0.02\,\mathrm{m}$ or torso tilt from vertical exceeding $80°$—or (iii) any state value becomes `NaN`. Episodes are otherwise truncated at a fixed length of 500 control steps ($5\,\mathrm{s}$ at the $100\,\mathrm{Hz}$ control rate inherited from Stage 1).

Network architecture. The task policy is an MLP with hidden sizes $[2048, 1024, 512]$ that maps the concatenation of $\mathbf{x}_t$ and $\mathbf{o}_t$ (normalized by running mean and variance) to the mean and log-standard-deviation of a tanh-squashed diagonal Gaussian over the residual, so that $\boldsymbol{\delta}_t \in [-1, 1]^{16}$. The critic

is a separate $[2048, 1024, 1024, 512]$ MLP estimating the state-value function. Both networks use SiLU activations with LayerNorm between hidden layers and are randomly initialized using LeCun initialization with zero bias. The motion prior p_ψ and decoder p_θ remain frozen throughout this task; the encoder q_ϕ is not used.

Training. We train the task policy with PPO [32] for 6×10^8 environment steps across $1{,}024$ parallel environments. Each PPO update uses a 20-step unroll, batch size $1{,}024$, and 4 epochs of 16 minibatches per batch, with Adam at learning rate 5×10^{-5} and gradient clipping at maximum norm 50.0. We use discount $\gamma = 0.99$, GAE $\lambda = 0.95$, PPO clip range $\varepsilon = 0.3$, entropy coefficient 10^{-2}, value-loss coefficient 0.5, and reward scaling 1.0. Evaluation rollouts are performed every 1.5×10^7 environment steps. The complete list of hyperparameters is given in Table 7.

E.7 Hyperparameters

Our hyperparameter choices draw on prior work where applicable, with task-specific tuning otherwise. PPO hyperparameters follow Brax's defaults [47] with minor adjustments to maximize task reward. Physics simulation and virtual rodent parameters are taken from MIMIC-MJX [6], and online distillation hyperparameters follow PULSE [13] with small modifications identified via constrained grid search to better suit the virtual rodent setting. All remaining hyperparameters were selected through grid search.

Table 2: Hyperparameters used for Stage 1 (train expert imitator) of the SCAMPER training pipeline.

Group	Hyperparameter	Value
Environment	Solver	Newton
	Solver iterations	5
	Line-search iterations	5
	Max contacts (n_{aconmax})	256
	Max constraint rows (n_{jmax})	128
	Simulation timestep	2×10^{-3} s (500 Hz)
	Control timestep	1×10^{-2} s (100 Hz)
	Mocap rate	50 Hz
	Clip length	250 frames
	Reference horizon	5 frames
	Episode start-frame range	$[0, 50]$
	Initial joint velocity	zeros
	Train / eval clip split	0.8 / 0.2
Domain randomization	Floor friction	$\mathcal{U}(0.4, 1.0)$
	Static friction scale	$\mathcal{U}(0.9, 1.1)$
	Armature scale	$\mathcal{U}(1.0, 1.05)$
	Center-of-mass jitter (m)	$\mathcal{U}(-0.05, 0.05)$
	Link mass scale	$\mathcal{U}(0.9, 1.1)$
	Torso mass jitter	$\mathcal{U}(-0.02, 0.02)$
	Initial q_{pos} jitter	$\mathcal{U}(-0.05, 0.05)$
Reward weights (w, scale)	Root position	w=1.0, σ=0.035
	Root orientation (quaternion)	w=1.0, σ=20.0
	Joint angles	w=1.0, σ=1.41
	Joint velocities	w=0.0, σ=1.0
	Body positions	w=0.0, σ=0.25
	End-effectors	w=1.0, σ=0.03
	Healthy torso z-range (m)	[0.0325, 0.5], w=1.0
	Control cost	w=0.02
	Control-diff cost	w=0.02
	Energy cost (clipped at 50)	w=0.01
Termination	Root-position deviation	> 0.1 m
	Root-orientation deviation	$> 60^\circ$
	Pose error (L2)	> 4.5
Network architecture	Policy architecture	MLP (no latent bottleneck)
	Policy hidden sizes	$[2048, 1024, 1024, 512]$
	Critic hidden sizes	$[2048, 1024, 1024, 1024, 1024, 512]$
	Observation normalization	running mean/std
PPO training	Parallel environments	4,096
	Total environment steps	3×10^9
	Batch size	1,024
	Minibatches per epoch	16
	Updates per batch	4
	Learning rate (Adam)	1×10^{-4}
	Gradient clip threshold	20.0
	PPO clip range ε	0.2
	Discount factor γ	0.95
	GAE / unroll length	20
	Action repeat	1
	Entropy coefficient	1×10^{-2}
	Value-loss coefficient	0.25
	Reward scaling	1
	Evaluation / reset interval	2.5×10^7 steps

Table 3: Hyperparameters used for Stage 2 (online distillation) of the SCAMPER training pipeline. Environment, termination, and domain-randomization settings are inherited from Stage 1 (Table 2).

Group	Hyperparameter	Value
Distillation loss	AR(1) smoothness weight α	5×10^{-3}
	Gradient clip threshold (global norm)	10.0
	Encoder / prior logvar clamp	$[-5, 0]$
KL schedule	Schedule type	linear ramp, then hold
	Start weight λ_0	1×10^{-3}
	End weight λ_1	1×10^{-4}
	Ramp range (fraction of training)	$[0\%,\ 25\%]$
Hard-negative mining	Reweight strength α_{HN}	0.1
	Max sample weight w_{max}	5.0
Network architecture	Latent size	16
	Encoder hidden sizes	$[2048, 1024, 1024]$
	Decoder hidden sizes	$[1024, 1024, 512]$
	Motion prior hidden sizes	$[1536, 1024, 512]$
Training	Parallel environments	4,096
	Total environment steps	1×10^9
	Batch size	1,024
	Minibatches per epoch	16
	Updates per batch	4
	Learning rate (Adam)	1×10^{-4}
	Unroll length	20
	Action repeat	1
	Train / eval clip split	0.8/0.2
	Evaluation / reset interval	2×10^7 steps

Table 4: Hyperparameters used for Stage 3 (behavior prior distillation) of the SCAMPER training pipeline. Environment settings are inherited from Stage 1 (Table 2); the encoder, decoder, and motion prior are loaded frozen from Stage 2 (Table 3). Only the behavior prior is updated.

Group	Hyperparameter	Value
Distillation loss	KL weight	1.0
	Gradient clip threshold (global norm)	10.0
Network architecture	Latent size (inherited from Stage 2)	16
	Behavior prior hidden sizes	$[2048, 2048, 1024, 512]$
Training	Parallel environments	4,096
	Total environment steps	2×10^9
	Batch size	1,024
	Minibatches per epoch	16
	Updates per batch	4
	Learning rate (Adam)	1×10^{-4}
	Unroll length	20
	Action repeat	1
	Train / eval clip split	0.8/0.2
	Evaluation / reset interval	2×10^7 steps

Table 5: Hyperparameters used to train SCAMPER on the arena foraging task (Stage 4: transfer learning). The decoder, motion prior, and behavior prior are loaded frozen from the Stage 3 checkpoint (Table 4); only the task policy and critic are trained at this stage.

Group	**Hyperparameter**	**Value**
Task / environment	Episode length	2,000 control steps
	Evaluation interval	1.5×10^7 env steps
	Number of treats per episode	10
	Treat sphere radius	0.06 m
	Treat height	0.03 m
	Minimum spawn radius	Variable
	Maximum spawn radius	2.0 m
	Scent kernel σ	0.2 m
	Treat-collection reward	500.0
Behavior–motion prior gating	Gate threshold b_{bias}	0.5
	Gate hysteresis margin	0.2
	Behavior-prior fixed log-variance	-2.25
Network architecture	Task policy hidden sizes	[2048, 1024, 512]
	Critic hidden sizes	[2048, 1024, 1024, 512]
PPO training	Parallel environments	1,024
	Total environment steps	2×10^9
	Batch size	1,024
	Minibatches per epoch	16
	Updates per batch	4
	Learning rate (Adam)	1×10^{-4}
	Gradient clip threshold (max norm)	50.0
	PPO clip range ε	0.3
	Discount factor γ	0.99
	GAE λ	0.95
	Entropy coefficient	1×10^{-2}
	Value-loss coefficient	0.5
	Reward scaling	1.0
	Unroll length	20
	Action repeat	1

Table 6: Hyperparameters used to train SCAMPER on the locomotion task. The motion prior p_ψ and decoder p_θ are loaded frozen from the Stage 2 checkpoint (Table 3); only the task policy and critic are trained at this stage. The policy outputs a residual $\boldsymbol{\delta}_t$ added to the (deterministic) motion-prior mean.

Group	Hyperparameter	Value
Task / environment	Episode length	1,000 control steps
	Evaluation interval	1.5×10^7 env steps
	Target forward speed	Variable
	Forward-velocity reward weight	1.0
Network architecture	Task policy hidden sizes	[2048, 1024, 512]
	Critic hidden sizes	[2048, 1024, 1024, 512]
PPO training	Parallel environments	1,024
	Total environment steps	4×10^8
	Batch size	1,024
	Minibatches per epoch	16
	Updates per batch	4
	Learning rate (Adam)	1×10^{-4}
	Gradient clip threshold (max norm)	50.0
	PPO clip range ε	0.3
	Discount factor γ	0.99
	GAE λ	0.95
	Entropy coefficient	1×10^{-2}
	Value-loss coefficient	0.5
	Reward scaling	1.0
	Unroll length	20
	Action repeat	1

Table 7: Hyperparameters used to train SCAMPER on the walk-rear task. The motion prior p_ψ and decoder p_θ are loaded frozen from the Stage 2 checkpoint (Table 3); only the task policy and critic are trained. The policy outputs a residual $\boldsymbol{\delta}_t$ added to the (deterministic) motion-prior mean.

Group	Hyperparameter	Value
Task / environment	Episode length	500 control steps
	Evaluation interval	1.5×10^7 env steps
	Target position $(x^\star, y^\star, z^\star)$	$(1.0,\ 0.0,\ 0.2)$ m
	xy-tolerance for success	0.01 m
	z-tolerance for success	0.02 m
	Required hold duration	1.0 s
	Distance-shaping margin	1.0 m
	Distance-to-target reward weight	1.0
	Walk-rear success reward	500.0
Network architecture	Task policy hidden sizes	[2048, 1024, 512]
	Critic hidden sizes	[2048, 1024, 1024, 512]
PPO training	Parallel environments	1,024
	Total environment steps	6×10^8
	Batch size	1,024
	Minibatches per epoch	16
	Updates per batch	4
	Learning rate (Adam)	5×10^{-5}
	Gradient clip threshold (max norm)	50.0
	PPO clip range ε	0.3
	Discount factor γ	0.99
	GAE λ	0.95
	Entropy coefficient	1×10^{-2}
	Value-loss coefficient	0.5
	Reward scaling	1.0
	Unroll length	20
	Action repeat	1

F Analysis of randomly generated behavior

F.1 Rollout collection protocol

All quantitative comparisons in this section (KID, Vendi score, root-height ECDF, spatial occupancy) are computed on rollouts collected with a single shared protocol so that the only difference between conditions is the controller that produced the trajectory.

For every condition, we start from the same set of initial states: the first frame of each of the 842 reference clips, with the rat placed in the corresponding generalized pose and zero joint velocity. This matches the initialization used during imitation training and keeps the starting distribution of body configurations identical across conditions. Each rollout is then advanced in MuJoCo MJX under the chosen controller for the same horizon as the reference clip ($\sim$5 s of simulated time, with control at 100 Hz on top of a 500 Hz physics step); the trajectory is driven purely by stochastic sampling of the controllers. At each step we log the rat's generalized coordinates $\mathbf{q} \in \mathbb{R}^{74}$ and the latent vector passed to the decoder. Clips are processed in parallel batches on a single GPU, and one randomization key is split per condition so that stochastic samples are reproducible and independent across conditions. We perform the rollouts using both the behavior prior and the motion prior and compare to the imitation rollouts as the ground truth. We test across both the deterministic sampling condition (no noise; $\sigma = 0$) and an array of different sampling noise levels logvar $= \{-4, -3, -2, -1, -0.5, 0\}$ ($\sigma \approx \{0.14, 0.22, 0.37, 0.61, 0.78, 1.00\}$).

F.2 Vendi score

What the Vendi score measures. The Vendi score [34] quantifies the effective number of distinct modes in a sample under a user-specified similarity kernel. It is defined as the exponential of the von Neumann (Shannon) entropy of the eigenvalues of the normalized similarity matrix and, in contrast to a clustering-based count, requires no reference distribution, no labels, and no choice of cluster number. It reduces to the number of items when they are mutually orthogonal and to 1 when they are all identical. Higher Vendi score indicates greater behavioral diversity.

Egocentric kinematic features. Unlike for KID, where the feature extractor is a learned VAE, we measure diversity directly in an egocentric kinematic feature space so that the score is interpretable in terms of physically meaningful movement variation rather than the geometry of a learned latent space. From each rollout's generalized coordinates $\mathbf{q} \in \mathbb{R}^{74}$, we compute, per frame, the root height, root roll and pitch, body-frame linear velocity, yaw rate, and the joint angles — 74 dims/frame. The body-frame velocities are intrinsically heading-invariant and the planar root translation is dropped, so the representation is invariant to the absolute location of the rat in the arena. Frames are then stacked into sliding windows of length 30 (0.6 s at the 50 Hz control rate) with a stride of 15, yielding $2{,}220$-dimensional feature vectors per window.

Joint PCA(50) embedding. To make the Vendi computation tractable and to apply a consistent similarity geometry across conditions, we fit a single PCA basis of dimension 50. The basis is fit jointly on a stratified subsample of $50{,}000$ windows, drawn proportionally across every condition (each prior at every noise level, plus the training-data baseline) so that no condition dominates the principal directions. All windows are then projected into this shared 50-dimensional embedding before the Vendi score is computed.

Computing the Vendi score. Let $X \in \mathbb{R}^{n \times 50}$ be the PCA-embedded windows of a single rollout sample, and let X_n denote X with rows ℓ_2-normalized. We use the cosine similarity kernel and exploit the dual identity $\mathrm{eig}(X_n X_n^\top / n) = \mathrm{eig}(X_n^\top X_n / n)$ (up to zero eigenvalues) to compute Vendi via the small 50×50 Gram matrix,

$$M = \frac{1}{n} X_n^\top X_n, \qquad \{\lambda_i\}_{i=1}^{50} = \mathrm{eigvalsh}(M),$$

which is exact for the cosine Vendi score and avoids forming the $n \times n$ kernel. The Vendi score is then the exponential of the Shannon entropy of the (positive) eigenvalues,

$$\mathrm{VS}(X) = \exp\left(- \sum_{i:\, \lambda_i > 0} \lambda_i \log \lambda_i \right),$$

with the convention $0 \log 0 = 0$.

Bootstrap over clips. The independent unit of replication is the imitation *clip* (windows within a clip share a starting condition and are not independent), so we bootstrap at the clip level. There are 842 clips per condition (motion-prior, behavior-prior, training-data). For each condition we bootstrap by resampling the clip indices with replacement and recompute the Vendi score. We perform $B = 500$ bootstrap iterations. The same clip indices are reused across all conditions in a given iteration, so cross-policy comparisons are exactly paired. We report bootstrap means with 95% confidence intervals.

F.3 Kernel Inception Distance

What KID measures. The Kernel Inception Distance (KID) quantifies the dissimilarity between two distributions in a learned feature space as the squared Maximum Mean Discrepancy (MMD^2) under a polynomial kernel [37]. In contrast to a metric like Fréchet Inception Distance (FID) [49], it is an unbiased estimator and remains well-defined for moderately sized samples. KID is zero when the two distributions match.

Training the feature extractor. There is no pre-trained "Inception" network for rodent kinematics, so we train a small variational autoencoder (VAE) on the real (training-data) joint-angle trajectories and use its encoder mean $\mu_\phi(\mathbf{x}) \in \mathbb{R}^d$ as the feature extractor. The VAE is optimized by the standard β-VAE objective

$$\mathcal{L}_{\mathrm{VAE}}(\phi, \theta) = \mathbb{E}_{\mathbf{x}} \left[\|\hat{\mathbf{x}}_\theta(\mathbf{x}) - \mathbf{x}\|_2^2 + \beta \, D_{\mathrm{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, \mathcal{N}(\mathbf{0}, \mathbf{I})) \right],$$

with hyperparameters listed in Table 8. After training, the decoder is discarded and only μ_ϕ is retained; we repeat training under several random seeds and report the mean and standard deviation of KID across them.

Trajectory inputs and preprocessing. Each training sample is a single 5 s trajectory of the rat's generalized coordinates $\mathbf{q} \in \mathbb{R}^{74}$ — planar root position (x, y), root height z, root quaternion (4 dims), and 67 joint angles — drawn from the training-data imitation rollouts. We use imitation rollouts as the ground truth as opposed to the original mocap because the alignment process from keypoints to joint angle trajectories (through STAC-MJX [6]) produces jagged transition frame-to-frame. Imitation of this reference allows the simulated physics to smooth out the trajectories and results in a fairer comparison to the generated motion. Before encoding, three operations are applied so that the comparison between training and generated distributions is invariant to nuisance factors and on a consistent scale: (i) the root quaternion is replaced by a yaw-removed 6D rotation representation [50]; (ii) the planar root position (x, y) is discarded; (iii) joint dimensions are z-score normalized using statistics computed on the real data. Each per-step vector therefore consists of root height, root rotation, and 67 normalized joint angles for 74 features per step; the full trajectory is flattened to a single vector before being fed to the encoder. The same preprocessing is applied to every generated rollout.

Computing KID. Let $X = \{\mu_i^r\}_{i=1}^m$ and $Y = \{\mu_j^f\}_{j=1}^m$ be the encoder features of the real and generated samples. We use the cubic polynomial kernel of Bińkowski et al. [37],

$$k(\mathbf{u}, \mathbf{v}) = \left(\frac{1}{d} \mathbf{u}^\top \mathbf{v} + 1 \right)^3,$$

and define $\widehat{\mathrm{MMD}}_u^2$ as the unbiased squared-MMD estimator of Gretton et al. [51],

$$\widehat{\mathrm{MMD}}_u^2(X, Y) = \frac{1}{m(m-1)} \sum_{i \neq j} k(\mu_i^r, \mu_j^r) + \frac{1}{m(m-1)} \sum_{i \neq j} k(\mu_i^f, \mu_j^f) - \frac{2}{m^2} \sum_{i,j} k(\mu_i^r, \mu_j^f).$$

To stabilize the per-seed estimate, we average $\widehat{\mathrm{MMD}}_u^2$ over S random subsets of size m drawn independently from X and Y:

$$\mathrm{KID}(X, Y) = \frac{1}{S} \sum_{s=1}^{S} \widehat{\mathrm{MMD}}_u^2 \left(X^{(s)}, Y^{(s)} \right).$$

Table 8: Hyperparameters for the VAE feature extractor ("inception network") used to compute KID.

Group	Hyperparameter	Value
Architecture	Encoder hidden sizes	$[512, 256]$
	Decoder hidden sizes	$[256, 512]$
	Latent dimensionality d	64
	Activation	SiLU
	Layer normalization	yes
Regularization	Dropout rate	0.1
	Weight decay (AdamW)	1×10^{-5}
β-VAE	Target β	1.0
	β warmup (linear)	2,500 epochs
Training	Optimizer	AdamW
	Learning rate	1×10^{-3}
	Gradient clip norm	1.0
	Batch size	128
	Total epochs	5,000
	Random seeds	$\{0, 1, 2, 3\}$
KID computation	Polynomial kernel degree	3
	Number of subsets S	100
	Subset size m	$\min(\lvert X\rvert, \lvert Y\rvert, 1{,}000)$

G Gait analysis

We compare the gaits produced by SCAMPER, MIMIC-MJX, and the from-scratch baseline against empirical rat gait data from Danner et al. [39], at three target speeds spanning walk, trot, and gallop. The same pipeline is applied to every model rollout so that the only difference between conditions is the controller.

Rollouts. For each (model, target speed) condition we run a fixed-length deterministic rollout of the locomotion task, with the velocity setpoint chosen to elicit the corresponding gait. The simulator state, the foot-body world positions, and the four limb touch-sensor traces (palm and sole on each side) are saved for offline analysis.

Contact barcode. For each rollout we convert the four raw touch-sensor traces into a binary stance/swing barcode of shape $[4, T]$, with rows ordered $[\mathrm{RH}, \mathrm{RF}, \mathrm{LF}, \mathrm{LH}]$. The pipeline is: (i) per-channel median filtering to suppress sensor noise; (ii) an optional spike-removal step that zeros out brief above-threshold transients on a sensor channel that is otherwise quiescent (so that fleeting contacts never get registered as stance bouts); (iii) per-channel thresholding to a binary stance signal, with the threshold tuned per limb to account for differing typical contact forces; and (iv) binary morphological closing on each channel to fill brief gaps in otherwise-continuous stance bouts. The four cleaned binary traces are then stacked into the barcode used for all subsequent steps.

Gait cycle detection. Gait cycles are extracted from the barcode using a chosen reference limb. We find every stance onset (a $0 \to 1$ transition on the reference channel), discard onsets that are closer together than a minimum cycle length (rejects double-detection of a single foot-strike), and define each gait cycle as the interval between two consecutive onsets that is at least the minimum cycle length. Detection is restricted to a per-condition analysis window so that the rat is at steady-state speed during the cycles being analyzed.

Per-cycle metrics. For every detected cycle we compute the standard set of behavioral-gait metrics used in rodent locomotion studies [40, 41]:

- **Stride length.** Euclidean distance between the reference foot's world position at the start and end of the cycle.
- **Mean velocity.** Stride length divided by cycle duration.

- **Cadence.** Stride frequency converted to steps per minute (counting both feet of a contralateral pair).
- **Stance percentage / duty factor.** Fraction of the cycle during which the reference limb is in stance.

We then aggregate these metrics across all cycles in the analysis window and report mean and standard deviation per (model, gait) condition.

Swing–stance gait diagrams. The diagrams in Figure 4c are computed by isolating, on each of the four limb channels, every complete stance bout within the analysis window (partial bouts that overlap the window edges are dropped), then computing per-limb mean stance duration and standard deviation, mean stride period (median inter-onset interval averaged over limbs), and the phase offset of each limb relative to a reference hind limb. Phase offsets are taken as the mean delay between the first stance onset of each limb after a reference-limb onset, modulo the stride period. The resulting per-limb stance bars are tiled across the cycle for visualization.

Empirical reference. The empirical rat data [39] are filtered to intact-spinal-cord animals for each of the three gait categories, and we extract per-stride duty factors for the four limbs, the inter-limb phase relationships (left–right hind, homolateral, and diagonal), and the stride duration. These are converted into the same set of per-limb mean stance, phase offset, mean stride period, and duty-factor summaries used for the model rollouts, providing a directly comparable reference distribution.

Quantitative summary. For the comparison table, every metric is summarized per (gait, model) as $\mu \pm \sigma$ over cycles. Each model is then scored against the empirical reference as the difference in means normalized by the empirical standard deviation, $|\Delta\mu|/\sigma_{\text{emp}}$, which expresses the discrepancy in units of the natural across-stride variability of real rats. The metrics included are stride duration, step frequency, per-limb stance and swing duration, per-limb duty factor, temporal symmetry (left/right hind duty-factor asymmetry), and per-limb phase offset — giving a single table that aggregates the full per-limb timing pattern alongside the cycle-level summaries.

H Supplementary analyses and results

H.1 Empirical tests of Propositions 1 and 2

To evaluate the encoder and the two priors on the states they actually encounter, we rolled out the reference dataset in closed loop through the frozen Stage-2 encoder–decoder and collected ~200,000 proprioceptive states. From these we drew 20,000 anchor states. For each anchor we identified the 32 most similar proprioceptive states in a 64-dimensional principal-component space, excluding the anchor's own clip to avoid autocorrelation, and took the goal that followed each of them as a candidate goal. Each anchor's proprioception was then held fixed while its goal was swapped for each of these $K = 32$ candidates.

Both propositions rest on the identity $\mathbb{E}\left[D_{\mathrm{KL}}(q_\phi \,\|\, p)\right] = I_\phi(Z; G \mid X) + \mathbb{E}\left[D_{\mathrm{KL}}(m_\phi \,\|\, p)\right]$, where m_ϕ is the encoder marginal. The first term depends only on the encoder, so it is identical for both priors and cancels in their difference. The comparison between priors therefore reduces to closed-form Gaussian KL divergences. The behavior prior p_φ is 8.46 nats closer to the encoder marginal than the motion prior p_ψ averaged across all individual states (Fig. 7a).

To measure how much of the encoder's behavior at a fixed body state is driven by the goal, we decompose the encoder's latent variance at each anchor with the law of total variance. The between-goal term is the variance of the encoder mean across the 32 candidate goals, $\mathrm{Var}_g[\mu_\phi]$, and the within-goal term is the expected encoder variance, $\mathbb{E}_g[\sigma^2_\phi]$. With proprioception held fixed, 49.7% of the encoder's latent variance is between-goal (Fig. 7b). The regularization in Proposition 1 therefore limits the conditional mutual information $I_\phi(Z; G \mid X)$ but does not drive it to zero. Driving it to zero would be undesirable, because the behavior prior distilled from the encoder could then no longer produce diverse motion.

Because all 32 candidates at an anchor share the same proprioceptive state, any above-chance decoding of the goal reflects information carried by the latent alone. We draw one candidate uniformly at

random, sample a latent from the encoder's distribution for that candidate, score all 32 candidates by the density their encoder Gaussians assign to the sample, and predict the highest-scoring one. Since the candidate prior is uniform and the densities are the encoder's own, this rule is Bayes-optimal and needs no trained probe. It identifies the correct goal 69.4% of the time, against 3.1% chance (a 22-fold improvement), and ranks it in the top 5 for 94.6% of anchors, against 15.6% chance (Fig. 7c).

Finally, we tested Proposition 1 directly by repeating the fixed-proprioception analysis on all 50 Stage-2 checkpoints, each with its own observation normalizer. We measure goal sensitivity as the mutual information between the latent and which of the K candidate goals was used, which is bounded above by $\log K$. The Stage-2 KL coefficient λ ramps linearly down from 10^{-3} to 10^{-4} over the first 25% of training and is then held at its floor. Goal sensitivity tracks this schedule. It falls from 1.70 to 1.33 nats while λ is near its maximum, which is the bottleneck Proposition 1 describes. It then rises monotonically to 2.52 nats as λ decays and stops changing when λ reaches its floor, varying by less than 0.03 nats over the remaining 75% of training (Fig. 7d). Nothing else in the training schedule changes at that point, so the strength of the bottleneck, rather than generic training dynamics, sets how much goal information the latent carries. Proposition 1 should therefore be read as describing a pressure that acts during training rather than a property of the final model: the schedule deliberately relaxes the bottleneck to improve reconstruction, and the converged encoder is goal-rich.

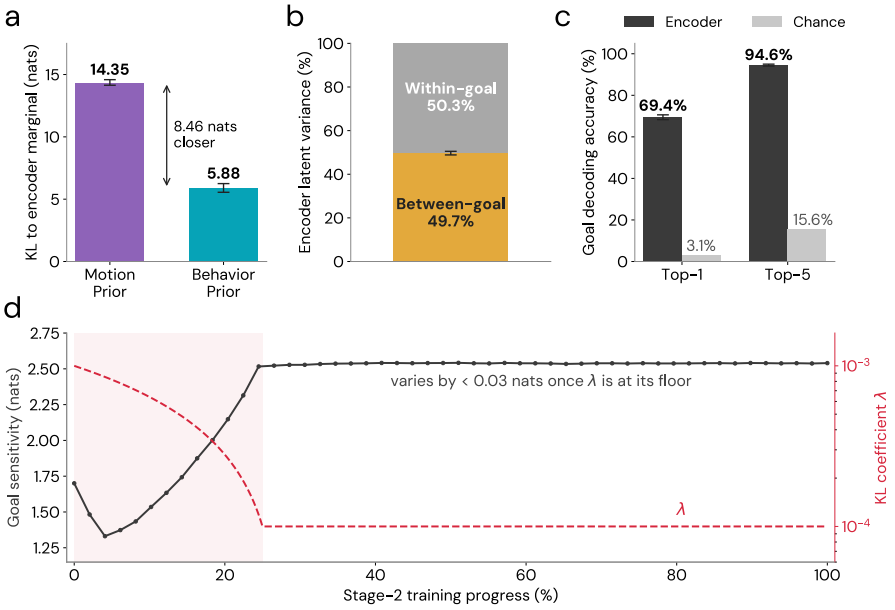


Figure 7: **(a)** KL divergence to the encoder marginal for the motion and behavior priors. **(b)** Between-goal and within-goal shares of the encoder's latent variance at fixed proprioception. **(c)** Goal decoding accuracy from the latent, compared with chance. **(d)** Goal sensitivity across Stage-2 training, with the KL coefficient λ on the right axis. Error bars are 95% clip-level bootstrap CIs.

H.2 Diversity and realism of prior sampling

In Fig. 2 we demonstrate that when sampling random motion trajectories, we obtain greater behavioral diversity from the behavior prior than from the motion prior quantified by the resulting Vendi scores. However, the observed difference in diversity could reflect discrepancies in effective perturbation magnitude of the latent rather than qualitatively different distributions. To address this, we compare the behavioral diversity by examining the impact of noise as a function of both the scales and the jitter of the sampled latents. To disambiguate whether the scale of the latents is in fact responsible for the difference in diversity, we measure the Vendi score against the mean per-dimension standard deviation of the sampled latents for both priors across all rollouts at various sampling noise levels (Fig. 8a). While we do find that the scale of the latents is different for the two priors, when accounting for this along the domain of the measured latent variance, the behavior prior is still more diverse than the motion prior. Secondly, to disambiguate whether the behavior prior's increased diversity

is a result of jitter (i.e., how quickly the latents change from timestep to timestep), we measure the Vendi score versus the one-step latent displacement (Fig. 8b). While we again find that both priors tend towards varying rates of per-step latent displacement for the same sampled noise level, when accounting for this difference, the behavior prior remains more behaviorally diverse than the motion prior over the domain of the latent jitter measured.

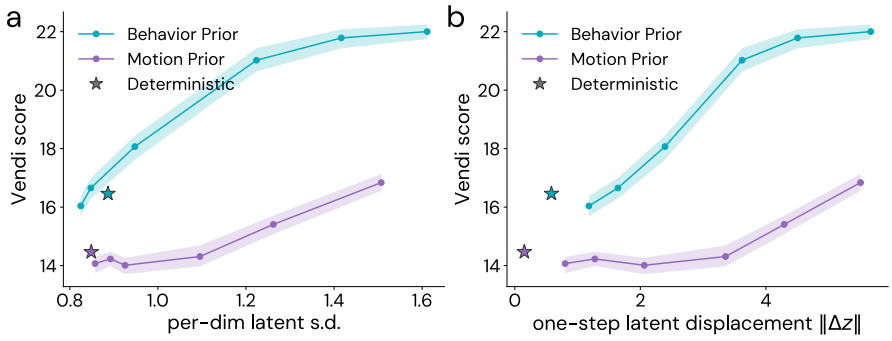


Figure 8: Diversity at matched latent scale. Vendi score of random generations vs. the realized per-dimension latent s.d. (**a**) and mean one-step latent displacement (**b**) at various levels of sampling noise (σ). Stars: deterministic ($\sigma = 0$). 95% bootstrap CIs shown as shaded regions.

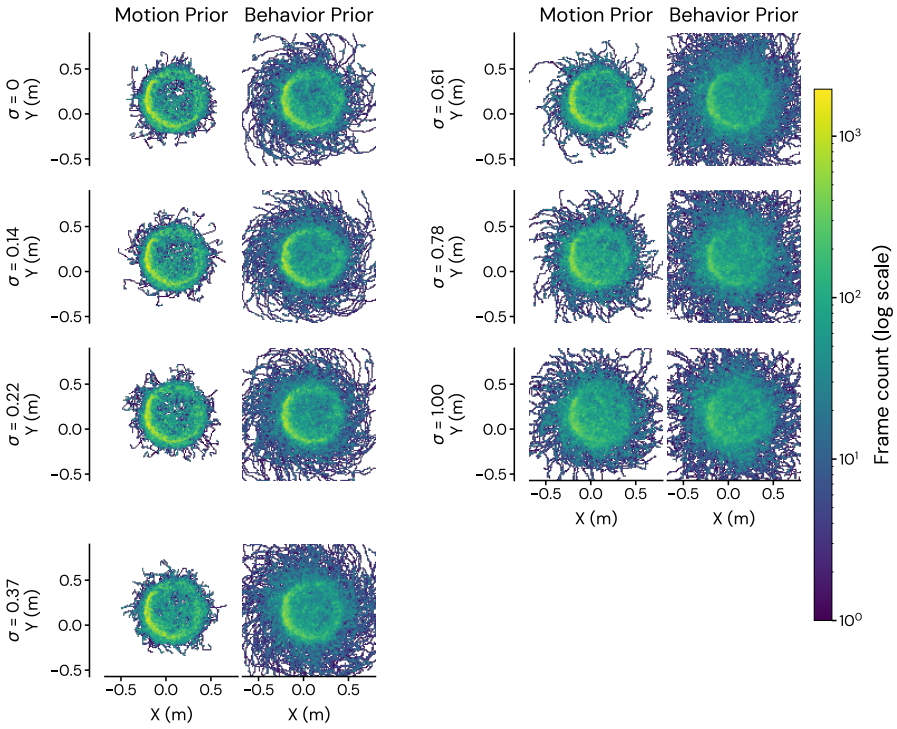


Figure 9: Spatial occupancy of all tested sampling noise levels plotted as 2D histograms of body location (log-scaled frame counts) for the motion prior and behavior prior.

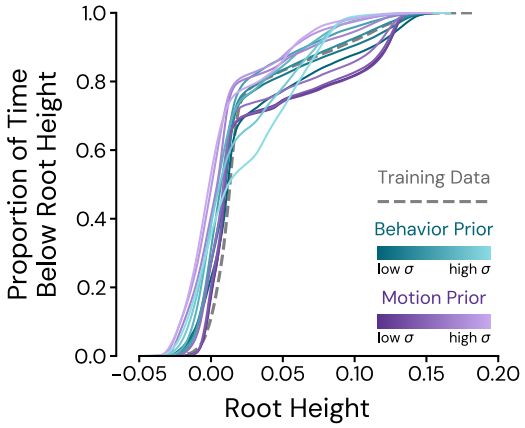


Figure 10: Empirical cumulative distribution function of root height at all tested sampling noise levels compared against the training data reference (*dashed gray*).

Table 9: Wasserstein distance between empirical root-height distributions of prior rollouts and the training data, across sampling noise levels σ. Corresponds to results from Figure 10.

	Wasserstein Distance (↓)						
Prior Networks	σ=0	σ=0.14	σ=0.22	σ=0.37	σ=0.61	σ=0.78	σ=1.00
Behavior Prior	**.0085**	**.0054**	**.0045**	**.0092**	**.0106**	**.0129**	**.0162**
Motion Prior	.0116	.0121	.0115	.0099	.0126	.0171	.0173

H.3 Foraging task

H.3.1 Arena foraging supplemental results

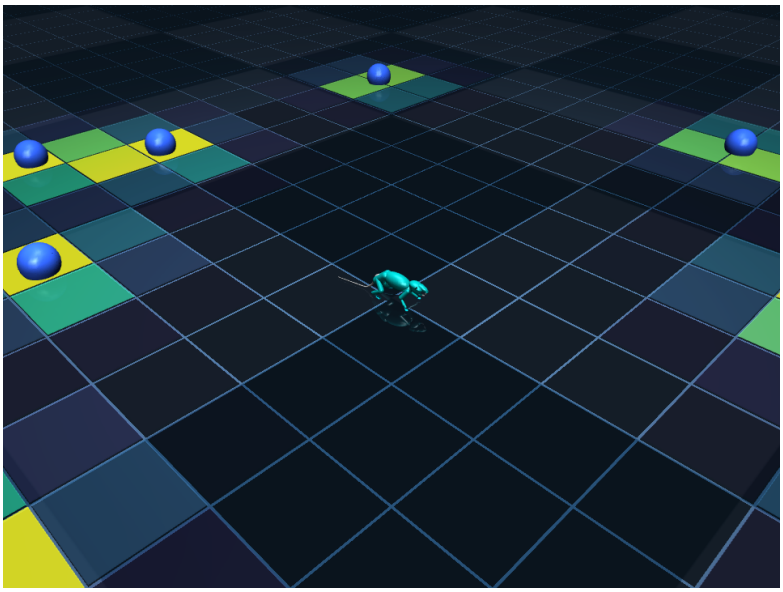


Figure 11: Rendering of the arena foraging task.

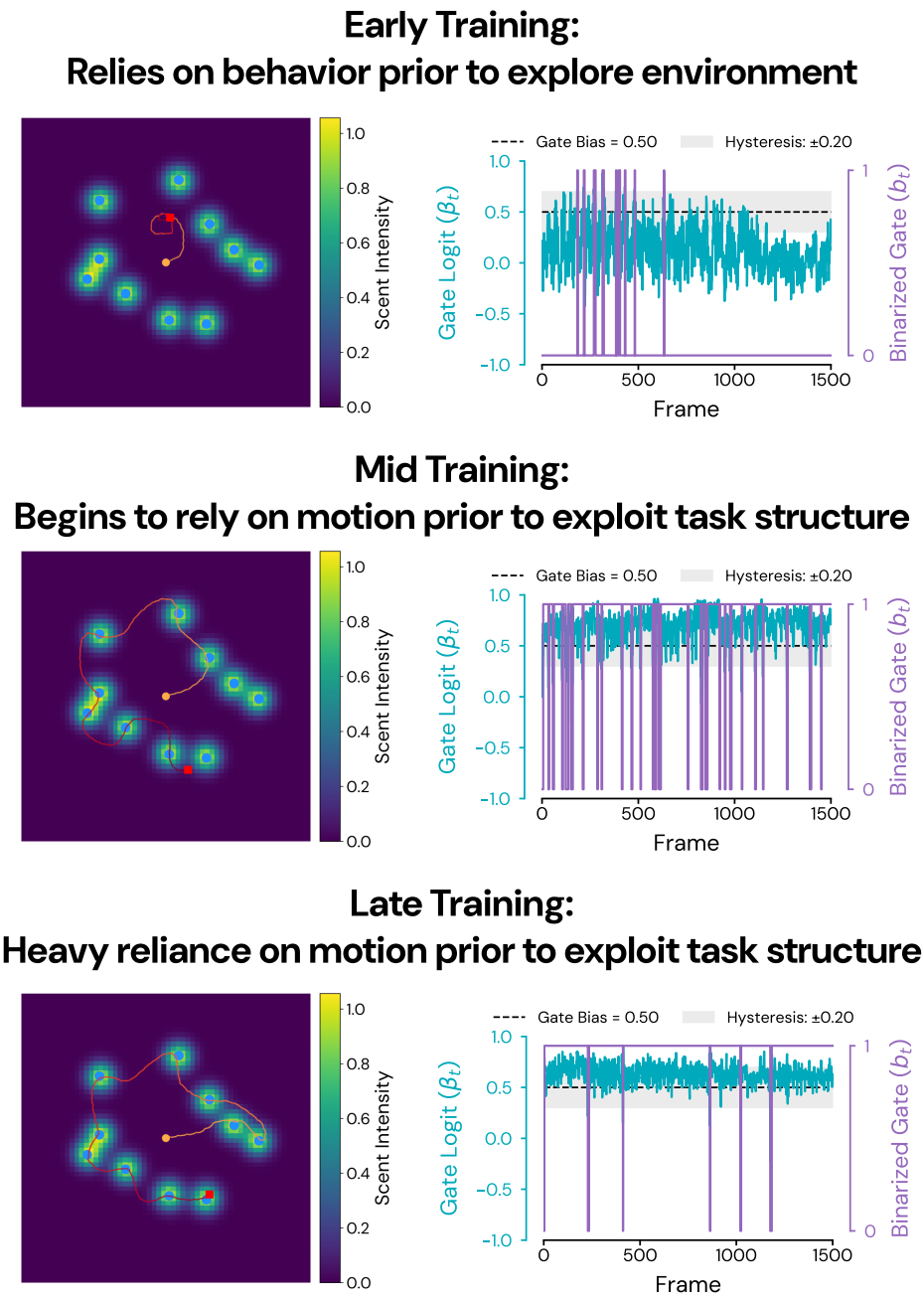


Figure 12: SCAMPER's training progression on the arena foraging task. Top-down view of trajectories the rodent takes along with the evolution of the gate logit (β_t) and its resulting binarization (b_t). Here when a treat is collected it is still displayed in the arena for visualization purposes, but in practice it disappear upon successful collection. $b_t = 0$ results in stochastic sampling of the behavior prior. $b_t = 1$ results in residual control of the motion prior. During early training the rodent relies heavily on the behavior prior to explore its environment. During mid training, the rodent begins to gate rapidly between both priors as it begins to exploit the reward signals from the collection of treats. During late training the rodent mostly uses residual control of the motion prior to effectively exploit the task structure.

H.3.2 Alternatives to the adaptive binary gate

The adaptive binary gate is one way to decide when the behavior prior or the motion prior controls the latent. To test whether the specific mechanism matters, we replaced it in the foraging task with two alternatives that need no learned gate: a fixed exploration schedule and a fixed linear interpolation between the two priors. In both variants the policy outputs only the residual, and the motion prior, behavior prior and decoder stay frozen, as with the adaptive gate.

Fixed exploration schedule. The latent is constructed exactly as with the adaptive gate, $\mathbf{z}_t = g_t\,(\boldsymbol{\mu}_t^{\mathrm{MP}} + \boldsymbol{\delta}_t) + (1 - g_t)\,\mathbf{z}_t^{\mathrm{BP}}$, where $\boldsymbol{\mu}_t^{\mathrm{MP}}$ is the motion prior mean, $\boldsymbol{\delta}_t$ is the policy's residual and $\mathbf{z}_t^{\mathrm{BP}}$ is a behavior prior sample. However, the binary gate g_t is no longer emitted by the policy. Instead, it switches stochastically, and the probability $p(g_t = 1)$ of motion-prior control follows a fixed schedule over training. It is held at 0 for an initial warmup fraction of training, ramped to 1 over a subsequent ramp fraction, and held at 1 thereafter. We vary when this switch happens.

Linear interpolation. Instead of selecting one prior at each step, the two priors are blended with a constant weight $\rho \in [0, 1]$ and the residual is added on top: $\mathbf{z}_t = \rho\,\boldsymbol{\mu}_t^{\mathrm{MP}} + (1 - \rho)\,\mathbf{z}_t^{\mathrm{BP}} + \mathbf{r}_t$. Here ρ is a fixed hyperparameter rather than a learned quantity. The endpoints $\rho = 1$ and $\rho = 0$ reduce to residual control of the motion prior alone and of the behavior prior alone, respectively.

Results. Table 10 reports the mean number of treats collected in the hardest foraging condition (minimum treat radius of $1.5\,\mathrm{m}$). Both alternatives can succeed, but only within a narrow range of settings. The fixed schedule trains successfully when control passes from the behavior prior to the motion prior early in training (warmup 0.05, ramp 0.25, so that the motion prior has full control by 30% of training). It fails when the switch happens in the middle or later phases of training, and training also fails if the switch never happens. Likewise, interpolation succeeds only when the blend is weighted towards the behavior prior ($\rho = 0.25$) and fails for $\rho = 0.5$ and $\rho = 0.75$.

These results show that what matters is the right mix of the behavior and motion priors over training. The adaptive binary gate is one means of achieving that mix, not the only one. Its advantage over the alternatives is that the user does not need to hand-tune a schedule or blend weight: the gate only needs to be biased towards the behavior prior at the start of training, and the transition to the motion prior then emerges from learning (Figure 12).

Table 10: Alternatives to the adaptive binary gate in the foraging task. Values are the mean number of treats collected in the hardest condition (minimum treat radius $1.5\,\mathrm{m}$). Schedule fractions are of total training steps. α is the weight on the motion prior.

Variant	Setting	Mean treats collected
Fixed schedule (warmup / ramp)	0.05 / 0.25	9.94 ± 0.03
	0.10 / 0.60	0.00 ± 0.00
	0.25 / 0.65	0.00 ± 0.00
Linear interpolation (ρ)	0.25	9.91 ± 0.04
	0.50	0.00 ± 0.00
	0.75	0.00 ± 0.00

H.4 Locomotion task

Here we present the full gait analysis results for the empirical rat gait data [39], the main SCAMPER model which residually controls the motion prior (MP), a control which residually controls the behavior prior (BP), MIMIC-MJX, and the from-scratch trained model. Notably, when performing residual control using the behavior prior, galloping is never discovered and the lower speed gaits also lack resemblance to the empirical data both qualitatively (Fig. 13) and quantitatively (Table 11).

Table 11: Full gait metric comparison to empirical rat data. Empirical, SCAMPER (MP), SCAMPER (BP), MIMIC-MJX, and Scratch columns show mean $\pm$ std. $|\Delta\mu|/\sigma_{\text{emp}}$ is the absolute difference in means normalized by empirical standard deviation.

		Empirical	SCAMPER (MP)	SCAMPER (BP)	MIMIC-MJX	Scratch	$\lvert\Delta\mu\rvert/\sigma_{\text{emp}}$ (↓)			
Gait	Metric						SCAMPER (MP)	SCAMPER (BP)	MIMIC-MJX	Scratch
Walk	Stride Duration (ms)	308 ± 91	332 ± 16	313 ± 34	374 ± 23	255 ± 293	0.27	**0.06**	0.73	0.58
	Step Frequency (Hz)	3.46 ± 0.75	3.02 ± 0.15	3.22 ± 0.33	2.69 ± 0.26	4.61 ± 2.62	0.58	**0.31**	1.02	1.53
	Stance LH (ms)	182 ± 82	191 ± 21	178 ± 26	221 ± 36	223 ± 295	0.11	**0.05**	0.48	0.50
	Stance RH (ms)	175 ± 66	183 ± 9	208 ± 59	247 ± 31	80 ± 0	**0.12**	0.51	1.09	1.45
	Stance LF (ms)	171 ± 71	172 ± 32	137 ± 11	153 ± 15	20 ± 10	**0.02**	0.48	0.25	2.12
	Stance RF (ms)	174 ± 69	183 ± 6	217 ± 13	187 ± 16	0 ± 0	**0.13**	0.62	0.19	2.53
	Swing LH (ms)	125 ± 20	144 ± 5	128 ± 7	152 ± 10	150 ± 45	0.96	**0.11**	1.33	1.23
	Swing RH (ms)	133 ± 31	148 ± 5	113 ± 21	123 ± 8	175 ± 0	0.51	0.65	**0.29**	1.37
	Swing LF (ms)	136 ± 25	152 ± 7	174 ± 9	219 ± 22	120 ± 0	**0.63**	1.52	3.33	0.66
	Swing RF (ms)	133 ± 30	149 ± 3	94 ± 5	184 ± 7	255 ± 0	**0.50**	1.30	1.67	3.99
	Duty Factor LH	$.576 \pm .074$	$.576 \pm .064$	$.568 \pm .082$	$.593 \pm .095$	$.877 \pm 1.157$	**0.00**	0.11	0.23	4.08
	Duty Factor RH	$.562 \pm .055$	$.550 \pm .027$	$.665 \pm .188$	$.660 \pm .083$	$.314 \pm .000$	**0.21**	1.87	1.77	4.47
	Duty Factor LF	$.544 \pm .059$	$.519 \pm .098$	$.438 \pm .036$	$.409 \pm .039$	$.079 \pm .039$	**0.44**	1.82	2.31	7.96
	Duty Factor RF	$.558 \pm .055$	$.552 \pm .020$	$.691 \pm .041$	$.502 \pm .042$	$.000 \pm .000$	**0.11**	2.42	1.03	10.14
	Temporal Symmetry	$.011 \pm .068$	$.016 \pm .024$	$-.050 \pm .067$	$-.049 \pm .074$	$.000 \pm .000$	**0.08**	0.89	0.88	0.16
	Phase RH rel. LH	$.459 \pm .041$	$.513 \pm .113$	$.539 \pm .104$	$.368 \pm .094$	$.432 \pm .000$	1.32	1.96	2.25	**0.67**
	Phase LF rel. LH	$.333 \pm .047$	$.459 \pm .094$	$.428 \pm .062$	$.516 \pm .102$	$.334 \pm .295$	2.69	2.04	3.93	**0.02**
	Phase RF rel. LH	$.824 \pm .045$	$.906 \pm .165$	$.755 \pm .077$	$.835 \pm .206$	$.000 \pm .000$	1.86	1.53	**0.26**	18.48
	Mean (Coordination)						**1.49**	1.60	1.83	4.83
	Mean (Gait Pattern)						**0.95**	1.68	1.68	6.55
	Mean (All)						**0.59**	1.01	1.28	3.44
Trot	Stride Duration (ms)	230 ± 31	267 ± 10	130 ± 4	285 ± 92	318 ± 391	**1.19**	3.24	1.76	2.84
	Step Frequency (Hz)	4.42 ± 0.56	3.75 ± 0.22	7.71 ± 0.26	4.74 ± 2.81	4.41 ± 2.52	1.18	5.84	0.57	**0.01**
	Stance LH (ms)	97 ± 27	135 ± 16	60 ± 2	181 ± 48	20 ± 0	1.46	**1.38**	3.18	2.88
	Stance RH (ms)	97 ± 25	134 ± 6	70 ± 3	87 ± 50	1251 ± 2321	1.49	1.10	**0.40**	46.48
	Stance LF (ms)	104 ± 23	93 ± 5	38 ± 8	116 ± 18	10 ± 0	0.51	2.93	**0.50**	4.16
	Stance RF (ms)	104 ± 24	118 ± 5	51 ± 4	155 ± 27	20 ± 0	**0.60**	2.25	2.15	3.56
	Swing LH (ms)	134 ± 14	129 ± 3	70 ± 3	135 ± 26	110 ± 10	0.33	4.64	**0.06**	1.72
	Swing RH (ms)	133 ± 15	135 ± 5	60 ± 3	80 ± 14	190 ± 43	**0.10**	4.98	3.67	3.88
	Swing LF (ms)	126 ± 16	175 ± 5	92 ± 9	211 ± 22	308 ± 0	3.13	**2.18**	5.44	11.69
	Swing RF (ms)	126 ± 16	150 ± 4	79 ± 4	173 ± 7	298 ± 0	**1.47**	2.93	2.89	10.62
	Duty Factor LH	$.413 \pm .065$	$.506 \pm .061$	$.460 \pm .012$	$.636 \pm .169$	$.063 \pm .000$	1.43	**0.72**	3.43	5.42
	Duty Factor RH	$.416 \pm .060$	$.502 \pm .024$	$.537 \pm .025$	$.306 \pm .176$	3.931 ± 7.291	**1.42**	2.02	1.85	58.61
	Duty Factor LF	$.450 \pm .053$	$.348 \pm .017$	$.292 \pm .061$	$.407 \pm .064$	$.031 \pm .000$	1.95	3.01	**0.83**	7.96
	Duty Factor RF	$.448 \pm .055$	$.442 \pm .017$	$.391 \pm .030$	$.543 \pm .094$	$.063 \pm .000$	**0.11**	1.02	1.71	6.95
	Temporal Symmetry	$-.004 \pm .058$	$.008 \pm .068$	$-.075 \pm .017$	$.172 \pm .233$	$-.434 \pm .162$	**0.21**	1.23	3.06	7.46
	Phase RH rel. LH	$.488 \pm .060$	$.543 \pm .092$	$.308 \pm .045$	$.474 \pm .210$	$.401 \pm .347$	0.91	2.98	**0.24**	1.45
	Phase LF rel. LH	$.473 \pm .046$	$.620 \pm .075$	$.406 \pm .044$	$.699 \pm .121$	$.000 \pm .000$	3.21	**1.47**	4.94	10.35
	Phase RF rel. LH	$.975 \pm .070$	$.921 \pm .173$	$.900 \pm .142$	$.108 \pm .070$	$.126 \pm .000$	**0.77**	1.07	12.36	12.11
	Mean (Coordination)						**1.27**	1.69	5.15	7.84
	Mean (Gait Pattern)						**1.40**	1.76	3.62	14.69
	Mean (All)						**1.19**	2.50	2.72	11.01
Gallop	Stride Duration (ms)	192 ± 19	175 ± 6	140 ± 5	255 ± 91	172 ± 100	**0.85**	2.67	3.32	1.01
	Step Frequency (Hz)	5.27 ± 0.53	5.71 ± 0.23	7.15 ± 0.25	5.22 ± 2.54	7.95 ± 3.11	0.83	3.52	**0.09**	5.04
	Stance LH (ms)	78 ± 12	56 ± 6	75 ± 9	88 ± 53	15 ± 8	1.83	**0.22**	0.81	5.32
	Stance RH (ms)	74 ± 13	66 ± 6	53 ± 5	75 ± 50	16 ± 17	0.62	1.61	**0.02**	4.33
	Stance LF (ms)	76 ± 11	31 ± 4	32 ± 5	138 ± 40	0 ± 0	4.00	**3.92**	5.49	6.80
	Stance RF (ms)	84 ± 16	47 ± 6	55 ± 5	141 ± 28	0 ± 0	2.30	**1.76**	3.57	5.20
	Swing LH (ms)	114 ± 13	119 ± 3	65 ± 8	115 ± 46	172 ± 88	0.45	3.76	**0.12**	4.52
	Swing RH (ms)	117 ± 10	108 ± 6	87 ± 5	93 ± 13	140 ± 103	**0.86**	2.93	2.40	2.24
	Swing LF (ms)	115 ± 11	144 ± 5	108 ± 5	189 ± 30	172 ± 0	2.55	**0.66**	6.53	5.07
	Swing RF (ms)	108 ± 9	128 ± 6	85 ± 5	187 ± 9	172 ± 0	**2.16**	2.45	8.42	6.82
	Duty Factor LH	$.406 \pm .042$	$.321 \pm .033$	$.538 \pm .061$	$.343 \pm .207$	$.087 \pm .044$	2.04	3.14	**1.52**	7.63
	Duty Factor RH	$.386 \pm .040$	$.377 \pm .037$	$.377 \pm .034$	$.292 \pm .197$	$.094 \pm .098$	**0.23**	0.24	2.37	7.36
	Duty Factor LF	$.398 \pm .032$	$.179 \pm .025$	$.230 \pm .037$	$.540 \pm .157$	$.000 \pm .000$	6.75	5.17	**4.41**	12.28
	Duty Factor RF	$.433 \pm .048$	$.266 \pm .036$	$.394 \pm .036$	$.552 \pm .110$	$.000 \pm .000$	3.47	**0.82**	2.46	8.99
	Temporal Symmetry	$.025 \pm .059$	$-.084 \pm .060$	$.173 \pm .062$	$.086 \pm .311$	$-.095 \pm .254$	1.83	2.49	**1.02**	2.03
	Phase RH rel. LH	$.789 \pm .073$	$.692 \pm .084$	$.697 \pm .102$	$.446 \pm .183$	$.540 \pm .298$	1.33	**1.26**	4.73	3.44
	Phase LF rel. LH	$.560 \pm .037$	$.486 \pm .034$	$.790 \pm .047$	$.495 \pm .211$	$.000 \pm .000$	1.99	6.26	**1.76**	15.20
	Phase RF rel. LH	$.188 \pm .062$	$.202 \pm .044$	$.227 \pm .044$	$.168 \pm .037$	$.000 \pm .000$	**0.23**	0.64	0.32	3.05
	Mean (Coordination)						**1.35**	2.66	1.96	5.93
	Mean (Gait Pattern)						**2.29**	2.50	2.51	8.28
	Mean (All)						**1.91**	2.42	2.74	5.91

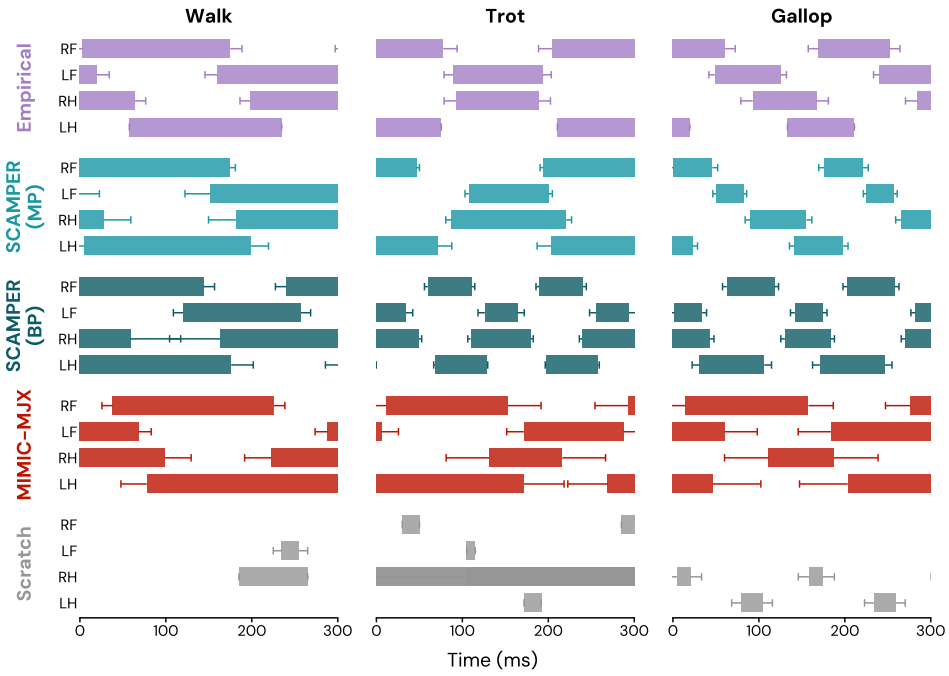


Figure 13: Stance-phase plots for each limb (RF, LF, RH, LH) across walk, trot, and gallop gaits, comparing empirical data (*purple*; [39]) to the base SCAMPER model (MP, *blue*), the behavior-prior-controlled SCAMPER model (BP, *dark blue*), MIMIC-MJX (*red*), and the from-scratch model (*gray*); bars show stance duration, error bars show variability across strides.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide extensive experimental evidence to demonstrate that the model allows for realistic motion during random generation and downstream task transfer.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations in Discussion

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The proofs are demonstrated in the appendix along with the full set of assumptions.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The implementation of each training stage, inference, and data analysis is fully described in the appendix, along with full lists of hyperparameters.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The paper provides access to the data and code in the supplementary material (with a publicly available repository on GitHub when we submit the camera-ready version). The README contains instructions to faithfully reproduce all training and experimentation.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: All of the training and test details are listed in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: For every presented metric we report the error either calculated across seeds (for task performance), across gait cycles, or bootstrapped means with 95% CIs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute used for this paper is discussed in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Due to the fact that this paper provides a model of rat motion and only deals with data collected from rats, there are no significant concerns related to Potential Harms or Societal Impacts. As well, the dataset used here was approved by the Animal Research Ethics board at the institution where the data was collected.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: A model of rat motion does not have any significant societal impact outside of neuro-AI research.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All third party data that was used has proper citation and link to their work as per their licensing.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.