# PreCo: Enhancing Generalization in Co-Design of Modular Soft Robots via Brain-Body Pre-Training

**Anonymous Author(s)**
Affiliation
Address
email

**Abstract:** Brain-body co-design, which involves the collaborative design of control strategies and morphologies, has emerged as a promising approach to enhance a robot's adaptability to its environment. However, the conventional co-design process often starts from scratch, lacking the utilization of prior knowledge. This can result in time-consuming and costly endeavors. In this paper, we present PreCo, a novel methodology that efficiently integrates brain-body pre-training into the co-design process of modular soft robots. Our approach is based on the insight of encoding co-design principles into models, achieved by pre-training a pleiotropic co-design policy on a diverse set of tasks. The pre-trained co-designer is utilized to generate initial designs and control policies and then fine-tune them for a specific co-design task. Through experiments on a modular soft robot system, our method demonstrates zero-shot generalization to unseen co-design tasks and enables rapid adaptation with significantly reduced iterations (See video).

**Keywords:** Co-design, Pre-training, Modular Soft Robots

## 1 Introduction

Nature does not treat the development of the brain and body as separate processes, indicating that cognitive processes are intricately connected to the body and the external environment in which organisms operate [1, 2]. This theory holds significant implications for the robotics community. To enable effective interaction with the environment, it is essential to prioritize the co-design of both physical bodies and control systems of robots. For instance, Modular Soft Robots (MSRs) considered in this work, are a promising category of flexible robotic systems that offer designers the ability to construct robot bodies by combining various types of deformable cubes and adjusting their volume to generate control signals [3]. If a robot needs to perform tasks involving climbing stairs, it can be designed with strong legs for enhancing strength and stability. Conversely, if the robot is intended to traverse uneven terrains, equipping it with slender legs can provide better adaptability. (Figure 1). Currently, the majority of co-design studies in the literature [3, 4, 5, 6, 7] mainly focus on "one robot one task", where the primary objective is to discover an optimal robot morphology and controller for a specific task. However, this approach seems to diverge from biological morphologies, such as the human body, which inherently possesses the ability to perform multiple tasks.

As a matter of fact, even improving a robot morphology for a single task can be highly challenging due to: (1) the presence of a severe combinatorial explosion within the robot design space and (2) the existence of incompatible state-action spaces that necessitate training a separate control policy for each morphology. Consequently, past studies [8, 9, 10, 11, 12] often address these challenges by considering the evolution of body and control as separate processes, directly conducted within the large, high-dimensional design space. In other words, these methods typically start from scratch without utilizing prior knowledge of co-design, which is expensive and inefficient.

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

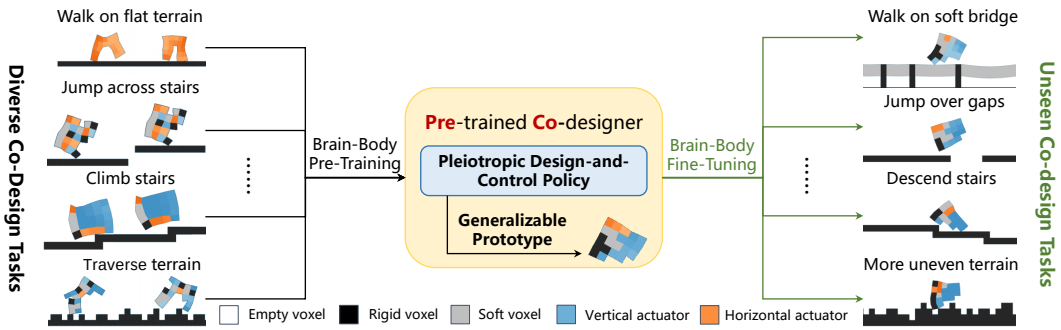


Figure 1: Workflow of PreCo. At the heart of PreCo lies a pleiotropic co-design policy, which undergoes pre-training using end-to-end reinforcement learning on a diverse set of co-design tasks. The resulting pre-trained co-designer is then utilized to generate initial designs and control policies, which are further fine-tuned for unseen tasks.

In this paper, we take a step towards a more challenging setting with the aim of designing a single modular soft robot that can perform a wide variety of tasks. To achieve this, we propose PreCo (Figure 1), a methodology to learn a universal co-designer. Our method entails pre-training a co-design policy on a diverse set of co-design tasks, enabling the model to grasp the interdependencies between the brain, body and tasks. Then, we fine-tune the model on unseen co-design tasks to enhance its adaptability and performance. Specifically, we propose to use deep reinforcement learning to train a pleiotropic co-design policy in an end-to-end manner. This approach implies that both the morphologies and control strategies of a robot stem from a single genome (parameters of the policy), where any mutation to the genome simultaneously affects the "brain" and "body." In contrast to conventional co-design methods that segregate not only the optimization processes of the brain and body but also the genomes that generate them, we draw inspiration from pleiotropy in biology, which refers to a single gene expressing multiple phenotypic traits [13, 14, 15]. By using a universal representation of the co-design policy, our method eliminates the need for a robot population and empirically demonstrates that pleiotropy facilitates a more effective exploration of the search space.

Our study offers the following key contributions: Firstly, we introduce PreCo, a novel approach that learns a universal design-and-control policy capable of handling multiple challenging co-design tasks for modular soft robots. Secondly, we showcase the effectiveness of brain-body pre-training, which facilitates fast adaptation to new co-design tasks by leveraging past co-design knowledge. Thirdly, through empirical analysis, we demonstrate that our pleiotropic co-design policy exhibits greater robustness in terms of mitigating premature convergence. The pleiotropic configuration excels at capturing the intrinsic brain-body link, resulting in improved sample efficiency and enhanced flexibility. Furthermore, our work provides the first experimental comparison among meta-learning, curriculum learning and pre-training methods in addressing co-design problems.

## 2 Related Work

**Robot Co-Design** The process of developing both the physical body and the cognitive capabilities in nature is intricately intertwined. To replicate this fundamental principle, robot designers are tasked with concurrently optimizing the morphology and control strategy of robots. In the field of Evolutionary Robotics (ER), researchers have extensively investigated the application of Evolutionary Algorithms (EAs) for co-designing robotic systems [8, 10, 12, 16]. A prominent focus in this field lies in the representation of robot morphologies. Various approaches, such as generative encoding schemes [17, 18], have been explored to facilitate the discovery of novel and efficient designs. Techniques like neural networks [19, 20], Neural Cellular Automata (NCA) [21, 22, 23] and Compositional Pattern-Producing Networks (CPPNs) [24, 25] have been employed to generate diverse and complex robot morphologies, enabling the exploration of a broad design space. In addition, EAs can also be integrated with Reinforcement Learning (RL), allowing robots to evolve and improve

their behaviors through interactions with the environment [3, 26]. These approaches, however, adopt separate genomes for the robot body and control, which are optimized in a bi-level co-design fashion. Consequently, they rely on a population of design prototypes to facilitate exploration, leading to challenges in terms of sample efficiency and computational requirements. In contrast to these population-based approaches, we propose an alternative methodology that utilizes a universal policy representation, enabling the brain and body components to be derived from the same genome and jointly optimized. Through empirical experiments, we demonstrate that this pleiotropic representation significantly facilitates exploration of the design space and effectively captures the inherent brain-body connection, leading to enhanced sample efficiency and increased flexibility.

In addition to EA methods, when we have access to certain aspects of the system's physical dynamics, a model-based differentiable simulator can be employed to jointly optimize the design parameters and control using Back-Propagation Through Time (BPTT) [27, 28, 29, 30, 31, 32]. In contrast, our work specifically addresses the model-free setting, where system modeling is not required. To achieve this, Policy Gradient (PG) methods can be used to approximate the gradient of design parameters or evaluate the fitness of a robot morphology [4, 6, 33, 34, 35]. Building upon this technology, PreCo is introduced as a novel approach that distinguishes itself from previous works by tackling a more challenging objective of addressing multiple co-design tasks simultaneously.

**Multi-task Reinforcement Learning** In this study, we approach the challenge of brain-body pre-training by adopting a Multi-task Reinforcement Learning (MTRL) framework, which has garnered considerable interest in the field of embodied intelligence [36, 37, 38, 39, 40]. MTRL involves the training of an agent to perform multiple tasks concurrently, aiming to leverage shared knowledge across tasks for improved learning efficiency and generalization. However, previous research based on the Transformer structure primarily focuses on training a universal controller for multiple robot bodies [11, 41, 42, 43, 44, 45]. PreCo takes a further step by not only focusing on learning a single brain-body co-design policy to leverage similarities and differences among co-design tasks but specifically exploring how the intrinsic brain-body link of various tasks can be utilized to improve efficiency and generalization when encountering new co-design tasks. In essence, our method aims to avoid learning from scratch, which shares a similar spirit but differs in the framework from curriculum learning [6, 19, 46] and meta-learning [26, 47, 48, 49].

## 3 Preliminaries

**Reinforcement Learning** In our study, we approach the problem of brain-body pre-training for a set of $K$ co-design tasks of modular soft robots by formulating it as a MTRL problem. In the domain of RL, the problem is typically formulated as a Markov Decision Process (MDP), defined by a 5-tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$. Here, $\mathcal{S}$ represents the state space and $\mathcal{A}$ represents the action space. The transition function $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ determines the probability of transitioning from one state to another given a specific action. The reward function $r(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ assigns a numeric value to the state-action pairs, indicating the desirability of taking a particular action in a given state, and the discount factor $\gamma \in (0, 1]$ specifies the degree to which rewards are discounted over time. Our goal is to find policy parameters $\theta$ which can maximize the average expected reward across all co-design tasks: $\frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{\infty}\gamma^t r_t^k(s_t, a_t)$, here the policy is represented by a deep neural network parameterized as $\pi_\theta(a_t|s_t)$, which maps from states to distributions over actions.

We employ Proximal Policy Optimization (PPO) [50], a popular RL algorithm that is widely used in a variety of robot tasks. The algorithm utilizes a surrogate objective function that approximates the policy gradient, and the objective function of PPO is:

$$J(\theta) = \mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t - \beta D_{KL}(\pi_{\theta_{old}}(\cdot|s_t)\|\pi_\theta(\cdot|s_t))\right] \tag{1}$$

where $\hat{A}_t$ is the advantage estimation, $\mathbb{E}_t$ represents the empirical average over a batch of generated samples. By iteratively collecting experiences and optimizing $J(\theta)$, the policy $\pi_\theta(a_t|s_t)$ is updated in the direction of maximizing the cumulative reward.

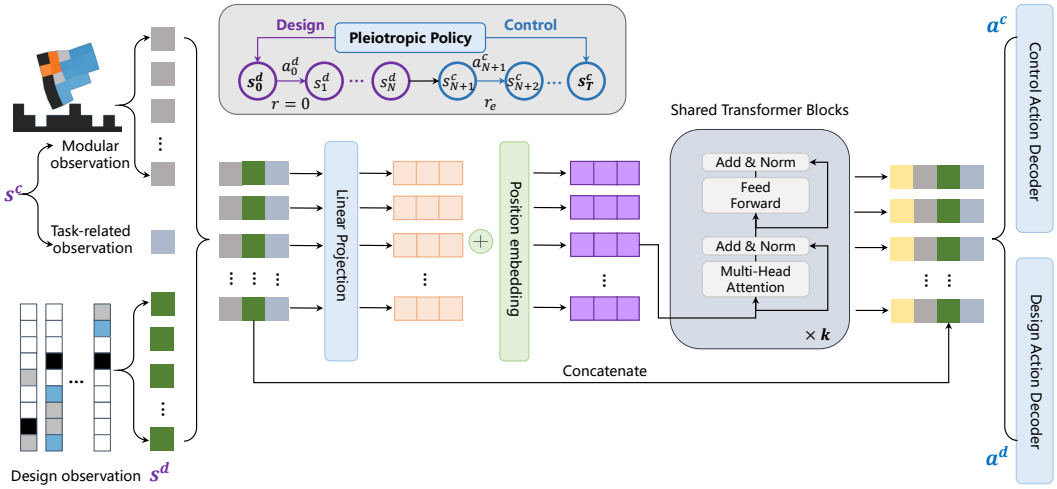


Figure 2: Pleiotropic co-design policy. Our policy is designed with a shared structure that influences both morphology and control. It receives unified design-and-control observations and generates corresponding actions. This pleiotropic policy operates under the framework of reinforcement learning, where the design and control processes are unified as a single MDP.

**Transformer.** Transformer [51] is a popular neural network architecture that has revolutionized the domain of natural language processing and computer vision [52, 53, 54, 55, 56], and has become a fundamental component in many cutting-edge models. At its core, Transformer employs a powerful self-attention mechanism to capture the dependencies and relationships among elements in a sequence. It allows the model to dynamically allocate attention to different parts of the input sequence based on their relevance. In each self-attention layer, attention weights are computed for each element by considering its interactions with all other elements in the sequence. These weights play a crucial role in aggregating information from the entire sequence, enabling the Transformer to generate comprehensive and informative representations for each element. Furthermore, the Transformer's encoder incorporates feed-forward neural networks, which introduce non-linear transformations and further enhance the quality of the representations.

The Transformer architecture is particularly well-suited for modular robot systems due to its ability to effectively handle sequences, which aligns with the nature of modular design and control observations. By leveraging the Transformer, we can tackle the challenge of dealing with incompatible state-action spaces arising from different morphologies. Additionally, inspired by the adaptive weights, our method aims to use this mechanism to capture the intrinsic link between morphology and control in diverse robot co-design tasks.

## 4 Brain-Body Pre-Training

In this section, we introduce our pleiotropic co-design policy, which is a transformer-based neural network designed for brain-body pre-training. Our policy is capable of being optimized via end-to-end reinforcement learning over unified design and control spaces. The actor network of this policy is outlined in Figure 2, and note that the critic network has the same architecture.

**Pleiotropic Co-Design Policy** We start by reviewing co-design methods [6, 35] that utilize RL to approximate the gradient of design and control parameters. These methods integrate design and control into a unified decision process, as depicted by the grey box in Figure 2. At the start of each episode, a dedicated design policy takes a finite number of actions in order to develop a robot morphology. Subsequently, the designed robot is consumed by a control policy to collect the environmental rewards. Through backpropagation, both policies will be informed and updated to maximize the performance. However, a notable issue of this approach is the presence of an imbalanced sample

distribution. For instance, the design steps in each episode are relatively short (e.g., 5) compared to the significantly longer execution steps (e.g., 500). Since separate policy representations are employed in these methods, the gradient information flowing to the design policy is substantially lower than that reaching the control policy when we use randomly sampled experiences to update both policies. Consequently, the design policy can quickly become optimized only for a limited region around the local morphological optimum, which hinders effective exploration of the design space.

To address this concern, our co-design policy is specifically designed to facilitate more information sharing between the two sub-policies, as shown in Figure 2. While it is challenging to explicitly represent the interconnections between morphology and controller, we employ shared parameters to implicitly capture their relationships. This approach ensures that both the "brain" and the "body" of a robot are derived from a single genome and develop together.

**Unified State-Action Space** Our study focuses on co-designing flexible MSRs that consist of various types of blocks, also known as voxels. Each voxel is represented by a discrete value that corresponds to its material or type of actuator. In practice, we employ one-hot encoding to represent these values. The pleiotropic co-design policy is uniformly denoted as $\pi_\theta(a_t|s_t)$ and integrated into the aforementioned design-and-control MDP. Here, $s_t = \{s_t^d, s_t^c\}$ represents the concatenation of the design observation $s_t^d$ and control observation $s_t^c$. Specifically, with $N$ denoting the size of the design space, $s_t^d = \{s_t^{d_1}, s_t^{d_2}, ..., s_t^{d_N}\}$, where $s_t^{d_i}$ for each module is a 9-dimensional vector consisting of its type and the types of its Moore neighborhood. $s_t^c$ represents the state information received at each time step during the interaction with the environment. The sequential input consists of local observations from all voxels, denoted as $s_t^c = \{s_t^v, s_t^g\}$, and $s_t^v = \{s_t^{v_1}, s_t^{v_2}, ..., s_t^{v_N}\}$, modular observation $s_t^{v_i}$ includes the relative position of each voxel's four corners with respect to the center of mass of the robot. $s_t^g$ represents task-related observations such as terrain information.

We utilize two feed-forward neural networks to decode shared information from the transformer-based encoder and produce the corresponding 9-dimensional design and control actions. In the training process, the determination of each module is accomplished by sampling from a categorical distribution, which is formulated based on the output logits. During the evaluation, the action corresponding to the highest logit value is selected. Additionally, The control action decoder generates the mean value $\mu$. By combining a constant standard deviation $\Sigma$, control signals can be sampled from this Gaussian distribution, and then clipped to actuate the robot. We estimate the overall value of the morphology by taking the average of the values assigned to each individual voxel.

**Learning Process** Following the convention in the literature, our method utilizes multiple CPU threads to accelerate the pre-training process. At every time step, $K$ state information of each co-design task is uniformly sampled and fed to the pleiotropic co-design policy. Once the desired number of trajectories is collected, the policy is updated using PPO [50]. After completing the pre-training phase, the resulting pre-trained co-designer is utilized to generate initial designs and control policies, which are subsequently fine-tuned for unseen tasks.

# 5 Experiments

In this section, we experimentally evaluate our proposed approach to answer the following questions: (1) Does our method, PreCo, can effectively perform brain-body pre-training and discover robot morphologies capable of executing multiple tasks? (2) How well does our method demonstrate zero-shot generalization and brain-body fine-tuning capabilities when faced with unseen co-design tasks? (3) What is the impact of the pleiotropic policy representations on the performance of PreCo?

## 5.1 Environments and Implementation

Based on the Evolution Gym platform [3], we establish a modular robot state-action space that supports brain-body pre-training as described in Section 4. Our focus lies on a fixed design space of size $5 \times 5$ which includes 9 locomotion co-design tasks with open-ended environments: **Walker-v0** (**easy**), **PlatformJumper-v0** (**hard**), **UpStepper-v0** (**medium**), **ObstacleTraverser-v0** (**medium**),

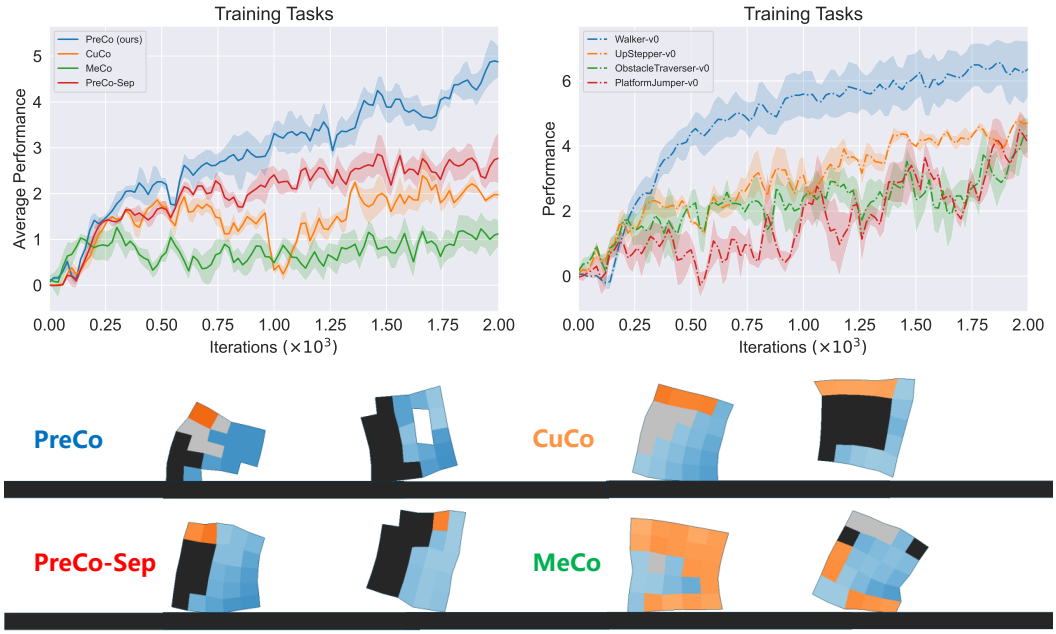


Figure 3: Learning curves and converged morphologies of brain-body pre-training. In the left figure, we demonstrate the mean and standard deviation of average task performance against the number of policy iterations for all methods. The right figure displays the individual learning curves of PreCo. The bottom figure shows two representative converged morphologies from each method.

**BridgeWalker-v0** (easy), **GapJumper-v0** (hard), **DownStepper-v0** (easy), **ObstacleTraverser-v1** (hard) and **Hurdler-v0** (hard). The difficulty levels are determined based on the performance of evolution-based co-design algorithms from the platform. For more information regarding the environmental details, please refer to Appendix A. We select the first four tasks as the pre-training tasks due to their coverage of all difficulty levels and representation of various aspects of robot locomotion (Figure 1). In terms of the remaining testing tasks, our goal is to assess whether the pre-trained co-designer can utilize the acquired brain-body connections to facilitate new co-design processes, such as traversing more uneven terrain, leaping over gaps and descending stairs.

We compare PreCo against the following baselines that are also suitable for multiple co-design tasks: (1) **PreCo-Sep**, which is based on PreCo but utilizes separate transformer-based design and control policies. This baseline allows us to investigate the effectiveness of the pleiotropic representation. (2) **CuCo** [6], a curriculum-based co-design method that consists of separate NCA-based design policy and transformer-based control policy. We set the curriculum of CuCo to be $3 \times 3 \rightarrow 5 \times 5$. (3) **MeCo**, which adopts the same network architecture as PreCo but is trained using Reptile [57], a popular meta-learning method. We use a 3-layer Transformer encoder and run all experiments with the same number of policy iterations. More implementation details can be found in Appendix B.

### 5.2 Results

**Brain-Body Pre-Training** We show the learning curves and converged robot morphologies of all training tasks for each method in Figure 3. For each method, the learning curve is reported over 7 different runs. It is evident from the figure that our method, PreCo, significantly outperforms the baselines in terms of learning speed and final performance. When comparing the robot morphologies designed by each method, we can see interesting differences. Both PreCo and PreCo-Sep are capable of discovering robot bodies with rigid legs, which are essential for maintaining balance in various locomotion tasks. PreCo, in particular, exhibits the ability to intelligently utilize empty voxels, resulting in robots with serrated and hollow structures that potentially contribute to enhanced performance. These morphological characteristics will be further analyzed in the next section. On

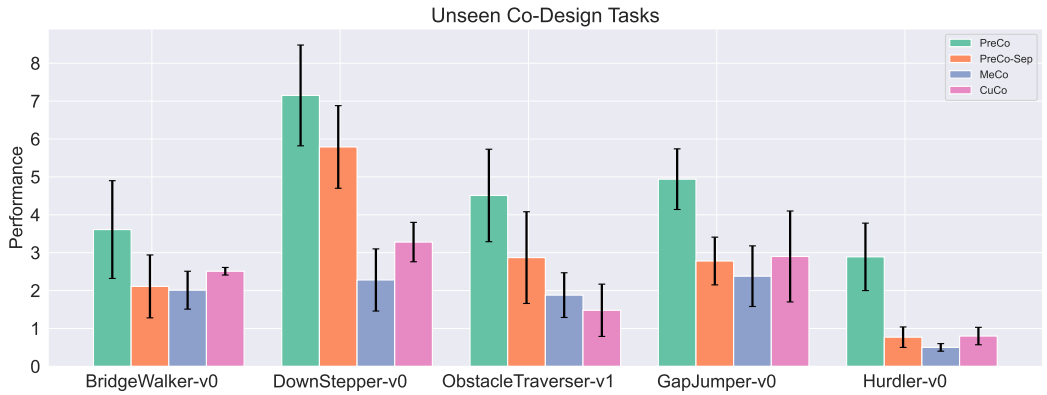


Figure 4: Evaluation of zero-shot generalization. The height of each bar represents the average zero-shot performance of a method, and error bars indicate the corresponding standard deviation.

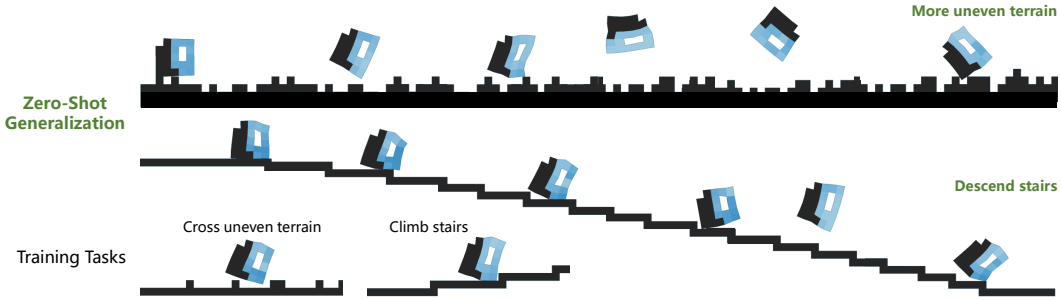


Figure 5: Visualization of PreCo's zero-shot behavior. The figure shows screenshots at consecutive time intervals of the designed robot. Compared with training tasks, We find that PreCo shows favorable generalization properties and intriguing behavior when facing new environments (video).

the other hand, CuCo produces robots with several repeated body segments. Although it benefits from curriculum learning, as evident from its learning curve, it converges to a local morphological optimum with lower performance. As for MeCo, the "meta-bodies" it discovered seem to prefer horizontal actuators, which may not be efficient in certain jumping tasks. However, our primary interest lies in evaluating its performance on the unseen co-design tasks (next section).

**Zero-Shot Generalization** Our objective in this work is to train a co-design policy capable of generating a single modular soft robot that can generalize to unseen co-design tasks. Figure 4 illustrates that PreCo consistently achieves significantly higher zero-shot performance across new environments when compared to the baseline methods. Remarkably, the robot designed by PreCo demonstrates an astonishing ability to employ the skill of somersaulting for traversing challenging terrains, relying on its environmental comprehension. Furthermore, it exhibits an understanding of the necessity to lean back to preserve stability while descending stairs, as depicted in Figure 5.

**Brain-Body Fine-Tuning** Besides the evaluation of zero-shot generalization, we also consider a more general setting that allows the co-design policy to fine-tune its parameters to adapt to new environments. Here, we limit the number of brain-body fine-tuning iterations to 300. As shown in Figure 6, PreCo exhibits intelligent behavior by retaining the beneficial serrated structure for effective stair climbing and obstacle traversal, while also making adaptive modifications to suit the new environment. For instance, it extends the length of the robot's legs on the right side to enhance ground contact in BridgeWalker and DownStepper tasks, and it develops stronger legs to facilitate jumping over gaps. Table 1 presents all results across unseen environments. The reference for the task's upper bound is determined by the final performance of the Genetic Algorithm (GA), obtained

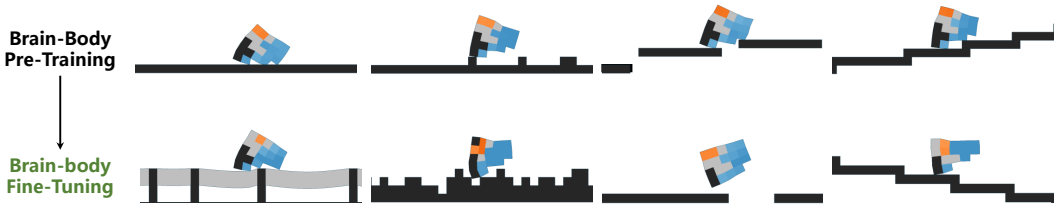


Figure 6: Visualization of brain-body fine-tuning. The pre-trained co-design policy shows the ability to swiftly adjust both the morphology and control strategy to adapt to the new co-design tasks.

Table 1: Mean and standard error of brain-body fine-tuning performance across environments and baselines, each result is reported over 7 different runs.

| Environment | PreCo | PreCo-Sep | MeCo | CuCo | Reference |
|---|---|---|---|---|---|
| BridgeWalker-v0 | $3.69 \pm 0.63$ | $3.11 \pm 0.91$ | $\mathbf{4.75 \pm 0.62}$ | $4.37 \pm 0.61$ | $6.11 \pm 0.21$ |
| Hurdler-v0 | $\mathbf{4.10 \pm 0.52}$ | $1.73 \pm 2.19$ | $1.88 \pm 1.30$ | $1.66 \pm 1.24$ | $4.51 \pm 1.12$ |
| DownStepper-v0 | $\mathbf{9.01 \pm 0.02}$ | $8.90 \pm 0.01$ | $4.58 \pm 0.74$ | $8.65 \pm 0.33$ | $9.01 \pm 0.01$ |
| GapJumper-v0 | $\mathbf{5.78 \pm 1.25}$ | $2.58 \pm 0.19$ | $2.83 \pm 0.50$ | $3.56 \pm 0.87$ | $3.23 \pm 0.35$ |
| ObstacleTraverser-v1 | $\mathbf{5.73 \pm 1.20}$ | $3.88 \pm 0.69$ | $3.03 \pm 0.50$ | $2.00 \pm 0.77$ | $4.31 \pm 1.23$ |

Table 2: Mean and standard error of final performance for each method, across 7 runs. All methods are trained from scratch with the same number of time steps.

| Environment | PreCo | PreCo-Sep | CuCo | CPPN-NEAT [3] | GA [3] |
|---|---|---|---|---|---|
| Walker-v0 | $10.47 \pm 0.01$ | $10.46 \pm 0.01$ | $10.45 \pm 0.02$ | $10.46 \pm 0.01$ | $10.46 \pm 0.03$ |
| UpStepper-v0 | $\mathbf{7.23 \pm 1.46}$ | $4.06 \pm 0.28$ | $4.48 \pm 0.74$ | $3.67 \pm 0.50$ | $5.11 \pm 0.53$ |
| PlatformJumper-v0 | $\mathbf{6.12 \pm 0.82}$ | $3.97 \pm 0.33$ | $3.11 \pm 0.50$ | $2.36 \pm 0.07$ | $2.69 \pm 0.63$ |
| ObstacleTraverser-v0 | $\mathbf{6.03 \pm 2.34}$ | $5.08 \pm 0.19$ | $4.03 \pm 0.50$ | $1.36 \pm 0.32$ | $4.60 \pm 0.43$ |

after tens of thousands of policy iterations. It is evident that PreCo consistently outperforms the baseline algorithms in the majority of environments. We also observe that PreCo fails to produce high-performance robots in BridgeWalker, whereas the MeCo achieves a better result. The separate policy representations of PreCo-Sep and CuCo result in relatively lower performance improvements.

**Importance of Pleiotropic Representations** As shown in Figure 3, despite a larger search space, a pleiotropic policy representation proves advantageous in facilitating the learning process, surpassing methods that employ separated representations. Table 2 shows that the pleiotropic representations can also help the algorithm learn from scratch, resulting in the highest mean performance.

## 6 Limitations and Conclusion

We have introduced PreCo, a co-design method that utilizes brain-body pre-training to generate modular soft robots capable of performing multiple tasks. Through the adoption of pleiotropic policy representations, which capture the inherent brain-body connections across various co-design tasks, we have observed its favorable zero-shot generalization and fine-tuning capabilities in addressing previously unseen co-design tasks.

Despite the interesting results, there are a number of areas for improvement. As shown in Table 1, our method does not perform very well in BridgeWalker. This might be because the selection of training tasks does not cover the variation of soft terrain, which may confuse the co-design policy. Exploring the selection of co-design tasks could be interesting for future research. Additionally, although the pleiotropic policy representation appears to facilitate the learning process, it is worth noting that destructive mutations of the network parameters can still occur. Further investigation into the genotype-phenotype-fitness mapping of the pleiotropic policy would be valuable.

## References


[1] H. Lipson, V. SunSpiral, J. C. Bongard, and N. Cheney. On the difficulty of co-optimizing morphology and control in evolved virtual creatures. In *IEEE Symposium on Artificial Life*, 2016.

[2] R. Pfeifer, F. Iida, and M. Lungarella. Cognition from the bottom up: on biological inspiration, body morphology, and soft materials. *Trends in Cognitive Sciences*, 18:404–413, 2014.

[3] J. Bhatia, H. Jackson, Y. Tian, J. Xu, and W. Matusik. Evolution gym: A large-scale benchmark for evolving soft robots. In *NeurIPS*, 2021.

[4] K. S. Luck, H. B. Amor, and R. Calandra. Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning. In *CoRL*, 2019.

[5] A. Gupta, S. Savarese, S. Ganguli, and L. Fei-Fei. Embodied intelligence via learning and evolution. *Nature Communications*, 12, 2021.

[6] Y. Wang, S. Wu, H. Fu, Q. Fu, T. Zhang, Y. Chang, and X. Wang. Curriculum-based co-design of morphology and control of voxel-based soft robots. In *The Eleventh International Conference on Learning Representations*, 2023.

[7] A. Zhao, J. Xu, M. Konakovic-Lukovic, J. Hughes, A. Spielberg, D. Rus, and W. Matusik. Robogrammar: graph grammar for terrain-optimized robot design. *ACM Trans. Graph.*, 39: 188:1–188:16, 2020.

[8] K. Sims. Evolving 3d morphology and behavior by competition. *Artificial Life*, 1:353–372, 1994.

[9] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *GECCO '13*, 2013.

[10] E. Medvet, A. Bartoli, F. Pigozzi, and M. Rochelli. Biodiversity in evolved voxel-based soft robots. *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021.

[11] T. Wang, Y. Zhou, S. Fidler, and J. Ba. Neural graph evolution: Towards efficient automatic robot design. *ArXiv*, abs/1906.05370, 2019.

[12] T. F. Nygaard, D. Howard, and K. Glette. Real world morphological evolution is feasible. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 2020.

[13] G. Williams. Pleiotropy, natural selection, and the evolution of senescence. *Evolution*, 11, 1957.

[14] N. Solovieff, C. Cotsapas, P. H. Lee, S. M. Purcell, and J. W. Smoller. Pleiotropy in complex traits: challenges and strategies. *Nature Reviews Genetics*, 14:483–495, 2013.

[15] D. Marzougui, M. Biondina, and F. Wyffels. A comparative analysis on genome pleiotropy for evolved soft robots. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022.

[16] J. Nordmoen, F. Veenstra, K. O. Ellefsen, and K. Glette. Quality and diversity in evolutionary modular robotics. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2109–2116, 2020.

[17] F. Veenstra, A. Faíña, S. Risi, and K. Støy. Evolution and morphogenesis of simulated modular robots: A comparison between a direct and generative encoding. In *EvoApplications*, 2017.

[18] E. Samuelsen, K. Glette, and J. Tørresen. A hox gene inspired generative approach to evolving robot morphology. In *Annual Conference on Genetic and Evolutionary Computation*, 2013.

[19] R. Wang, J. Lehman, J. Clune, and K. O. Stanley. Poet: open-ended coevolution of environments and their optimized solutions. *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019.

[20] K. Walker and H. Hauser. Evolution of morphology through sculpting in a voxel based robot. In *ALIFE*, 2021.

[21] A. Mordvintsev, E. Randazzo, E. Niklasson, and M. Levin. Growing neural cellular automata. *Distill*, 2020.

[22] S. Sudhakaran, E. Najarro, and S. Risi. Goal-guided neural cellular automata: Learning to control self-organising systems. *ArXiv*, abs/2205.06806, 2022.

[23] R. B. Palm, M. G. Duque, S. Sudhakaran, and S. Risi. Variational neural cellular automata. *ArXiv*, abs/2201.12360, 2022.

[24] N. Cheney, J. C. Bongard, V. SunSpiral, and H. Lipson. Scalable co-optimization of morphology and control in embodied machines. *Journal of The Royal Society Interface*, 15, 2018.

[25] F. H. K. dos Santos Tanaka and C. C. Aranha. Co-evolving morphology and control of soft robots using a single genome. *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1235–1242, 2022.

[26] Á. Belmonte-Baeza, J. Lee, G. Valsecchi, and M. Hutter. Meta reinforcement learning for optimal design of legged robots. *IEEE Robotics and Automation Letters*, 7:12134–12141, 2022.

[27] T.-H. Wang, P. Ma, A. E. Spielberg, Z. Xian, H. Zhang, J. B. Tenenbaum, D. Rus, and C. Gan. Softzoo: A soft robot co-design benchmark for locomotion in diverse environments. *arXiv preprint arXiv:2303.09555*, 2023.

[28] P. Ma, T. Du, J. Z. Zhang, K. Wu, A. Spielberg, R. K. Katzschmann, and W. Matusik. Diffaqua: A differentiable computational design pipeline for soft underwater swimmers with shape interpolation. *ACM Trans. Graph.*, 40:132:1–132:14, 2021.

[29] M. Bächer, E. Knoop, and C. Schumacher. Design and control of soft robots using differentiable simulation. *Current Robotics Reports*, 2(2):211–221, 2021.

[30] J. Z. Zhang, Y. Zhang, P. Ma, E. Nava, T. Du, P. Arm, W. Matusik, and R. K. Katzschmann. Sim2real for soft robotic fish via differentiable simulation. *arXiv preprint arXiv:2109.14855*, 2021.

[31] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 International conference on robotics and automation (ICRA)*, pages 6265–6271. IEEE, 2019.

[32] T. Du, J. Hughes, S. Wah, W. Matusik, and D. Rus. Underwater soft robot modeling and control with differentiable simulation. *IEEE Robotics and Automation Letters*, 6(3):4994–5001, 2021.

[33] C. B. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter. Jointly learning to construct and control agents using deep reinforcement learning. *2019 International Conference on Robotics and Automation (ICRA)*, pages 9798–9805, 2019.

[34] D. R. Ha. Reinforcement learning for improving agent design. *Artificial Life*, 25:352–365, 2019.

[35] Y. Yuan, Y. Song, Z. Luo, W. Sun, and K. M. Kitani. Transform2act: Learning a transform-and-control policy for efficient agent design. *ArXiv*, abs/2110.03659, 2022.

[36] Y. W. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. M. O. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. In *NIPS*, 2017.

[37] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *ArXiv*, abs/1802.01561, 2018.

[38] M. Hessel, H. Soyer, L. Espeholt, W. M. Czarnecki, S. Schmitt, and H. V. Hasselt. Multi-task deep reinforcement learning with popart. In *AAAI Conference on Artificial Intelligence*, 2018.

[39] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Gradient surgery for multi-task learning. *ArXiv*, abs/2001.06782, 2020.

[40] Z. Xu, K. Wu, Z. Che, J. Tang, and J. Ye. Knowledge transfer in multi-task deep reinforcement learning for continuous control. *ArXiv*, abs/2010.07494, 2020.

[41] A. Sanchez-Gonzalez, N. M. O. Heess, J. T. Springenberg, J. Merel, M. A. Riedmiller, R. Hadsell, and P. W. Battaglia. Graph networks as learnable physics engines for inference and control. *ArXiv*, abs/1806.01242, 2018.

[42] W. Huang, I. Mordatch, and D. Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *ICML*, 2020.

[43] A. Gupta, L. J. Fan, S. Ganguli, and L. Fei-Fei. Metamorph: Learning universal controllers with transformers. *ArXiv*, abs/2203.11931, 2022.

[44] B. Trabucco, M. Phielipp, and G. Berseth. Anymorph: Learning transferable polices by inferring agent morphology. In *ICML*, 2022.

[45] G. Cheng, L. Dong, W. Cai, and C. Sun. Multi-task reinforcement learning with attention-based mixture of experts. *IEEE Robotics and Automation Letters*, 8:3811–3818, 2023.

[46] R. Wang, J. Lehman, A. Rawal, J. Zhi, Y. Li, J. Clune, and K. O. Stanley. Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *ICML*, 2020.

[47] T. Anne, J. Wilkinson, and Z. Li. Meta-reinforcement learning for adaptive motor control in changing robot dynamics and environments. *ArXiv*, abs/2101.07599, 2021.

[48] T. Yu, D. Quillen, Z. He, R. C. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, 2019.

[49] K. Rakelly, A. Zhou, D. Quillen, C. Finn, and S. Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *ArXiv*, abs/1903.08254, 2019.

[50] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[51] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[52] K. He, X. Chen, S. Xie, Y. Li, P. Doll'ar, and R. B. Girshick. Masked autoencoders are scalable vision learners. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15979–15988, 2021.

[53] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021.

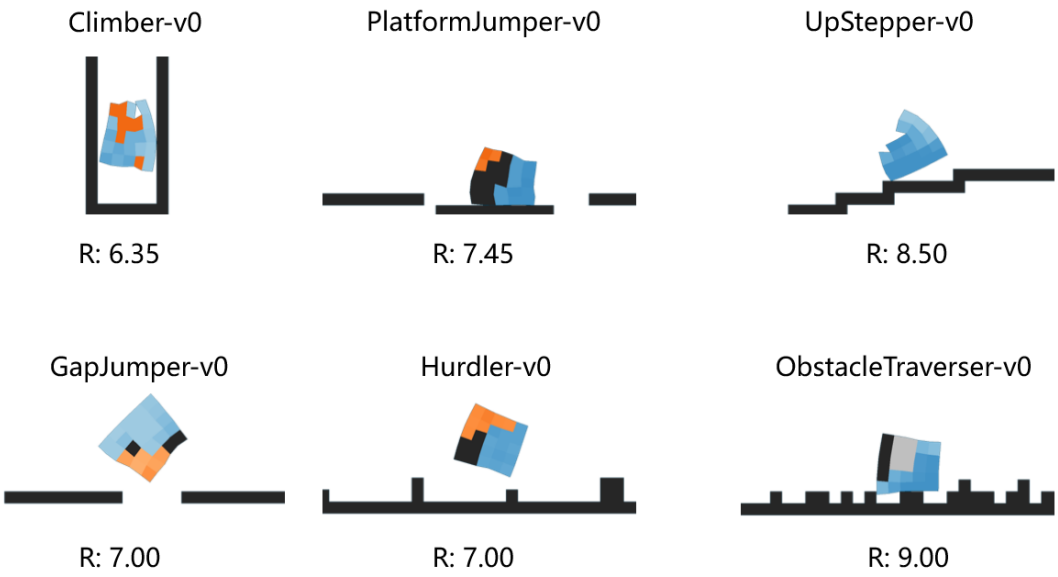


Figure 7: Visualization of some environments used in our work.

[54] C. Raffel, N. M. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683, 2019.

[55] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *ArXiv*, abs/1901.02860, 2019.

[56] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *ArXiv*, abs/2004.05150, 2020.

[57] A. Nichol and J. Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4, 2018.

[58] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

# A Environment Details

Our environment is based on platforms used in [3, 6]. In this section, we provide additional details of the environment.

**Position.** $p^o$ is a 2-dim vector that represents the position of the center of mass of an object $o$ in the simulation at time $t$. $p^o_x$ and $p^o_y$ are $x$ and $y$ components of this vector, respectively. $p^o$ is calculated by averaging the positions of all the point-masses that make up object $o$ at time $t$.

**Velocity.** $v^o$ is a 2-dim vector that represents the velocity of the center of mass of an object $o$ in the simulation at time $t$. $v^o_x$ and $v^o_y$ are $x$ and $y$ components of this vector, respectively. $v^o$ is calculated by averaging the velocities of all the point-masses that make up object $o$ at time $t$.

**Orientation.** $\theta^o$ is a 1-dim vector that represents the orientation of an object $o$ in the simulation at time $t$. Let $p_i$ be the position of point mass $i$ of object $o$. $\theta^o$ is computed by averaging over all $i$ the angle between the vector $p_i - p^o$ at time $t$ and time 0. This average is a weighted average weighted by $||p_i - p^o||$ at time 0.

**Other observations.** $h_b^o(d)$ is a vector of length $(2d+1)$ that describes elevation information around the robot below its center of mass. More specifically, for some integer $x \leq d$, the corresponding entry in vector $h_b^o(d)$ will be the highest point of the terrain which is less than $p_y^o$ between a range of $[x, x+1]$ voxels from $p_x^o$ in the $x$-direction.

### A.1 Walker-v0

In this task, the robot is rewarded by walking as far as possible on flat terrain. The target design space is $7 \times 7$. The task-specific observation is $v^{robot}$, and the reward $R$ is:

$$R = \Delta p_x^{robot} \tag{2}$$

which rewards the robot for moving in the positive $x$-direction. The robot receives a reward of $1$ for reaching the end of the terrain. The episode duration reaches a $500$ time steps.

### A.2 Upstepper-v0

In this task, the robot climbs up stairs of varying lengths. The task-specific observation is formed by concatenating vectors $\{v^{robot}, \theta^{robot}, h_b^{robot}(5)\}$, and the reward $R$ is:

$$R = \Delta p_x^{robot} \tag{3}$$

which rewards the robot for moving in the positive $x$-direction. The robot also receives a one-time reward of $2$ for reaching the end of the terrain. The episode duration reaches a $600$ time steps.

### A.3 Downstepper-v0

In this task, the robot climbs down stairs of varying lengths. The task-specific observation is formed by concatenating vectors $\{v^{robot}, \theta^{robot}, h_b^{robot}(5)\}$, and the reward $R$ is:

$$R = \Delta p_x^{robot} \tag{4}$$

which rewards the robot for moving in the positive $x$-direction. The robot also receives a one-time reward of $2$ for reaching the end of the terrain, and a one-time penalty of $-3$ for rotating more than $90$ degrees from its originally orientation in either direction. The episode duration reaches a $500$ time steps.

### A.4 ObstacleTraverser-v0

In this task, the robot walks across terrain that gets increasingly more bumpy. The task-specific observation is formed by concatenating vectors $\{v^{robot}, \theta^{robot}, h_b^{robot}(5)\}$, and the reward $R$ is:

$$R = \Delta p_x^{robot} \tag{5}$$

which rewards the robot for moving in the positive $x$-direction. The robot also receives a one-time reward of $2$ for reaching the end of the terrain, and a one-time penalty of $-3$ for rotating more than $90$ degrees from its originally orientation in either direction. The episode duration reaches a $1000$ time steps.

### A.5 ObstacleTraverser-v1

In this task, the robot walks through very bumpy terrain. The task-specific observation is formed by concatenating vectors $\{v^{robot}, \theta^{robot}, h_b^{robot}(5)\}$, and the reward $R$ is:

$$R = \Delta p_x^{robot} \tag{6}$$

which rewards the robot for moving in the positive $x$-direction. The robot also receives a one-time reward of $2$ for reaching the end of the terrain. The episode duration reaches a $1000$ time steps.

### A.6 Hurdler-v0

In this task, the robot walks across terrain with tall obstacles. The task-specific observation is formed by concatenating vectors $\{v^{robot}, \theta^{robot}, h_b^{robot}(5)\}$, and the reward $R$ is:

$$R = \Delta p_x^{robot} \tag{7}$$

which rewards the robot for moving in the positive $x$-direction. The robot also receives a one-time penalty of $-3$ for rotating more than $90$ degrees from its originally orientation in either direction. The episode duration reaches a $1000$ time steps.

### A.7 GapJumper-v0

In this task, the robot traverses a series of spaced-out floating platforms all at the same height. The task-specific observation is formed by concatenating vectors $\{v^{robot}, \theta^{robot}, h_b^{robot}(5)\}$, and the reward $R$ is:

$$R = \Delta p_x^{robot} \tag{8}$$

which rewards the robot for moving in the positive $x$-direction. The robot also receives a one-time penalty of $-3$ for falling off the platforms. The episode duration reaches a $1000$ time steps.

### A.8 PlatformJumper-v0

In this task, the robot traverses a series of floating platforms at different heights. The target design space is $5 \times 5$. The task-specific observation is formed by concatenating vectors $\{v^{robot}, \theta^{robot}, h_b^{robot}(5)\}$, and the reward $R$ is:

$$R = \Delta p_x^{robot} \tag{9}$$

which rewards the robot for moving in the positive $x$-direction, The robot also receives a one-time penalty of $-3$ for rotating more than $90$ degrees from its originally orientation in either direction or for falling off the platforms (after which the environment resets). The episode duration reaches a $1000$ time steps.

## B Implementation Details

We use PyTorch [58] to implement all modules used in our work. All hyperparameters for PreCo are listed in Table 3. For baseline algorithms, we use the official implementations of GA, CPPN-NEAT from Evolution Gym [3] and we employ a population of 12 agents for all population-based methods. Furthermore, we use the official implementation of CuCo from [6] and Reptile from [57].

Table 3: Hyperparameters of PreCo.

| | **Hyperparameter** | **Value** |
|---|---|---|
| PPO | GAE | True |
| | GAE $\lambda$ | 0.95 |
| | Learning rate | $2.5 \cdot 10^{-4}$ |
| | Linear learning rate decay | True |
| | Clip parameter | $0.1$ |
| | Value loss coefficient | $0.5$ |
| | Entropy coefficient | $0.01$ |
| | Time steps per rollout | $5120$ |
| | Optimizer | Adam |
| | Evaluation interval | $10$ |
| | Discount factor $\gamma$ | $0.99$ |
| | Clipped value function | True |
| | Observation normalization | True |
| | Observation clipping | $[-10, 10]$ |
| | Reward normalization | True |
| | Reward clipping | $[-10, 10]$ |
| | Policy epochs | $8$ |
| Transformer | Neighborhood | Moore |
| | Design steps | 1 |
| | Number of layers | 3 |
| | Number of attention heads | 1 |
| | Embedding dimension | 128 |
| | Feedforward dimension | 256 |
| | Non linearity function | ReLU |
| | Dropout | 0.0 |