Acknowledgments

This work was supported by the Academy of Finland Flagship programme: Finnish Center for Artificial Intelligence FCAI and by Academy of Finland through grant number 328399. We acknowledge the computational resources provided by the Aalto Science-IT project. The data used in this project was obtained from mocap.cs.cmu.edu and was created with funding from NSF EIA-0196217.

Appendix

A Imitation Learning algorithms

Here we describe the core RL and imitation algorithms we used in this work. For consistency and following the large ablation study conducted by [26], we used SAC [16] as the RL algorithm for all methods. SAC is an actor-critic algorithm which optimizes a soft Q-function with the loss

$$\mathcal{L}(Q) = \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{a}_t,\boldsymbol{s}_{t+1}\sim\tau_\pi}\left[\frac{1}{2}(Q(\boldsymbol{s}_t,\boldsymbol{a}_t)-\hat{Q})^2\right], \tag{10}$$

with the target

$$\hat{Q} = r(\boldsymbol{s}_t,\boldsymbol{s}_{t+1}) + \gamma\mathbb{E}_{\boldsymbol{a}_t\sim\pi}Q(\boldsymbol{s}_{t+1},\boldsymbol{a}_t), \tag{11}$$

and the policy loss

$$\mathcal{L}(\pi) = \mathbb{E}_{\boldsymbol{s}_t\in\tau_\pi}\left[\alpha_{\text{SAC}}\log\pi(\boldsymbol{a}_t|\boldsymbol{s}_t) - Q(\boldsymbol{s}_t,\boldsymbol{a}_t)\right].$$

including the automatic entropy tuning of α_{SAC} introduced in [17].

Generative Adversarial Imitation Learning The authors of GAIL used TRPO to maximize the adversarial reward generated by the GAIL discriminator. Adapting it to SAC is straightforward, as only the reward depends on the discriminator output.

State-Alignment Imitation Learning In addition to the adversarial-style rewards SAIL uses a modified policy objective which adjusts the policy towards a *policy prior*

$$\pi_p(\boldsymbol{a}_t|\boldsymbol{s}_t) \propto \exp\left(-\left\|\frac{g_{\text{inv}}(\boldsymbol{s}, f(\boldsymbol{s}))}{\sigma}\right\|^2\right) \tag{12}$$

where g_{inv} is an inverse dynamics model trained using transitions sampled from the policy, $f(\boldsymbol{s})$ is a $\beta-$Variational Auto Encoder (VAE) trained using demonstration data, and σ is a constant (see Liu et al. [23] for details). We use 50K timesteps worth of data using random actions to pretrain the inverse dynamics and pretrain the VAE.

The authors used on-policy PPO as the base RL algorithm. In order to use SAC, we make the following adjustments to the SAC policy objective:

$$\begin{aligned}\mathcal{L}(\pi) = \mathbb{E}_{\boldsymbol{s}_t\in\tau_\pi}\big[&\alpha_{\text{SAC}}\log\pi^I(\boldsymbol{a}_t|\boldsymbol{s}_t) - Q(\boldsymbol{s}_t,\boldsymbol{a}_t)\\ &+(\pi^I(\boldsymbol{a}_t|\boldsymbol{s}_t) - \pi_p(\boldsymbol{a}_t|\boldsymbol{s}_t))^2\big],\end{aligned} \tag{13}$$

where α_{SAC} is the entropy scalar tuned as in [17]. In this work, we include the gradient penalty term introduced in [13] to the discriminator loss.

B Further Discussions of the Co-Imitation Framework

In this section we give further intuition for matching the proposed trajectory distributions of expert and imitator by using two variations of the Kullback-Leibler (KL) divergence as example divergence to minimize. Our aim is to provide further intuition and discuss and shed a light at the unique properties of the selected co-imitation problems presented in this paper. We will start the discussion with a look at the standard imitation learning problem (i.e. behaviour cloning) in the context of co-adaptation. Thereafter, we discuss further the co-imitation setting selected for CoIL, namely to match the state distributions between imitator and expert. As a reminder, the trajectory distribution of the expert is given by

$$q(\tau) = q(\mathbf{s}_0) \prod_{t=0}^{T-1} q(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\pi^E(\mathbf{a}_t|\mathbf{s}_t), \tag{14}$$

while the imitator trajectory distribution is dependent on the imitator policy $\pi^I(\mathbf{a}|\mathbf{s}, \xi)$ and chosen morphology ξ

$$p(\tau|\pi^I, \xi) = p(\mathbf{s}_0|\xi) \prod_{t=0}^{T-1} p_I(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \xi)\pi^I(\mathbf{a}_t|\mathbf{s}_t, \xi). \tag{15}$$

To improve readability and for the purpose of this discussion we will assume a shared state S and action A spaces for both imitator and expert.

B.1 Behavioural Cloning in the Co-Imitation Setting

While classic imitation learning is concerned with the problem of minimizing

$$\min_{\pi^I} D(q(\tau), p(\tau|\pi^I)) \tag{16}$$

for a measure or divergence $D(\cdot, \cdot)$ with matching transition probabilities, we consider in this work the extension where we aim to optimize

$$\min_{\pi^I, \xi} D(q(\tau), p(\tau|\pi^I, \xi)), \tag{17}$$

assuming a parameterization of the imitator transition probability of $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}, \boldsymbol{a}, \xi)$. In our setting, we assume the variable ξ to be morphological parameters such as lengths, sizes, weights or other shape inducing variables. These parameters are observable and not latent as we assume them to be needed for the instantiation of the simulation[2], or production of the hardware components via 3D-printing, for example.

While many potential choices for $D(\cdot, \cdot)$ exist (see e.g. Ghasemipour et al. [12] for some options) we will provide some further intuition behind the proposed co-imitation learning problem by utilizing the KL-divergence. Applying the KL-divergence to the problem in Eq. 17 results in

$$\begin{aligned} D_{\text{KL}}\left(q(\tau)||p(\tau|\pi^I, \xi)\right) &= \int_\tau q(\tau) \ln \frac{q(\tau)}{p(\tau|\pi^I, \xi)} \\ = \underbrace{\mathbb{E}_{q(\tau)}\left[\ln\left(\frac{q(\boldsymbol{s}_0)}{p(\boldsymbol{s}_0|\xi)}\right)\right]}_{\text{match initial state distribution}} &+ \underbrace{\mathbb{E}_{q(\tau)}\left[\ln\left[\frac{\prod q(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})}{\prod p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}, \xi)}\right]\right]}_{\text{match transition distribution}} \\ &+ \underbrace{\mathbb{E}_{q(\tau)}\left[\ln\left[\frac{\prod \pi^E(\boldsymbol{a}|\boldsymbol{s})}{\prod \pi^I(\boldsymbol{a}|\boldsymbol{s})}\right]\right]}_{\text{match expert policy}}, \end{aligned} \tag{18}$$

where we can see that the equation can be rearranged into three problems: (1) matching the initial state-distribution, (2) matching the transition distributions, and (3) matching the policies of imitator and expert. For better readability the notation of $\boldsymbol{s}$ for the current state and $\boldsymbol{s}'$ for the following state is used. The expectation is in respect to the trajectory distribution $q(\tau)$ of the expert, which in practice is replaced by a set of sampled trajectories. One can re-formulate Eq. 18 by using the distribution of state, action and next state, induced by the trajectory distribution of the expert leading to a simplified form with

$$\begin{aligned} \int_{\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'} q(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') \left(\ln\left(\frac{q(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})}{p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}, \xi)}\right) + \ln\left(\frac{\pi^E(\boldsymbol{a}|\boldsymbol{s})}{\pi^I(\boldsymbol{a}|\boldsymbol{s})}\right)\right] \\ \approx \mathbb{E}_{q(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')}\left[-p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}, \xi) - \pi^I(\boldsymbol{a}|\boldsymbol{s})\right] + C, \end{aligned} \tag{19}$$

---

[2] I.e. these parameters are needed for construction of the URDF/XML files needed for simulation

where we show the resulting morphology-behaviour optimization objective plus the constant term. It is worth to note that thus far we have been using $\pi^I(\boldsymbol{a}|\boldsymbol{s})$ for the policy distribution, i.e. a policy π^I which does not depend on ξ. However, in practice[3] one may want to utilize a policy $\pi^I(\boldsymbol{a}|\boldsymbol{s},\xi)$ which is capable of predicting optimal (imitation) actions given both the current state and morphology of the imitator[4]. Thus, leading to the alternative objective function

$$\mathbb{E}_{q(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')}\left[-p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a},\xi)-\pi^I(\boldsymbol{a}|\boldsymbol{s},\xi)\right]. \tag{20}$$

B.2 Co-Imitation Learning by State Distribution Matching

While the equation above depends on the knowledge of the imitator transition distribution, an alternative to the chosen KL-divergence for trajectories as objective is to consider state-distribution matching divergences. Here, we want to compute a measure for

$$D(q(\boldsymbol{s}|\pi^E),p(\boldsymbol{s}|\pi^I)). \tag{21}$$

It is straight forward to show that knowledge of the trajectory distributions $q(\tau)$ and $p(\tau|\pi^I,\xi)$ allows us to derive the state distributions given the respective policy with

$$\begin{aligned} p(\boldsymbol{s}|\pi) &= \int_\tau p(\boldsymbol{s},\tau|\pi) = \int_\tau p(\boldsymbol{s}|\tau,\pi)p(\tau|\pi) \\ &= \mathbb{E}_{p(\tau|\pi)}\left[p(\boldsymbol{s}|\tau)\right] = \mathbb{E}_{p(\tau|\pi)}\left[\frac{1}{T}\sum_{t=0}^{T}\mathbb{1}(\boldsymbol{s}_t=\boldsymbol{s})\right]. \end{aligned} \tag{22}$$

Intuitively, the state distribution $p(\boldsymbol{s}|\pi)$ can be computed by sampling trajectories with the policy π and computing the expected occurrence of a state $\boldsymbol{s}$ in a trajectory τ. This insight allows us to develop the presented co-imitation learning method using state-distribution matching without requiring access to the true transition probability $p(\boldsymbol{s}^I|\boldsymbol{s},\boldsymbol{a},\xi)$ of the imitator like in Eq. 20.

While the main paper proposes to use a Wasserstein distance, we will present here some analysis using the KL-divergence as before. Using our state distributions defined above we arrive at the following objective with

$$\begin{aligned} D(q(\boldsymbol{s}|\pi^E),p(\boldsymbol{s}|\pi^I)) &\overset{\text{def}}{=} \mathrm{KL}(q(\boldsymbol{s}|\pi^E)\parallel p(\boldsymbol{s}|\pi^I)) \\ &= \mathrm{KL}\left(q(\boldsymbol{s}|\pi^E)\middle\|\mathbb{E}_{p(\tau|\pi^I,\xi)}\left[\frac{1}{T}\sum_{t=0}^{T}\mathbb{1}(\boldsymbol{s}_t=\boldsymbol{s})\right]\right) \qquad (23) \\ &= -\mathbb{E}_{q(\boldsymbol{s}|\pi^E)}\left[\ln\mathbb{E}_{p(\tau|\pi^I,\xi)}\left[\frac{1}{T}\sum_{t=0}^{T}\mathbb{1}(\boldsymbol{s}_t=\boldsymbol{s})\right]\right]+C, \qquad (24) \end{aligned}$$

where we move terms which do not depend on π^I or ξ into a constant C. We can now rearrange this objective with

$$\begin{aligned} &-\mathbb{E}_{q(\boldsymbol{s}|\pi^E)}\Bigg[\ln\Bigg(\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}_{\boldsymbol{s}_0\sim p(\boldsymbol{s}_0|\xi)}[\mathbb{E}_{\boldsymbol{a}_1\sim\pi^I(\boldsymbol{a}|\boldsymbol{s}_0,\xi)}[\mathbb{E}_{\boldsymbol{s}_1\sim p(\boldsymbol{s}|\boldsymbol{s}_0,\boldsymbol{a}_0,\xi)}[\cdots \\ &\mathbb{E}_{\boldsymbol{a}_{t-1}\sim\pi^I(\boldsymbol{a}|\boldsymbol{s}_{t-1},\xi)}[\cdots\mathbb{E}_{\boldsymbol{s}_t\sim p(\boldsymbol{s}|\boldsymbol{s}_{t-1},\boldsymbol{a}_{t-1},\xi)}\left[\mathbb{1}(\boldsymbol{s}_t=\boldsymbol{s})\right]]\cdots]]\Bigg)\Bigg], \end{aligned} \tag{25}$$

which computes the expected probability of being in state $\boldsymbol{s}$ at the end of a sub-trajectory unrolled for t timesteps, where $\boldsymbol{s}$ is sampled from the expert state distribution. To simplify this objective and make it tractable for optimization we can apply Jensen's inequality and arrive at a lower bound with

$$\begin{aligned} -T\cdot\text{Eq. (25)} \geq \sum_{t=0}^{T}\mathbb{E}_{q(\boldsymbol{s}|\pi^E)}\Bigg[\ln\Bigg(&\mathbb{E}_{\boldsymbol{s}_0\sim p(\boldsymbol{s}_0|\xi)}[\mathbb{E}_{\boldsymbol{a}_1\sim\pi^I(\boldsymbol{a}|\boldsymbol{s}_0,\xi)}[\mathbb{E}_{\boldsymbol{s}_1\sim p(\boldsymbol{s}|\boldsymbol{s}_0,\boldsymbol{a}_0,\xi)}[\cdots \\ &\mathbb{E}_{\boldsymbol{a}_{t-1}\sim\pi^I(\boldsymbol{a}|\boldsymbol{s}_{t-1},\xi)}[\cdots\mathbb{E}_{\boldsymbol{s}_t\sim p(\boldsymbol{s}|\boldsymbol{s}_{t-1},\boldsymbol{a}_{t-1},\xi)}\left[\mathbb{1}(\boldsymbol{s}_t=\boldsymbol{s})\right]]\cdots]]\Bigg)\Bigg], \end{aligned} \tag{26}$$

---

[3] As we do in the proposed co-imitation learning method utilizing state-distribution matching.

[4] Note that different actions may be optimal depending on the current morphology of the agent.

where we removed the negative sign and consider this as a maximization problem in respect to the imitator policy π^I and imitator morphology ξ. The main insight we get from this exercise is that, unlike in the simpler behavioural cloning case discussed above in Equations 18 and previous, the behavioural policy π^I and morphology ξ are here inherently entangled and have to be optimized concurrently, i.e. a separation is not possible without further assumptions or simplifications. While in Eq. 18 we were able to separate the optimization problem into clearly defined components for matching transition probabilities (depending on ξ) and matching the expert and imitator policies, we find that this is not the case for state-distribution matching as derived[5] in Eq. 26.

C Experimental set-up

C.1 Q-function baseline

Here we describe how we adapt the morphology optimization procedure described in Luck et al. [24] to the imitation learning setting to serve as a baseline. To do this, we use the Q-function learned by SAC on the SAIL reward as a surrogate for computing returns from entire episodes and optimizing the morphology using those returns. We use a linearly decreasing ϵ-greedy exploration schedule and Particle Swarm Optimization [7] (as proposed by the original authors) to find the best morphology according to the Q-function in the case of exploitation. For exploration episodes, we sample morphology parameters uniformly from within the bounds described by table 6. The ϵ is linearly reduced over 1 million timesteps.

C.2 Tasks

We extract marker positions and velocities from the CMU motion capture data [4] and apply preprocessing, such as resampling from 120hz to the MuJoCo speed of 66.67hz. For all tasks we include the preprocessed demonstrator data in the code supplement, as well as the preprocessing code.

C.3 HalfCheetah tasks

In the HalfCheetah tasks we directly optimize the lengths of each limb. The lower bound for optimization is 1e-6 and the upper bound is twice the original value. When the imitator has three leg segments, there are a total of six parameters to optimize, while for the other case there are four. Figure 8 gives the learned parameters for the 2to3 task, as well as the meaning of each parameter.

Humanoid tasks

For the Humanoid tasks we optimize a scaling factor for the torso, legs and arms. The standard MuJoCo Humanoid corresponds to scaling factors $[1, 1, 1]$. Table 6 gives the bounds for this optimization. The specific motions and subject IDs are detailed in the code supplement. Due to variable amount of data available, all three Humanoid tasks have a different amount of demonstrator trajectories and all episode lengths are different. For ”Soccer kick” the data includes multiple subjects.

C.4 Co-Imitation Learning Hyper-parameters

Table 1 gives the common hyper-parameters used for all experiments, while Table 4 gives SAIL-specific hyper-parameters and Table 5 gives GAIL-specific hyper-parameters. Table 3 gives hyper-parameters for the Q-function baseline adapted from [24]. Batch size corresponds to the size of mini-batch used for discriminator and policy updates. γ is the discount factor. VAE scaler is the balancing term between the policy prior and the SAC policy loss (see Eq. (13)). Disc decay is the weight decay multiplier for the discriminator weights. Updates per step means how many policy and discriminator updates we take for each environment timestep. SAC τ is the soft target network update constant.

---

[5] Here for the case of the KL divergence.

Table 1: Hyper-parameters values shared throughout all experiments.

Hyperparameter	Value
Batch size	1024
γ	0.97
Use transitions	False
disc decay	0.00001
All networks	MLP
Network layers	3
Hidden nodes	200
Activation	ReLU
normalize obs	False
Updates per step	1
Q weight decay	1e-5
Optimizer	Adam
Entropy tuning	True
SAC τ	0.005
learning rate	0.0003

Table 2: Task-specific hyper-parameter values

Hyperparameter	Value
Cheetah episode length	1000
Humanoid max episode length	300
Cheetah early termination	False
Humanoid early termination	True
Max steps	1.5M

Table 3: Q-function baseline-specific hyper-parameters values.

Hyperparameter	Value
Morphology Optimizer	PSO
PSO particles	250
PSO iters	250
ϵ decay over	1M steps

Table 4: SAIL-specific hyper-parameters values.

Hyperparameter	Value
VAE scaler	1
β of VAE	0.2

Table 5: GAIL-specific hyper-parameters values.

Hyperparameter	Value
Reward style	AIRL
Use $\log r(\cdot)$	True

Table 6: Amount of morphology parameters E_ξ and their bounds in each environment. Here "2x" means the optimization is bounded to twice the default size. We use (T) for torso, (L) for legs, (S) for segments and (H) for hands.

Environment	E_ξ	Bounds	Description
Cheetah	6	$0 - 2$x	Two L, three S
2-seg Cheetah	4	$0 - 2$x	Two L, two S
Humanoid	3	0.5x $- 2$x	T, H and L scale

Table 7: Software packages used

Software	Version used
Python	3.8
PyTorch	1.12
NumPy	1.23
GPy	1.10.0
Gym	0.24
MuJoCo	2.1
mujoco-py	2.1.2
GPy	1.10
GPyOpt	1.2.6

C.5 Morphology Optimization details

As described in Section 5.2 we use Bayesian Optimization for proposing the next candidate morphology. As surrogate model we use a basic Gaussian Process regression model. We use the Matern52 kernel and a constant mean function. The kernel and mean function hyper-parameters are trained by minimizing the negative marginal log-likelihood (MLL), where the kernel function uses automatic relevance determination (ARD) [31]. The optimization method selected is L-BFGS algorithm.

In order to compute the next morphology candidate ξ_{next} we compute the predictive posterior distribution for the test morphologies $\tilde{\xi} \in \mathbb{R}^{M \times E_\xi}$, where M is the number of test candidates to evaluate and E_ξ is the dimension of morphology attributes. The range of values for the test morphologies is shown in Table 6, where the bounds column indicates the range for each of the morphology parameters in the respective environment.

The basic Gaussian Process regression model complexity is $\mathcal{O}(N^3)$, where N is the number of samples used for training [31]. Thus, the GP suffers from poor scalability and using the entire data-set of collected imitation trajectories and morphologies Ξ would highly increase the algorithm training process. In addition, as discussed in Section 5.2 policies that were evaluated early in the training have worse performance than the recent ones, and hence, we have lower confidence on the performance of those morphologies. For this reason, we limit the number of previous morphologies to consider to $N = 200$, so that only the most recent policies and morphologies are taken into account for modeling the relationship between the distance distribution and the morphology parameters.

D Hardware and software

D.1 Hardware and runtime

We ran our experiments on a Dell PowerEdge C4140 with 2x8 core Intel Xeon Gold 6134 3.2GHz and a V100 32GB GPU. The Humanoid experiments took around 24h to finish on this setup, while the HalfCheetah took 22h.

D.2 Software versions used

The experiments were ran on a cluster running CentOS 7. The exact software packages used are reported in Table 7

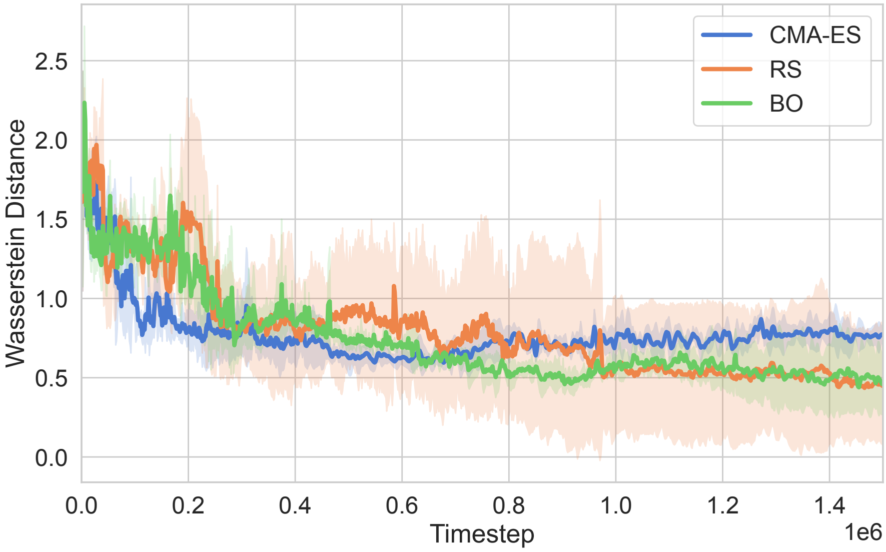

Figure 7: Wasserstein distance for the jogging experiment using CMA-ES (blue), Random Search (orange) and Bayesian Optimization (green). Both RS and BO perform comparably on average, but BO has much lower variance.

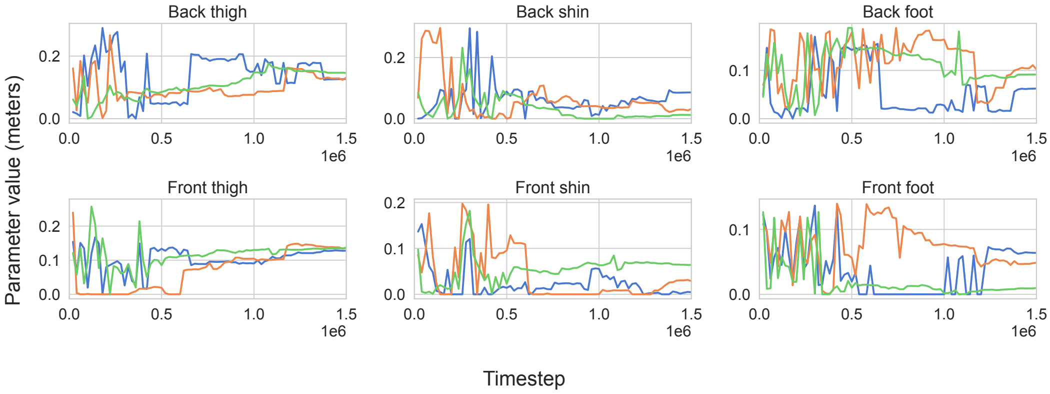

Figure 8: Morphology parameters learned by CoIL for three seeds in the 2to3 task. Note how all seeds set either the shin or the foot to close to zero which allows very close matching of the real demonstrator.

E Experimental results

E.1 Choice of the morphology optimizer

In addition to the proposed Algorithm 2 based on BO, we evaluate two other algorithms to optimise the morphology parameters: Random Search (RS) [2], and covariance matrix adaptation evolutionary strategy (CMA-ES) [18]. As discussed in Algorithm 1, the policies are trained using the same morphology for $N_\xi = 20$ episodes, as changing it too often would hinder learning. For both CMA-ES and RS we follow a similar procedure where the optimization algorithms take as input the observations $X = \{\xi_n\}, \forall \xi_n \in \Xi$ and use as targets $Y = \{y_n\}$, $\forall(\xi_n, \mathrm{T}^I_n) \in \Xi$, where y is computed following Eq. (9). As opposed to BO, we use the entire data-set Ξ as data points in CMA-ES since its performance does not suffer from the number of samples. By contrast, our implementation of RS keeps the best previous morphology and randomly proposes a new morphology.

The results for each optimization method in the Humanoid jogging experiment are shown in Figure 7. Since the number of different morphologies we can evaluate is relatively low, the BO approach benefits from the low data regime, presenting low mean and variance. Similarly, the RS results present a low mean but higher variance due to the randomness inherent in the algorithm. By contrast, we can observe that the performance of CMA-ES is lower than both BO and RS as it suffers from the low number of morphologies evaluated, getting stuck in a local optima. These results shows that exploring the morphologies using the BO algorithm is beneficial for the task of co-imitation, as we can find an optimal solution while evaluating a low number of morphologies and keeping a relatively low variance.

E.2 Analysis of morphology parameters

Here we show the evolution of the morphology as a function of time for each seed of the main CoIL experiment. Figure 8 gives the plots for each parameter, where the parameter value is the argmax of the GP mean, i.e the best morphology so far according to the GP.

In this task we imitate a simpler 2-leg-segment cheetah using a 3-leg-segment cheetah. It is possible for the imitator to adapt its morphology in such a way that that it matches the demonstrator exactly, by setting either both shins, both feet or one of each to zero. We can see in 8 that all seeds set either the shin or the foot to close to zero, meaning they are able to closely replicate the demonstrator morphology.