GENESIS: ADVANCING TOWARDS EFFICIENT EMBODIMENT CO-DESIGN

Anonymous authors
Paper under double-blind review

ABSTRACT

Embodiment co-design aims to optimize a robot's morphology and control simultaneously. Previous research has demonstrated its potential for generating environment-adaptive robots. However, the problem is inherently combinatorial and the morphology is changeable and agnostic in its vast search space, optimization efficiency remains complex and challenging to address. We prove that the inefficient morphology representation and unbalanced reward signals between the design and control stages are key obstacles against efficiency. In order to advance towards efficient embodiment co-design to unlock its full potential, we propose *Genesis*, which utilizes (1) a novel topology-aware self-attention architecture, enabling efficient morphology representation while enjoying lightweight model sizes; (2) a temporal credit assignment mechanism for co-design that ensures balanced reward signals for optimization. With our simple-yet-efficient methods, Genesis achieves average **60.03%** performance improvement against the strongest baselines. We provide codes and more results on the website: `https://genesisorigin.github.io`.

1 INTRODUCTION

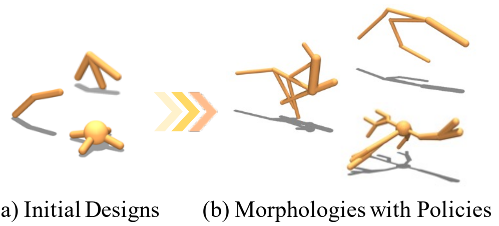

(a) Initial Designs (b) Morphologies with Policies

Figure 1: Embodied Agents generated by Genesis

Species in nature are blessed with millions of years to evolve for remarkable capacities to adapt to the environment (Pfeifer & Scheier, 2001; Vargas et al., 2014). Time has gifted them with perfect physical bodies for movement and navigation, powerful processors for centralized information processing, and effective actuators for rapid interaction with their surroundings. Inspired by this observation, *embodiment co-design* (Sims, 1994; Ha, 2019; Yuan et al., 2021; Wang et al., 2023), where a robot's morphology and control are optimized simultaneously, has gained increasing attention, and demonstrates significant potential in various downstream fields, such as automated robot design and bio-inspired robot generation (Kriegman et al., 2020; Nakajima et al., 2018; Judd et al., 2019; Pan et al., 2021; Whitman et al., 2023). However, this task encounters extreme difficulties: (1) the morphology search space is quite vast and combinatorial, with each morphology corresponding to unique action and state spaces; (2) evaluating each candidate design requires an expensive roll-out to find its optimal control policy, which is almost unfeasible for the expensive computation.

Previous methods (Sims, 1994; Wang et al., 2018b; Zhao et al., 2020; Gupta et al., 2021b) typically utilize evolutionary search (ES) for embodiment co-design. Specifically, they maintain a fixed-size population of agents with different morphology designs for random mutations, and only preserve the top-performing agents' children for further optimization. These methods inevitably result in inefficient sampling within a vast design space and prevent the sharing of valuable experiences and skills across different morphologies.

Substantial efforts have been made to address the shortcomings of ES-based methods. A natural approach is introducing more human morphology priors, such as symmetry (Gupta et al., 2021b; Dong et al., 2023), into the design process to reduce the search space. However, these methods can

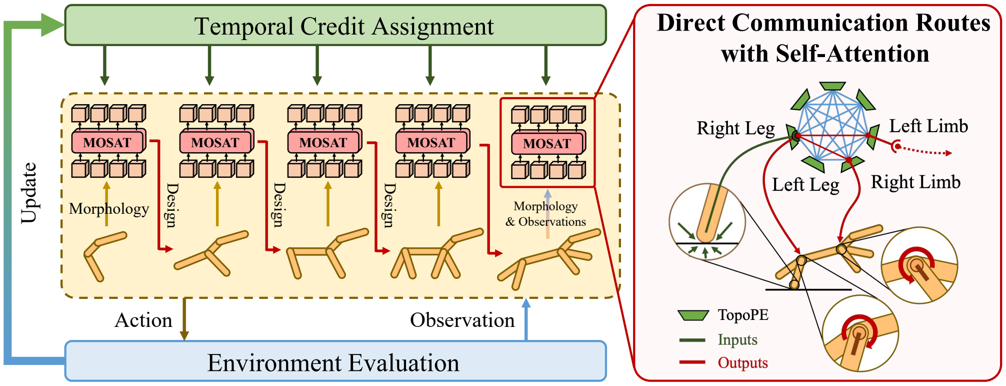


Figure 2: Overview of our Genesis, which leverages an RL-based framework for joint evolving of morphology and control policy, and an attention-based network equipped with Topology Position Encoding (TopoPE) for centralized message processing.

only offer a constant factor reduction in search space and significantly compromise functionality in specific scenarios. Meanwhile, a class of methods based on Graph Neural Networks (GNNs) has been proposed (Wang et al., 2018b; Yuan et al., 2021; Huang et al., 2020) to facilitate weight sharing across different morphologies. However, these GNN-based methods face inefficiencies in message transmission between joints due to the limitation of aggregating multi-hop information (Kurin et al., 2020), reducing final performance.

Notably, Yuan et al. (2021) propose to optimize a robot's morphology and control concurrently through reinforcement learning (RL), where an agent first applies a sequence of transform actions to modify its morphology without environment interaction, then it uses this morphology to interact with the environment and receive rewards. The introduction of reinforcement learning effectively facilitates learning, however, it also brings the problem of delayed rewards: compared with the second phase (control phase), the first phase (design phase) is almost zero-reward guided, and this leads to an unbalanced reward allocation.

Driven by this analysis, we present Genesis, an end-to-end reinforcement learning framework that optimizes message and reward delivery during co-design. Genesis utilizes (1) a morphology self-attention architecture to achieve direct message delivery during optimization, with our proposed topology-aware positional encoding to achieve message localization and morphological knowledge sharing; (2) a temporal credit assignment mechanism that ensures an agent receives balanced reward signals in both the morphology design and control phases.

To summarize, our contributions are as follows:

- We propose Genesis, an end-to-end reinforcement learning framework for efficient embodiment co-design.
- We design a Morphology Self-Attention architecture (MoSAT) to provide direct message delivery, featuring our proposed Topological Position Encoding (TopoPE) for efficient morphology representation.
- We propose a temporal credit assignment mechanism that ensures balanced reward signals in both the morphology design and control phases, greatly facilitating efficient learning.

Comprehensive experiments across various tasks demonstrate Genesis's superiority against previous methods in terms of both convergence speed and performance. Genesis achieves an average performance improvement of **60.03%** against the strongest baselines.

2 RELATED WORK

Universal Morphology Control Embodiment co-design requires controlling robots with changeable morphologies and adapting to their incompatible action and state spaces. Universal Morphology Control (UMC), which employs a shared network to control each actuator separately, presents a promising solution to this problem. To better perceive the topological structures of various morphologies, some methods Pathak et al. (2019); Wang et al. (2018a); Huang et al. (2020) employed Graph Neural Networks (GNNs) to enable communication between neighboring actuators. Recent works also use Transformers (Vaswani et al., 2017) to overcome the limitations of multi-hop information aggregation brought by GNNs (Kurin et al., 2020; Hong et al., 2021; Gupta et al., 2021a; Dong et al., 2022). While most of these methods focus on parametric variations of a limited number (*e.g.* 2-3) of predefined morphologies, our approach addresses a much more challenging task, which not only requires controlling various morphology-agnostic robots within a vast design space but also needs to concurrently learn for the best morphology. Furthermore, most previous works do not fully leverage morphology information or only consider its simple form, and even the usefulness of such information remains controversial (Kurin et al., 2020; Hong et al., 2021; Gupta et al., 2021a; Xiong et al., 2023). In this work, we prove that morphology information is quite crucial, and the *correctness* of morphology representation significantly influences performance. Consequently, we introduce a novel positional encoding technique called TopoPE that not only facilitates message localization within the body but also enhances knowledge sharing among similar morphologies.

Embodiment Co-design As for embodied artificial agents, control policy has been well studied in the robotics community (Lillicrap et al., 2015; Schulman et al., 2015a; Haarnoja et al., 2018; Schulman et al., 2017; Lowrey et al., 2018), while another critical component, the physical form of the embodiment, is currently attracting more and more attention (Kriegman et al., 2020; Bhatia et al., 2021; Xu et al., 2021; Huang et al., 2024). Embodiment co-design aims to optimize a robot's morphology and control simultaneously and is considered a promising way to stimulate the embodied intelligence embedded in morphology. Previous methods (Sims, 1994; Wang et al., 2018b; Gupta et al., 2021b) typically utilize evolutionary search (ES) to learn directly within the vast design space, which unavoidably brings inefficient sampling and expensive computation. A line of works (Wang et al., 2018b; Gupta et al., 2021a) introduces more human morphology priors, such as symmetry, to reduce the search space. Wang *et al*. Wang et al. (2018b) utilize GNNs to facilitate weight sharing across morphologies. Yuan *et al*. Yuan et al. (2021) proposes to jointly optimize a robot's morphology and control policy via reinforcement learning. This paper focuses on the RL-based approach for joint optimization for both morphology and control. We aim to establish a comprehensive framework for embodiment co-design, systematically addressing the issues of message and reward delays during the training process.

3 PRELIMINARIES

Morphology Representation. The morphology of an agent can be formally defined as an undirected graph $\mathcal{G} = (V, E, A^v, A^e)$, where each node $v \in V$ represents a limb of the robot, and each edge $e = (v_i, v_j) \in E$ represents a joint connecting two limbs. A^v and A^e are two mapping functions that map the limb node v to its physical attributes $A^v : V \to \Lambda^v$, and map the edge $e = (v_i, v_j)$ to its joint attributes $A^e : E \to \Lambda^e$, respectively. Here $\Lambda^v = \{\Lambda^{v_i}\}$ is the limb attribute space, consisting of attributes Λ^{v_i} like limb lengths, sizes and materials, and $\Lambda^e = \{\Lambda^{e_i}\}$ is the joint attribute space consisting of attributes Λ^{e_i} like rotation ranges and maximum motor torques. Consequently, the design space D is defined on all valid robot morphologies $\mathcal{G} \in \mathcal{D}$.

Co-Design Optimization. The fitness F of an agent represents its performance in a specific environment, and is typically evaluated by rewards. In traditional control problems with a fixed morphology $\mathcal{G}_0$, we aim to optimize its control policy π towards the optimal $\pi^* = \arg\max_\pi F(\pi, \mathcal{G}_0)$ for maximum fitness. In co-design problems, we not only optimize the control policy but also the morphology design simultaneously. This co-design process is formulated as a bi-level optimization problem:

$$\begin{aligned} \mathcal{G}^* &= \arg\max_{\mathcal{G}} F(\pi^*_{\mathcal{G}}, \mathcal{G}) \\ \text{s.t. } \pi^*_{\mathcal{G}} &= \arg\max_{\pi_{\mathcal{G}}} F(\pi_{\mathcal{G}}, \mathcal{G}), \end{aligned} \tag{3.1}$$

where the inner loop defines the optimal control policy of a given morphology, and the outer loop defines the optimal morphology using its optimal policy. Previous works typically use evolutionary algorithms (Sims, 1994; Wang et al., 2018b; Gupta et al., 2021b) to solve this problem. In this work, Genesis leverages an RL-based framework, and jointly optimizes both loops:

$$\pi^*(\cdot|\mathcal{G}^*), \mathcal{G}^* = \arg\max_{\pi(\cdot|\mathcal{G}), \mathcal{G}} F(\pi(\cdot|\mathcal{G}), \mathcal{G}), \tag{3.2}$$

using the universal control policy $\pi(\cdot|\mathcal{G})$, which not only benefits co-design for both morphology and control, but also promotes knowledge sharing among agents, and greatly improves learning efficiency.

Reinforcement Learning. We formally define the problem formulation of Morphology-Conditioned Reinforcement Learning for embodiment co-design. We consider the augmented Markov Decision Process (MDP), which can be described by a 6-element tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{D}, \Phi)$. Φ is a flag to distinguish design and control stages. $\mathcal{S}$ denotes the state space. $\mathcal{A}(\Phi)$ represents the action space, where $a \in \mathcal{A}(\Phi = \text{Design})$ changes the morphology of the agent, and $\mathcal{A}(\Phi = \text{Control})$ defines the action space for motion control. $\mathcal{T} : \mathcal{S} \times \mathcal{A}(\Phi) \times \mathcal{S} \to [0, 1]$ represents the environmental transitioning probability from one state s_t to another s_{t+1}, given an action a_t. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the state-action reward, and the fitness function F is defined as the episodic return $\sum_{t=1}^{T} r_t(s_t, a_t)$ based on rewards. As defined above, $\mathcal{D}$ represents the morphology design space, and our goal is to find some co-design policy $\pi : \mathcal{S} \times \mathcal{D} \to \mathcal{A}$ that can maximize the environmental fitness F.

4 METHOD

Embodiment co-design involves optimizing an agent's morphology and corresponding control policy. The co-design procedure can be divided into two consecutive stages: (1) **Design Stage**, where an agent starts with an initial morphology $\mathcal{G}_0$ and iteratively transforms it towards the final design $\mathcal{G}_{done}$ through a series of morphology transforming actions via a design policy π^D; and (2) **Control Stage**, where the agent interacts with the environment with its corresponding control policy π^C.

During the co-design process, we identified three key **factors** that impede optimization efficiency: (1) Intra-agent message signal transmission decay (Kurin et al., 2020). Coordinated control requires low-latency transmission and efficient message processing as sensors and actuators are distributed throughout the agent's body. (2) Inter-agent morphological structure knowledge transfer delay. The co-design process leads to diverse morphological structures, and existing methods struggle to efficiently share knowledge between similar agents. (3) Agent-environment reward delay. Sparse rewards during the Design Stage result in the agent's delayed perception of environmental fitness. In this study, we systematically analyze and address the above issue throughout the co-design process.

In this section, we consistently considers the situation at timestep t to erase the subscript t for clear formulation.

4.1 ATTENTION-BASED CO-DESIGN NETWORK

Genesis further divides the **Design Stage** into two sub-stages: ***Topology Design Stage*** and ***Attribute Design Stage***, which transforms the topology (V_0, E_0) and the corresponding attributes (A_0^v, A_0^e) of the agent's morphology, respectively. Consequently, the design policy π^D is also divided into two sub-policies $\pi^D = (\pi^{topo}, \pi^{attr})$ for according action control.

During the ***Topology Design Stage***, the agent can modify the topology through three basic actions: (1) `Addition` $add(v)$: add a new child limb v_{new} to v, along with a new joint $e_{new} = (v, v_{new})$ connecting them. (2) `Deletion` $del(v)$: delete the limb v and the joint to its parent $e = (v_p, v)$ if v is a leaf node. (3) `NoChange` $pass(v)$: take no changes for node v. This stage will last up to N^{topo} steps. The agent's policy π^{topo} is conditioned on the current topology (V, E) of timestep t, denoting as the product of action distributions π_v^{topo} from all limbs [1]:

$$\boldsymbol{a}^{topo} \sim \pi^{topo}(\boldsymbol{a}^{topo}|\mathcal{G}) \triangleq \prod_{v \in V} \pi_v^{topo}(a_v^{topo}|p_v, \mathcal{G}) \tag{4.1}$$

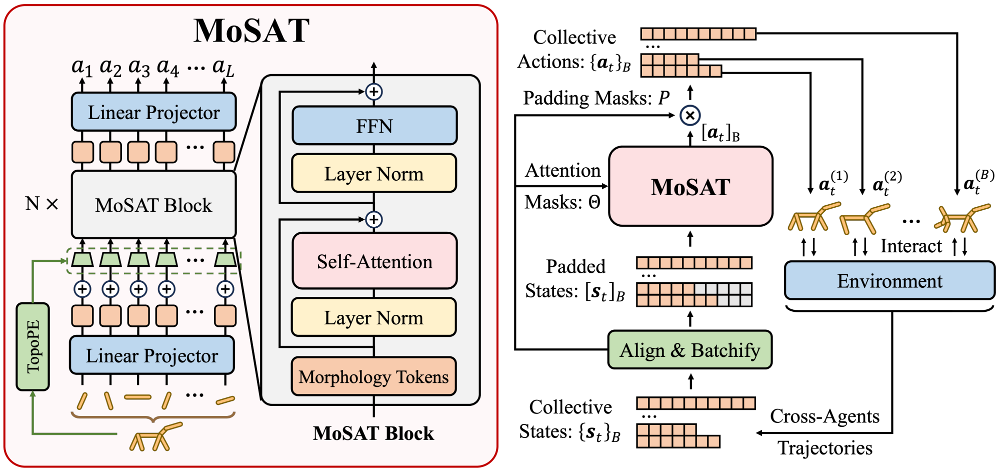
where p_v represents the topology position of node v.

In the ***Attribute Design Stage***, the agent further generates limb and joint attributes based on the given topology $\mathcal{G}_{done} = (V_{done}, E_{done})$. The agent's attribution policy π^{attr} can be formulated as:

$$\boldsymbol{a}^{attr} \sim \pi^{attr}(\boldsymbol{a}^{attr}|\mathcal{G}) \triangleq \prod_{v \in V} \pi_v^{attr}(a_v^{attr}|p_v, \mathcal{G}) \tag{4.2}$$

---

[1] We use the limb-level action distribution, where each limb corresponds to its own action distribution, and the entire agent's action distribution is composed of all limbs' distributions. This effectively resolves the incompatibility of state and action spaces across the changeable topological morphologies.



(a) MoSAT Architecture (b) Batch Processing for MoSAT

Figure 3: **The Morphology Self-Attention (MoSAT) architecture**. (a) The sensor observations from different limbs are projected to hidden tokens for *centralized* processing with several MoSAT blocks and generate *separate* actions. (b) The MoSAT network processes different morphologies in a batch manner and learns a universal control policy $\pi(\cdot|\mathcal{G})$, thus improving training efficiency.

Finally, in the **Control Stage**, the agent uses the morphology generated in the **Design Stage** to interact with the environment using the control policy π^C.

$$\boldsymbol{a}^{ctrl} \sim \pi^{ctrl}(\boldsymbol{a}^{ctrl}|\boldsymbol{s}, \mathcal{G}_{done}) \triangleq \prod_{v \in V} \pi_v^{ctrl}(a_v^{ctrl}|\boldsymbol{s}, p_v, \mathcal{G}_{done}), \tag{4.3}$$

where $\boldsymbol{s} = \{s_v\}$ denotes the sensor states of every limp, including forces, positions, velocities, *etc*. We use a_v^{ctrl} to represent the torque of the joint connecting node v with its parent v_p.

The policy network must adapt to different morphologies during training to during the co-design process. This presents two main challenges: 1) This adaptation is necessary within a single agent to accommodate the growing body and across different agents to achieve unified control. 2) Although sensors and actuators are distributed throughout different parts of the agent, we demand coordinated control over the entire body. Inspired by the centralized signal processing of mammals in real-world nature, generating remarkable intelligence, we propose the **Mo**rphology **S**elf-**A**tten**T**ion architecture (**MoSAT**) for efficient, centralized message processing. Figure 3 (a) provide an overview of MoSAT.

Latent Projection. We encode information from each limb's sensor to enhance network processing capabilities and map it into a latent space as *message tokens*. Specifically, limb sensor states s_v are first processed through a parameter-shared linear mapping layer $\phi_h(\cdot)$:

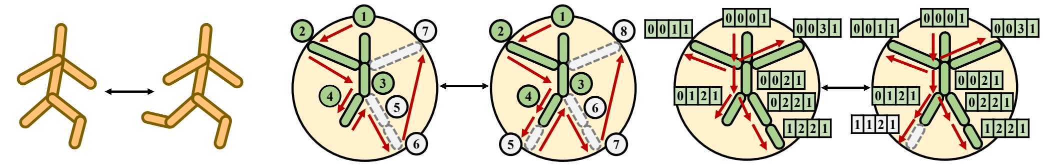
$$\mathbf{m} = \phi_h(\mathbf{s}) + \mathbf{E}_{pos}(V, E) \quad \mathbf{s} \in \mathbb{R}^{L \times d}, \mathbf{m} \in \mathbb{R}^{L \times D} \tag{4.4}$$

where d is the input state dimension and D is the hidden dimension. We employ a novel position encoding, called TopoPE for morphology representation to localize message sources, which will be further discussed later in Section 4.2. The position encodings $\mathbf{e}_v$ are added to message tokens m_v to get position-embedded message tokens $\hat{m}_v$.

Centralized Processing. As illustrated in Figure 2, we aim for efficient message interaction. Unlike previous GNN-based methods (Wang et al., 2018b; Yuan et al., 2021), where information must pass through several intermediaries before reaching the target location, our model utilizes the scaled dot-product self-attention $\text{Attention}(\cdot)$ for point-to-point, centralized processing. Specifically, each message $\mathbf{m}$ use q_{v_i} to query the key of another message k_{v_j} weighting its value v_{v_i}:

$$\text{Attention}(\mathbf{m}) = \text{SoftMax}(\frac{QK^T}{\sqrt{d_k}})V \quad \text{where } Q = \mathbf{m}W_Q, K = \mathbf{m}W_K, V = \mathbf{m}W_V, \tag{4.5}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{D \times D}$ are learnable matrices. For MoSAT block design, we adopt Pre-LN (Xiong et al., 2020) for layer normalization (LN) and add residual connections (He et al., 2016;



(a) Morphology Changes (b) Traversal PE (c) **TopoPE (Ours)**

Figure 4: The motivation of our proposed topology-aware position encoding TopoPE. (a) During the co-design procedure, the agent's morphology keeps changing. (b) A typical traversal-based PE in previous works resulted in inconsistency across mythologies. (c) TopoPE can better adapt to similar morphology structures using a reasonably align-able manner.

Dosovitskiy et al., 2020) to facilitate learning. This procedure can be formulated as:

$$\mathbf{m}'_{l+1} = \text{Attention}(\text{LN}(\mathbf{m}_l)) + \mathbf{m}_l \quad l = 0...N-1 \tag{4.6}$$

$$\mathbf{m}_{l+1} = \text{MLP}(\text{LN}(\mathbf{m}'_l), r) + \mathbf{m}'_l \quad l = 0...N-1 \tag{4.7}$$

where N is the stacked block number, and the $\text{MLP}(\cdot, r)$ is an MLP layer with one hidden layer, which first upscale the token dimension to $r \times D$, then project it back to D.

Forwarding. In the end, we need to output actions for each actuator. We decode the attended messages using a linear projector $\phi_a(\cdot)$ to generate the action logits for each actuator:

$$\pi(\mathbf{a}|\mathbf{s}) = \begin{cases} \text{SoftMax}(\phi_a(\mathbf{m}_N)), & \text{Discrete Action Space} \\ \mathcal{N}(\mathbf{a}; \phi_a(\mathbf{m}_N), \Sigma), & \text{Continuous Action Space.} \end{cases} \tag{4.8}$$

The above process has equipped MoSAT with the capability to handle various morphologies. To maximize training efficiency, we further offer MoSAT the ability to process multiple morphologies in a batch mode. As shown in Figure 3 (b), for a batch of state inputs $\{\boldsymbol{s}_t\}_B$, we first pad them to equal length $[\boldsymbol{s}_t]_B \in \mathbb{R}^{B \times L_m \times d}$, where L_m is the max limb number of morphologies within this batch, and generate a padding matrix $P \in \mathbb{R}^{B \times L_m}$, where $P_{ij} = 1$ for $j \leq L_i$ and $P_{ij} = 0$ for $j > L_i$. To keep the messaging logic exactly equivalent to the regular mode, we can eliminate the influence of padding by modifying the attention operation with an attention mask $\Theta \in \mathbb{R}^{B \times L_m \times L_m}$:

$$\text{Attention}([\mathbf{m}_t]_B) = \text{SoftMax}(\frac{QK^T}{\sqrt{d_k}} + \Theta)V, \tag{4.9}$$

where $\Theta_{ijk} = \log(P_{ik} + \epsilon)$. Finally, we remove the batch padding and re-allocate actions to joints of different agents via: $\{\boldsymbol{a}\}_B = [\boldsymbol{a}]_B \odot P$, where $\odot$ represents the bool-selection operation according to the padding matrix P.

4.2 TOPOLOGY-AWARE POSITION ENCODING FOR MORPHOLOGY REPRESENTATION

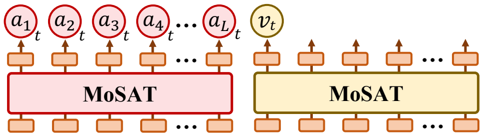
So far, we've achieved centralized message processing by performing self-attention in latent space. The vanilla attention operation treats each token equally, neglecting morphology information. However, it is crucial to inject positional information for co-design decision-making for: (1) Similar information from different morphology positions has varying meanings, and message source localization is significant; (2) Similar morphology structures may share similar local control policies, and positional information facilitates knowledge alignment and sharing across different agents. To better capture the differences between morphological structures and share structural knowledge among similar morphologies, we propose **Topology Position Encoding** (**TopoPE**), a topology-aware position encoding mechanism to handle the above two issues efficiently.

As demonstrated in Figure 4, the traversal-based limb indexing method (Hong et al., 2021; Gupta et al., 2021a; Xiong et al., 2023) is less stable, and slight morphological changes can cause global indexing offsets. To mitigate the effect of offsets due to morphological changes, TopoPE uses a hash-map $\mathcal{H}(\cdot)$ for position encoding, which maps the path between the root limb v_{root} and the current limb v_i to a unique embedding $\mathbf{e}_{v_i}$:

$$\begin{aligned} \mathbf{e}_{v_i} &= \mathcal{H}([v_i \mapsto v_{root}]) \\ \text{where } [v_i \mapsto v_{root}] &= [(v_i, p(v_i)), (p(v_i), p^2(v_i)), ..., (p^{l-1}(v_i), v_{root})], \end{aligned} \tag{4.10}$$

where $p^n(v)$ is the n-th ancestor of v. Practically, if v is the k-th child of its parent $p(v)$, the edge $(v, p(v))$ is denoted by the integer k, allowing the path index to be represented as a sequence of integers. Experiments demonstrate that TopoPE effectively adapts growing morphologies, facilitating knowledge alignment and sharing across agents, which leads to a better performance.

4.3 CO-DESIGN OPTIMIZATION WITH TEMPORAL CREDIT ASSIGNMENT



(a) Policy Network (b) Value Network

Figure 5: Genesis leverages an actor-critic paradigm for policy optimization. The Policy and value networks are both based on MoSAT.

To achieve efficient reward-driven co-design, Genesis leverages an actor-critic paradigm based on reinforcement learning, which trains a value function $V_\theta(s_t)$ and a policy function $\pi_\theta(a_t|s_t)$ and updates them using collected trajectories. We employ the Proximal Policy Optimization (PPO) (Schulman et al., 2017) to optimize the policy π_θ in the actor-critic framework. PPO uses the advantage function $\hat{A}_t(a_t, s_t)$ to define how better an action a_t is for current state s_t, and optimizes the following surrogate objective function as:

$$\mathcal{L}^{policy} = -\min\left\{\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t,\ \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right\} \tag{4.11}$$

For the co-design process, the Vanilla PPO leads to limited performance. We find two main reasons for this problem: 1) in the co-design process, only the **Control Stage** can receive direct environmental rewards, leading to an unbalanced reward distribution; 2) a body-modifying action taken in the **Design Stage** will equally affect every future interaction timestep, while a motion-control action in the **Control Stage** has decreasing impact on the future. To improve the training performance of Genesis, we introduce our modified Generalized Advantage Estimation (GAE) (Schulman et al., 2015b) for enhanced temporal credit assignment for embodiment co-design:

$$\begin{aligned} \hat{A}_t &= \begin{cases} \delta_t + \gamma\lambda\hat{A}_{t+1} \cdot (1 - \mathbb{T}_t \vee \mathbb{C}_t), & \text{for Control Stage} \\ U_t - V_\theta(s_t), & \text{for Design Stage} \end{cases} \\ \text{where} \quad \delta_t &= r_t + \gamma V_\theta(s_{t+1}) \cdot (1 - \mathbb{T}_t) - V_\theta(s_t) \\ U_t &= r_t + U_{t+1} \cdot (1 - \mathbb{T}_t \vee \mathbb{C}_t) \end{aligned} \tag{4.12}$$

where γ is the discounting factor, λ is the exponentially weighted for GAE and $\mathbb{T}_t, \mathbb{C}_t$ are two environment flags denoting environment termination and truncation, respectively. The value loss function $\mathcal{L}^{value}$ is defined as:

$$\mathcal{L}^{value} = \left(V_\theta(s_t) - \hat{R}_t\right)^2, \quad \text{where } \hat{R}_t = \text{sg}\left[V_\theta(s_t) + \hat{A}_t\right], \tag{4.13}$$

where sg$[\cdot]$ stands for the stop-gradient operator.

As shown in Figure 5, both policy and value networks use MoSAT as the backbone. Inspired by BERT (Devlin et al., 2018), we use the token output of every limb to generate the action policy in Equation (4.8) for each actuator (Figure 5 (a)), and use the token output of the root limb for value prediction of the entire body at timestep t (Figure 5 (b)). Each co-design stage has a separate value network to avoid potential conflicts in gradient descent (Yu et al., 2020; Liu et al., 2021) due to different credit assignment strategies.

5 EXPERIMENTAL EVALUATIONS

Our experiments aim to validate our primary hypothesis: that efficient message and reward delivery can effectively overcome bottlenecks in the co-design process, leading to embodied agents that can better adapt to the environment. Additional visualization results are presented in Appendix A.7. Visit our project website for videos and more visualization results [2].

---

[2] `https://genesisorigin.github.io`

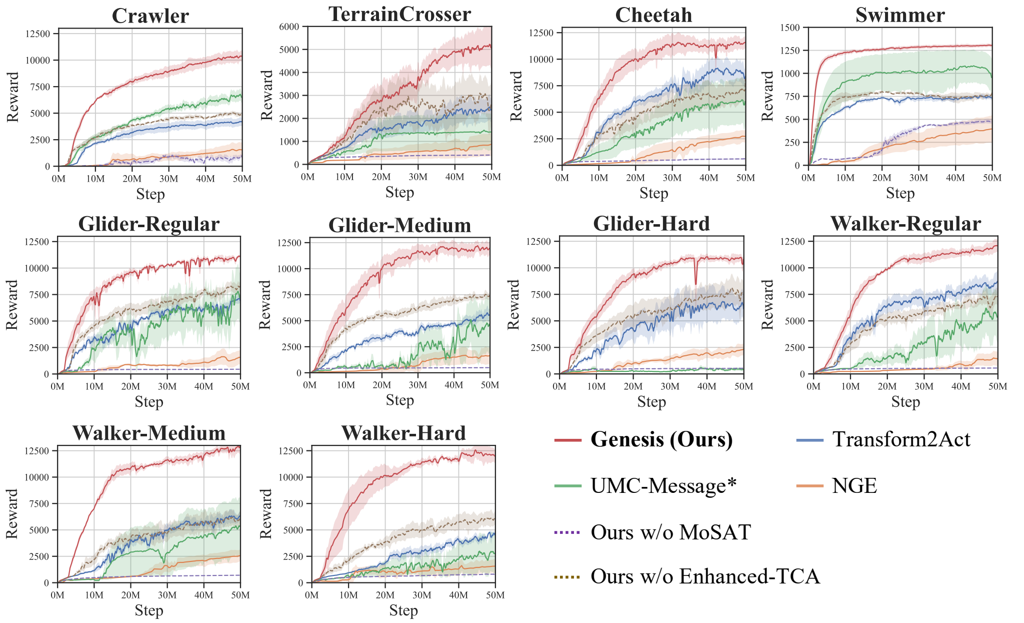


Figure 6: Performance of **Genesis**, **NGE**, **Transform2Act**, **UMC-Message***, **Genesis w/o MoSAT**, and **Genesis w/o Enhanced-TCA** on ten co-design environments, with error regions to indicate standard error over four random seeds.

Environments. We conduct a comprehensive evaluation of Genesis with baselines in ten challenging co-design environments (Crawler, TerrainCrosser, Cheetah, Swimmer, Glider-Regular, Glider-Medium, Glider-Hard, Walker-Regular, Walker-Medium and Walker-Hard) on MuJoCo (Todorov et al., 2012). These environments encompass a diverse range of physical world types (2D, 3D), environment tasks, search space complexities, ground terrains, and initial designs to provide a multilevel evaluation. See Appendix A.1 for detailed descriptions.

5.1 Comparison with Baselines

We compare Genesis with the following baselines to highlight Genesis's performance: 1) Evolution Based Algorithms: **NGE** (Wang et al., 2018b) maintains a population of agents with different morphologies for random mutation and only preserves top-performing agents' children for further optimization. 2) RL Based Algorithms: **Transform2Act** (Yuan et al., 2021) propose to optimize a robot's morphology and control concurrently through reinforcement learning and achieve co-optimization. It utilizes graph neural networks (GNNs) and joint-specific MLPs (JSMLP) to foster knowledge sharing and specification. 3) Universal Control Algorithms: **UMC-Message** (Wang et al., 2018a; Huang et al., 2020) leverages a localized message transition mechanism for information exchange within the body, which is a typical method for universal morphology control. To make it suitable for embodiment co-design, we equip it with a policy network and our enhanced temporal credit assignment using reinforcement learning and denote it as UMC-Message*. The implementation details and full hyper-parameter of Genesis and all baselines are provided in Appendix A.2 and Appendix A.3.

As shown in Figure 6, Genesis achieves the highest task performance in all ten environments, with faster convergence speeds than baselines. Unlike the Universal Morphology Control (UMC) task, which focuses on limited specific morphologies Wang et al. (2018a); Huang et al. (2020), embodiment co-design deals with various changeable, morphology-agnostic robots. Consequently, UMC-Message* fails to converge within a limited time for complex tasks such as Glider-Hard, and Walker-Hard, due to its insufficient knowledge alignment mechanism for complicated, changable morphologies (*e.g*. JSMLP in Transform2Act and TopoPE in Genesis). Compared to evolutionary algorithms like NGE, we also find that RL-based methods demonstrate significant performance advantage due to a great sampling efficiency improvement within the same number of environmental

Table 1: Comparison of different position encoding choices for morphology representation.

Methods	CRAWLER	TERRAINCROSSER	CHEETAH	SWIMMER	GLIDER-REGULAR
TopoPE (ours)	**10381.96 ± 353.97**	**5056.01 ± 703.57**	**11611.52 ± 522.86**	1305.17 ± 15.25	**11082.29 ± 99.21**
w/ Traversal PE	8582.24 ± 987.44	4339.60 ± 260.60	10581.62 ± 846.69	1292.05 ± 16.71	9801.31 ± 748.13
w/o TopoPE	7490.83 ± 267.70	1122.29 ± 659.38	7451.37 ± 2275.37	**1371.20 ± 30.74**	10137.83 ± 713.60
Methods	GLIDER-MEDIUM	GLIDER-HARD	WALKER-REGULAR	WALKER-MEDIUM	WALKER-HARD
TopoPE (ours)	**11996.82 ± 595.51**	**10798.06 ± 298.39**	**12062.49 ± 513.07**	**12962.08 ± 537.34**	**11982.07 ± 520.78**
w/ Traversal PE	10758.70 ± 401.90	9106.77 ± 679.59	10389.40 ± 1080.94	10972.13 ± 584.04	11255.89 ± 121.04
w/o TopoPE	4099.99 ± 2057.92	109.48 ± 10.03	10149.67 ± 255.99	6730.01 ± 705.06	6529.87 ± 1863.59

interactions, supported by Yuan et al. (2021). By overcoming the bottlenecks in co-design, our approach goes even further: it achieves an average **60.03%** performance improvement over the strongest baseline in all the ten tasks.

5.2 ABLATION STUDIES

As mentioned in Section 4, our approach addresses inefficiencies in message and reward delivery, which includes the intra-agent level, inter-agent level, and agent-environment level. To better support our hypothesis and understand the importance of our key corresponding components (MoSAT, TopoPE, Enhanced-TCA), we designed four variants of our approach: (i) *Ours w/o MoSAT*, which removes the MoSAT structure to remove our attention-based centralized information processing across different limbs; (ii) *Ours w/o Enhanced-TCA*, which removes our temporal credit assignment mechanism and employs original PPO for optimization; (iii) *Ours w/o TopoPE*, which removes TopoPE from our methods. For a more comprehensive comparison, we also introduced another position encoding method from recent UMC methods, as: (iv) *Ours w/ Traversal PE*, where TopoPE is replaced with a traversal-based position embedding (Hong et al., 2021; Gupta et al., 2021a). Figure 6 presents the ablation studies for TopoPE and Enhanced-TCA, while Table 1 highlights the differences for different positional embedding choices. Additional detailed experimental results in are available in the Appendix (Table 11, Table 12).

(1) Intra-agent level: The MoSAT module provides centralized information processing. Removing this module results in significant performance degradation. Notably, Transform2Act adds an MLP to each limb, enhancing local message processing and model performance, but it increases the model size to 19.64M, which grows linearly with the complexity of the morphology. In contrast, our method is more lightweight, with each model only with 1.43M parameters, significantly enhancing scalability. **(2) Inter-agent level**: TopoPE facilitates morphological knowledge sharing among agents, aiding in adjusting knowledge for similar morphologies and reducing redundant learning costs. Compared to "Traversal PE" and "w/o TopoPE", TopoPE enhances agent performance and stabilizes learning. **(3) Agent-environment level**: Our proposed temporal credit assignment ensures that an agent receives balanced reward signals during both morphology design and control phases, markedly improving final performance across all the environments for embodiment.

6 CONCLUSIONS AND LIMITATIONS

This work proposes Genesis, a general reinforcement learning framework for efficient embodiment co-design. Our approach achieves efficient message and reward delivery through zero-decay message processing, effective morphological knowledge sharing, and balanced temporal credit assignment. Experiments demonstrate that Genesis surpasses previous methods in convergence speed and final performance while being efficient, lightweight, and scalable.

Limitations and Future Work. We acknowledge at least two limitations. Firstly, our approach remains focused on simulation environments, and further efforts are needed in transferring learned strategies to real physical systems. Secondly, our approach is a reward-driven reinforcement learning method, focusing on improving control effects, yet it lacks the ability to simulate the rich perception and execution capabilities of real biological intelligent systems. In future research, we expect embodied intelligence to evolve perception and execution components akin to biological evolutionary principles, enabling the realization of more efficient tasks for embodied intelligence.

REFERENCES

Jagdeep Bhatia, Holly Jackson, Yunsheng Tian, Jie Xu, and Wojciech Matusik. Evolution gym: A large-scale benchmark for evolving soft robots. *Advances in Neural Information Processing Systems*, 34:2201–2214, 2021.

Brain for AI Fandom. Central nervous system, 2024. URL `https://brain-for-ai.fandom.com/wiki/Central_nervous_system`. Accessed: 2024-05-21.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Heng Dong, Tonghan Wang, Jiayuan Liu, and Chongjie Zhang. Low-rank modular reinforcement learning via muscle synergy. *Advances in Neural Information Processing Systems*, 35:19861–19873, 2022.

Heng Dong, Junyu Zhang, Tonghan Wang, and Chongjie Zhang. Symmetry-aware robot design with structured subgroups. In *International Conference on Machine Learning*, pp. 8334–8355, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Encyclopaedia Britannica. Diffuse nervous system, 2024. URL `https://www.britannica.com/science/nervous-system/Diffuse-nervous-systems`. Accessed: 2024-05-21.

Agrim Gupta, Linxi Fan, Surya Ganguli, and Li Fei-Fei. Metamorph: Learning universal controllers with transformers. In *International Conference on Learning Representations*, 2021a.

Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. Embodied intelligence via learning and evolution. *Nature communications*, 12(1):5721, 2021b.

David Ha. Reinforcement learning for improving agent design. *Artificial Life*, 25(4):352–365, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Sunghoon Hong, Deunsol Yoon, and Kee-Eung Kim. Structure-aware transformer policy for inhomogeneous multi-task reinforcement learning. In *International Conference on Learning Representations*, 2021.

Suning Huang, Boyuan Chen, Huazhe Xu, and Vincent Sitzmann. Dittogym: Learning to control soft shape-shifting robots. *arXiv preprint arXiv:2401.13231*, 2024.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pp. 4455–4464, 2020.

Euan Judd, Gabor Soter, Jonathan Rossiter, and Helmut Hauscr. Sensing through the body - non-contact object localisation using morphological computation. In *Proceedings of the IEEE International Conference on Soft Robotics*, pp. 558–563, 2019.

Sam Kriegman, Douglas Blackiston, Michael Levin, and Josh Bongard. A scalable pipeline for designing reconfigurable organisms. *Proceedings of the National Academy of Sciences*, 117(4): 1853–1859, 2020.

Vitaly Kurin, Maximilian Igl, Tim Rocktäschel, Wendelin Boehmer, and Shimon Whiteson. My body is a cage: the role of morphology in graph-based incompatible control. In *International Conference on Learning Representations*, 2020.

Boyu Li, Haoran Li, Yuanheng Zhu, and Dongbin Zhao. Mat: Morphological adaptive transformer for universal morphology policy learning. *IEEE Transactions on Cognitive and Developmental Systems*, 16(4):1611–1621, 2024. doi: 10.1109/TCDS.2024.3383158.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. volume 34, pp. 18878–18890, 2021.

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

Kohei Nakajima, Helmut Hauser, Tao Li, and Rolf Pfeifer. Exploiting the dynamics of soft materials for machine learning. *Soft robotics*, 5(3):339–347, 2018.

Xinlei Pan, Animesh Garg, Animashree Anandkumar, and Yuke Zhu. Emergent hand morphology and control from optimizing robust grasps of diverse objects. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 7540–7547, 2021.

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

Rolf Pfeifer and Christian Scheier. *Understanding intelligence*. 2001.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 15–22, 1994.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033, 2012.

Brandon Trabucco, Mariano Phielipp, and Glen Berseth. Anymorph: Learning transferable polices by inferring agent morphology. In *International Conference on Machine Learning*, pp. 21677–21691. PMLR, 2022.

Patricia A Vargas, Ezequiel A Di Paolo, Inman Harvey, and Phil Husbands. *The horizons of evolutionary robotics*. 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Benyou Wang, Donghao Zhao, Christina Lioma, Qiuchi Li, Peng Zhang, and Jakob Grue Simonsen. Encoding word order in complex embeddings. *arXiv preprint arXiv:1912.12333*, 2019.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*, 2018a.

Tingwu Wang, Yuhao Zhou, Sanja Fidler, and Jimmy Ba. Neural graph evolution: Towards efficient automatic robot design. In *International Conference on Learning Representations*, 2018b.

Yuxing Wang, Shuang Wu, Tiantian Zhang, Yongzhe Chang, Haobo Fu, Qiang Fu, and Xueqian Wang. Preco: Enhancing generalization in co-design of modular soft robots via brain-body pre-training. In *Conference on Robot Learning*, pp. 478–498, 2023.

Julian Whitman, Matthew Travers, and Howie Choset. Learning modular robot control policies. *IEEE Transactions on Robotics*, 2023.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533, 2020.

Zheng Xiong, Jacob Beck, and Shimon Whiteson. Universal morphology control via contextual modulation. In *International Conference on Machine Learning*, pp. 38286–38300, 2023.

Jie Xu, Andrew Spielberg, Allan Zhao, Daniela Rus, and Wojciech Matusik. Multi-objective graph heuristic search for terrestrial robot design. In *2021 IEEE international conference on robotics and automation*, pp. 9863–9869, 2021.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. volume 33, pp. 5824–5836, 2020.

Ye Yuan, Yuda Song, Zhengyi Luo, Wen Sun, and Kris M Kitani. Transform2act: Learning a transform-and-control policy for efficient agent design. In *International Conference on Learning Representations*, 2021.

Allan Zhao, Jie Xu, Mina Konaković-Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Robogrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics*, 39(6):1–16, 2020.

A APPENDIX

The supplementary material provides additional results, discussions, and implementation details.

Our code is available in our supplementary material for reproduction and further study. Visit our website for videos and more additional visualizations.

A.1 ENVIRONMENT AND TASK DETAILS

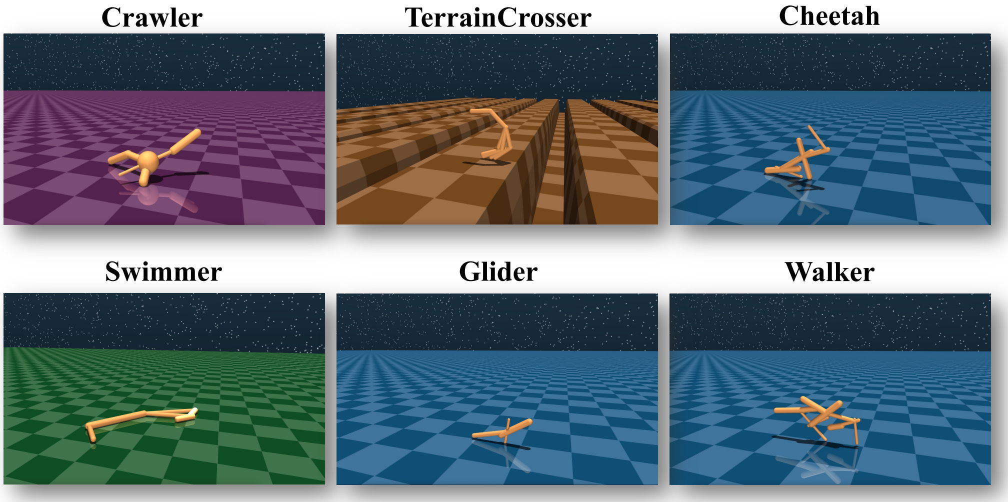


Figure 7: Randomly generated agents in six different environments for visualization. **Purple ground** indicates agents in a 3D physical world, **Green ground** represents agents in the xy-plane physical world, **Blue ground** denotes agents in the xz-plane physical world, and **Brown ground** denotes a physical world with variable terrain.

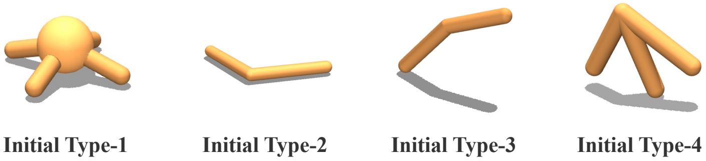


Figure 8: Visualization of four initial designs in the environments. *Type-1* consists of a structure with four limbs. *Type-2* and *Type-3* each includes two limbs connected by a joint, located in the xy-plane and xz-plane respectively. *Type-4* is composed of three limbs connected by two joints. Note that, Genesis can support almost *arbitrary* initial designs, and is not limited to the types specified.

In this section, we provide additional descriptions of the environments and tasks used in our experiments. Figure 7 displays randomly generated agents in six different environments. The first four environments: CRAWLER, TERRAINCROSSER, CHEETAH, and SWIMMER are derived from previous work (Yuan et al., 2021) to ensure a fair comparison. We have also introduced two additional environments, GLIDER and WALKER, to broaden testing scope and provide a more comprehensive algorithm evaluation.

Each agent consists of multiple limbs connected by joints, each joint equipped with a motor for controlling movement. Sensors within the limbs monitor positional coordinates, velocity, and angular velocity. For each limb, its attributes include limb length and limb size. For each joint, its attributes cover rotation range and maximum motor torque. Each episode starts with a simple initial design, as demonstrated in Figure 8. Through a series of topological and attribute modifications, the agent evolves to its final morphology. Meanwhile, the control policy is required to optimize concurrently.

Crawler The agent inhabits a 3D environment with flat ground at $z = 0$. The initial design is the *Type-1* in Figure 8. Each limb can have up to two child limbs, except for the root limb. For the root limb, its height and 3D world velocity are also included in the environment state. The reward function is defined as:

$$r_t = \frac{|p^x_{t+1} - p^x_t|}{\Delta t} - w \cdot \frac{1}{J} \sum_{u \in V_t} \|a^e_u\|^2 \tag{A.1}$$

where $w = 0.0001$ is the weighting factor for the control penalty term, J is the total number of limbs, and $\Delta t = 0.04$.

TerrainCrosser The agent evolves in a terrain-variable environment, where the terrain features varying height differences. The maximum height difference of the terrain is $z_{max} = 0.5$. The agent must navigate these gaps to move forward. The initial design is the *Type-3* in Figure 8. The terrain is generated from a single-channel image, with different values representing different height rates. Each limb of the agent can have up to three child limbs. For the root limb, its height, 2D world velocity, and a variable encoding the terrain information are included in the environment state. The reward function is defined as:

$$r_t = \frac{|p^x_{t+1} - p^x_t|}{\Delta t}, \tag{A.2}$$

where $\Delta t = 0.008$, and the episode is terminated when the root limb height is below 1.0.

Cheetah The agent in this environment evolves with flat ground at $z = 0$. The initial design is the *Type-3* in Figure 8. Each limb of the agent can have up to three child limbs. For the root limb, its height and 2D world velocity are added to the environment state. The reward function is defined in Equation (A.2). The episode is terminated when the root height is below 0.7.

Swimmer Swimmer is designed to cover snake-like creatures in water. The agent evolves in water with a viscosity of $vis = 0.1$ for water simulation. The initial design is the *Type-2* in Figure 8. Each limb support up to three child joints. For the root limb, its 2D world velocity is incorporated into the environment state. The reward function is the same as TerrainCrosser in Equation (A.2).

Glider The agent in this environment evolves on flat ground. The initial design is the *Type-4*, as shown in Figure 8. In Glider, the agent's search depth is limited to three times that of the initial design, encourage fully exploration of a relatively shallow search space. We also provide three different task levels: regular, medium, and hard, where each limb of the agent can have up to one, two, three child limbs. The reward function is defined in Equation (A.2).

Walker The agent evolves on flat ground. The starting design is *Type-4* in Figure 8. The search depth for the agent is capped at four times the initial design to promote thorough exploration within a comparatively shallow search space. Similarly, three levels of task difficulty are offered, which have the same meaning as described in Glider. The reward function is specified in Equation (A.2).

Note that the reward function are kept simple and consistent in all environments. Unlike common practices in OpenAI-Gym (Brockman et al., 2016), we do not provide any additional reward priors (*e.g.*alive bonus) to facilitate learning, which presents higher requirements to the algorithm robustness.

A.2 IMPLEMENTATION DETAILS

A.2.1 TRAINING DETAILS

In line with standard reinforcement learning practices, we employed distributed trajectory sampling across multiple CPU threads to accelerate training. Each model is trained using four random seeds on a system equipped with 112 Intel® Xeon® Platinum 8280 cores and six Nvidia RTX 3090 GPUs. Our main code framework is based on Python 3.9.18 and PyTorch 2.0.1. For all the environments used in our work, it takes approximately only 30 hours to train a model with 20 CPU cores and a single NVIDIA RTX 3090 GPU on our server.

A.2.2 HYPERPARAMETERS

For Genesis, we ran a grid search over MoSAT layer normalization $\in \{\text{w/o-LN}, \text{Pre-LN}, \text{Post-LN}\}$, Policy network learning rate $\in \{5e{-}5, 1e{-}4, 3e{-}4\}$, Value network learning rate $\in \{1e{-}4, 3e{-}4\}$, and MoSAT hidden dimension $\in \{32, 64, 128, 256\}$. We did not search further for the environmental settings, optimizer configurations, PPO-related hyperparameters, or the training batch size and minibatch size. Instead, we strictly maintained consistency with previous works Wang et al. (2018b); Yuan et al. (2021); Kurin et al. (2020) to ensure a fair comparison. With further hyperparameter tuning, our algorithm could achieve higher performance levels. Table 2 displays the hyperparameters Genesis adopted across all experiments.

For Transform2Act, we followed previous work Yuan et al. (2021) and its official released code repository [3], and used $\mathrm{GraphConv}$ as the GNN layer type, policy GNN size $(64, 64, 64)$, policy learning rate $5e-5$, value GNN size $(64, 64, 64)$, value learning rate $3e-4$, JSMLP activation function Tanh, JSMLP size $(128, 128, 128)$ for the policy, MLP size $(512, 256)$ for the value function, which were the best values they picked using grid searches.

For UMC-Message, to make them suitable for embodiment co-design, we equip them with a policy network and employ our temporal credit assignment via reinforcement learning. The network parameters and training settings are kept consistent with those used in Genesis and Transform2Act to ensure a fair comparison. It adopted GNN layer type of $\mathrm{GraphConv}$, policy GNN size $(64, 64, 64)$, policy MLP size $(128, 128)$, policy learning rate $5e-5$, value GNN size $(64, 64, 64)$, value MLP size $(512, 256)$, value learning rate $3e-4$. We followed previous work Huang et al. (2020) and also referred to the publicly released code [4] for implementation.

For NGE, we follow previous works Wang et al. (2018b); Yuan et al. (2021) according to the public release code [5], and used number of generations 125, agent population size 20, elimination rate 0.15, GNN layer type $\mathrm{GraphConv}$, MLP activation Tanh, policy GNN size $(64, 64, 64)$, policy MLP size $(128, 128)$, value GNN size $(64, 64, 64)$, value MLP size $(512, 256)$, policy learning rate $5e-5$, and value learning rate $3e-4$, which were the best searched values described by previous work.

Table 2: Hyperparameters of Genesis adopted in all the experiments

Hyperparameter	Value
Number of Topology Design N^{topo}	5
Number of Attribute Design N^{attr}	1
MoSAT Layer Normalization	Pre-LN
MoSAT Activation Function	SiLu
MoSAT FNN Scaling Ratio r	4
MoSAT Block Number (Policy Network)	3
MoSAT Block Number (Value Network)	3
MoSAT Hidden Dimension (Policy Network)	64
MoSAT Hidden Dimension (Value Network)	64
Optimizer	Adam
Policy Learning Rate	5e-5
Value Learning Rate	3e-4
Clip Gradient Norm	40.0
PPO Clip ϵ	0.2
PPO Batch Size	50000
PPO Minibatch Size	2048
PPO Iterations Per Batch	10
Training Epochs	1000
Discount factor γ	0.995
GAE λ	0.95

---

[3] `https://github.com/Khrylx/Transform2Act`

[4] `https://github.com/huangwl18/modular-rl`

[5] `https://github.com/WilsonWangTHU/neural_graph_evolution`

A.3 ALGORITHM DETAILS

Algorithm 1 illustrates the overall training process of Genesis, which is based on PPO for efficient reinforcement learning. We highlight three key components: the interaction process, our temporal credit assignment based on GAE, and the main loop for iterative optimization.

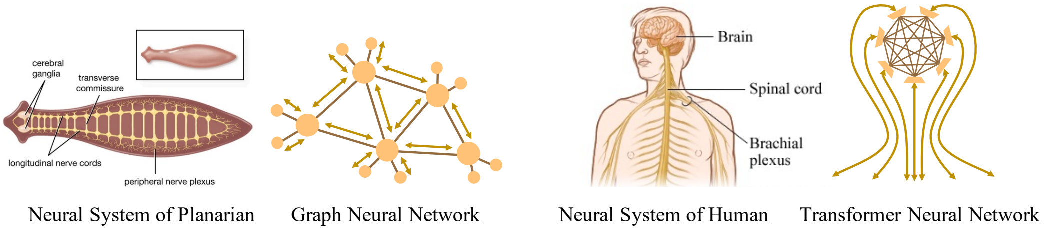
Algorithm 1: Synchronous Learning Algorithm for Genesis

```
Input: Replay Buffer $\mathbf{B}$, Batch $\mathbb{B}$, Optimizer optimizer
Initialize: Policy networks: $\pi_\theta : \{\pi_\theta^{topo}, \pi_\theta^{attr}, \pi_\theta^{C}\}$; Value networks: $V_\theta : \{V_\theta^{topo}, V_\theta^{attr}, V_\theta^{C}\}$
            $\mathbf{B} \leftarrow \varnothing, \mathbb{B} \leftarrow \varnothing$, Discount factor $\gamma$, GAE Exponential Weight $\lambda$
Function INTERACT(Policy: $\pi$, Replay Buffer: $\mathbf{B}$):
    while $\mathbf{B}$ not reaching max buffer size do
        $\mathcal{G}_0 \leftarrow$ initial design
        $\Phi \leftarrow topo$                                            ▷ Topology design stage
        for $t = 0, 1, ..., N^{topo} - 1$ do
            $\boldsymbol{a}_t^{topo} \sim \pi^{topo}(\boldsymbol{a}_t^{topo} | \mathcal{G}_t)$                    ▷ Sample topology actions from all limps
            $\mathcal{G}_{t+1} \leftarrow$ apply $\boldsymbol{a}_t^{topo}$ to modify the topology $(V_t, E_t)$ of current design $\mathcal{G}_t$
            $r_t = 0$ ; $\mathcal{S}_t = \mathcal{S}$; store $\{r_t, \varnothing, \boldsymbol{a}_t^{topo}, \mathcal{G}_t, \mathcal{S}_t, 0, 0\}$ into $\mathbf{B}$   ▷ Update Buffer $\mathbf{B}$ with transition
        end
        $\Phi \leftarrow attr$                                            ▷ Attribute design stage
        for $t = N^{topo}, ..., N^{topo} + N^{attr} - 1$ do
            $\boldsymbol{a}_t^{attr} \sim \pi^{attr}(\boldsymbol{a}_t^{attr} | \mathcal{G}_t)$                    ▷ Sample attribute actions from all limps
            $\mathcal{G}_{t+1} \leftarrow$ apply $\boldsymbol{a}_t^{attr}$ to modify the attribute $(A_t^v, A_t^e)$ of current design $\mathcal{G}_t$
            $r_t = 0$ ; $\mathcal{S}_t = \mathcal{S}$; store $\{r_t, \varnothing, \boldsymbol{a}_t^{attr}, \mathcal{G}_t, \mathcal{S}_t, 0, 0\}$ into $\mathbf{B}$   ▷ Update Buffer $\mathbf{B}$ with transition
        end
        $\Phi \leftarrow ctrl$                                            ▷ Control stage
        $\boldsymbol{s}_t \leftarrow$ Env.Reset(0)                  ▷ $\boldsymbol{s}_t = \{s_{v,t}\}$ denotes the sensor states from all limps
        for $t = N^{topo} + N^{attr}, ..., T - 1$ do
            $\boldsymbol{a}_t^{ctrl} \sim \pi^{ctrl}(\boldsymbol{a}_t^{ctrl} | \boldsymbol{s}_t, \mathcal{G}_{done})$
            $r_t, \boldsymbol{s}_{t+1}, \mathbb{T}_t, \mathbb{C}_t \leftarrow$ Env.Step($\boldsymbol{a}_t^{ctrl}$)        ▷ $\mathbb{T}_t, \mathbb{C}_t$ denotes termination and trunction
            $\mathcal{S}_t = \mathcal{S}$; store $\{r_t, \boldsymbol{s}_t, \boldsymbol{a}_t^{ctrl}, \mathcal{G}_t, \mathcal{S}_t, \mathbb{T}_t, \mathbb{C}_t\}$ into $\mathbf{B}$      ▷ Update Buffer $\mathbf{B}$ with transition
        end
    end
end
Function ENHANCEDGAE(Value Function: $V_\theta$, Replay Buffer: $\mathbf{B}$):
    for $t = T - 1, ..., 0$ do
        $U_t = r_t + U_{t+1} \cdot (1 - \mathbb{T}_t \vee \mathbb{C}_t)$                              ▷ Calculate return
        $\delta_t = r_t + \gamma V_\theta(s_{t+1}) \cdot (1 - \mathbb{T}_t) - V_\theta(s_t)$               ▷ Calculate the TD-error term
        if $\mathcal{S}_t = ctrl$ then
            $\hat{A}_t = \delta_t + \gamma\lambda\hat{A}_{t+1} \cdot (1 - \mathbb{T}_t \vee \mathbb{C}_t)$            ▷ Calculate advantage for the control stage
        else
            $\hat{A}_t = U_t - V_\theta(s_t)$                                 ▷ Calculate advantage for the design stage
        end
        $\hat{R}_t = V_\theta(s_t) + \hat{A}_t$                                  ▷ Calculate the target value
        store $\{\hat{A}_t, \hat{R}_t\}$ into $\mathbf{B}$            ▷ Append $\hat{A}_t$ and $\hat{R}_t$ to the corresponding transition item in $\mathbf{B}$.
    end
end
Function MAIN():
    while not reaching max iterations do
        $Th_i \leftarrow$ Thread(INTERACT, $\pi_\theta$, $\mathbf{B}$)                ▷ We use multiple CPU threads for sampling
        $Th_i.join()$                                          ▷ Gather trajectories collected from threads
        ENHANCEDGAE($V_\theta$, $\mathbf{B}$)                   ▷ Perform temporal credit assignment for co-design
        while not reaching max epochs do
            Update $\mathbb{B} \leftarrow \mathbf{B}$                           ▷ Sample a random batch $\mathbb{B}$ from Buffer $\mathbf{B}$
            Calculate PPO loss $\mathcal{L}_{ppo} = \mathcal{L}^{policy} + \mathcal{L}^{value}$      ▷ According to Equation (4.11) and (4.13)
            optimizer $\leftarrow$ Gradient from $\mathcal{L}_{ppo}$                  ▷ Gradient descent to update $\pi_\theta$ and $V_\theta$
        end
    end
end
```

A.4 MOTIVATIONS

In this section, we will detail the motivations behind designing MoSAT and TopoPE for embodiment co-design, aiming to provide further insights.

A.4.1 CENTRALIZED MESSAGE PROCESSING AND MOSAT

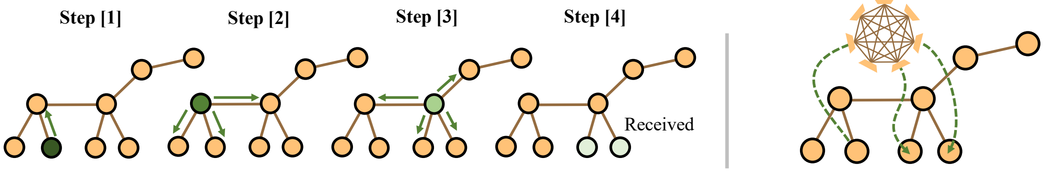


(a) Localized Message Processing

(b) Centralized Message Processing

Figure 9: Comparative overview of natural and artificial neural processing systems. (a) Localized message processing in planarians and GNNs. (b) Centralized message processing in human brains and Transformers. Relevant images are sourced from Encyclopaedia Britannica (2024); Brain for AI Fandom (2024).

As demonstrated in Figure 9, GNN-like neural systems are commonly found in simple organisms such as planarians, where sensory information are connected through neural networks, for distributed and localized processing. In contrast, advanced creatures such as humans utilize a centralized signal processing approach, where signals from various body parts are centrally processed in the brain, leveraging scalability advantages similar to the self-attention mechanism within transformers.



(a) Message via Aggregation and Broadcasting

(b) Messaging via Self-Attention

Figure 10: Comparison of different message delivery mechanisms between GNN and Transformer.

Figure 10 further illustrates the different message delivery mechanisms between GNN and Transformer. GNN uses aggregation and broadcasting for message transmission, resulting in a progressive reduction of information. As demonstrated in Figure 10 (a), the dog-like robot needs to adjust its posture throughout its motion. The GNN's localized message processing approach requires signals from distant locations to propagate multiple times before reaching the target actuator. In contrast, Transformers can provide faster message transfer and interaction, by employing self-attention to facilitate direct point-to-point and point-to-multipoint message delivery. Inspired by this, we propose MoSAT in Section 4.1. MoSAT first maps sensor information to the latent space, and leverage self-attention for signal interactions for centralized decision making.

Meanwhile, in GNNs, the message propagation mechanism allows for an implicit representation of morphology. However, while transformers leverage self-attention for direct message delivery, they do not offer an asymmetric information propagation mechanism to differentiate positions between different body parts.

A.4.2 MORPHOLOGY POSITION EMBEDDING AND TOPOPE

Position encoding have proven effective in location representation within the natural language processing field (Vaswani et al., 2017; Shaw et al., 2018; Raffel et al., 2020; Wang et al., 2019).

An effective representation of the robot's morphology is crucial for co-designing morphology and control policies. In our work, we propose the **Topology Position Embedding (TopoPE)** to encode the morphology in a way that is compatible with Transformer-based architectures. TopoPE assigns a unique embedding to each limb based on its topological position within the robot's morphology tree. Specifically, the embedding index for a limb is derived from the path from the root node to the limb, capturing the structural relationships within the morphology.

In previous works (Trabucco et al., 2022; Gupta et al., 2021a), morphology encodings often rely on traversal sequences like depth-first search (DFS) or manual naming (Trabucco et al., 2022; Li et al., 2024) conventions based on a "full model" of the robot. When limbs are removed to generate variants, the names of the remaining limbs remain unchanged, facilitating consistent encoding.

However, in our setting, there is no predefined "full model," and the robot's morphology is dynamically generated during the co-design process. Manually naming limbs is impractical in this context. Our TopoPE addresses this challenge by using a topology indexing mechanism, which uses the path to the root as the embedding index. This method naturally extends to dynamically changing morphologies and ensures that similar substructures share similar embeddings, promoting generalization across different morphologies.

Moreover, unlike learnable position embeddings that are specific to particular morphologies, our approach can be extended using non-learnable embeddings, such as sinusoidal embeddings (Vaswani et al., 2017), which offer better extrapolation to unseen morphologies and eliminate the need for training the embeddings.

To demonstrate the effectiveness of TopoPE, we conducted ablation studies comparing models with and without TopoPE. As shown in Table 1 and Figure 11, incorporating TopoPE significantly improves performance across various tasks. This indicates that TopoPE provides a more informative and stable encoding of the morphology, facilitating better learning of control policies.

In contrast to other morphology-aware positional encodings, our TopoPE is specifically designed to handle dynamic and diverse morphologies without relying on a fixed full model or manual limb naming. Additionally, our approach aligns well with the Transformer architecture, allowing the use of standard attention mechanisms to capture interactions between different limbs based on their topological relationships.

A.5 ADDITIONAL RESULTS

A.5.1 QUANTITATIVE RESULTS

Table 3: Comparison of Genesis, its ablation variants, and baseline methods.

Methods	CRAWLER	TERRAINCROSSER	CHEETAH	SWIMMER	GLIDER-REGULAR
Genesis (Ours)	**10381.96 ± 353.97**	**5056.01 ± 703.57**	**11611.52 ± 522.86**	**1305.17 ± 15.25**	**11082.29 ± 99.21**
- w/o MoSAT	818.92 ± 57.78	407.30 ± 4.50	662.88 ± 74.88	476.26 ± 19.95	447.72 ± 7.56
- w/o Enhanced-TCA	4994.44 ± 160.14	2668.66 ± 844.22	8158.74 ± 55.71	786.32 ± 19.39	8317.88 ± 498.26
Transform2Act	4185.63 ± 334.04	2393.84 ± 692.96	8405.70 ± 815.64	732.20 ± 22.61	6901.68 ± 374.42
NGE	1545.13 ± 626.54	881.71 ± 459.96	2740.79 ± 515.51	395.90 ± 173.85	1567.84 ± 756.74
UMC-Message	6492.90 ± 441.04	1411.51 ± 705.68	5785.40 ± 2110.77	961.20 ± 183.03	7354.34 ± 2145.22
Methods	GLIDER-MEDIUM	GLIDER-HARD	WALKER-REGULAR	WALKER-MEDIUM	WALKER-HARD
Genesis (Ours)	**11996.82 ± 595.51**	**10798..06 ± 298.39**	**12062.49 ± 513.07**	**12962.08 ± 537.34**	**11982.07 ± 520.78**
- w/o MoSAT	489.75 ± 5.74	533.17 ± 14.20	555.33 ± 18.15	708.32 ± 12.72	827.33 ± 47.71
- w/o Enhanced-TCA	7454.55 ± 289.93	7592.03 ± 1023.70	7286.30 ± 735.55	6069.51 ± 652.96	6126.73 ± 572.85
Transform2Act	5573.44 ± 519.22	6120.37 ± 1380.74	8685.47 ± 1008.88	6287.15 ± 426.99	4645.31 ± 294.81
NGE	1649.60 ± 763.55	2339.90 ± 487.22	1402.85 ± 595.54	2600.39 ± 481.74	1575.87 ± 508.11
UMC-Message	4726.44 ± 2406.35	425.49 ± 141.02	5417.14 ± 2019.43	5347.70 ± 2397.85	2783.09 ± 1587.06

As demonstrated in Figure 6, we present the full training curves for Genesis with baselines including Transform2Act, UMC-Message, NGE, and ablation variants of *ours w/o MoSAT* and *ours w/o Enhanced-TCA* across ten co-design environments. Each model was trained using four random seeds. For all baselines, we employed the best performance configurations reported by previous works, as is detailed in Section A.2. Table 3 further presents related metrics, with each cell showing the mean and standard deviation of episode rewards for the corresponding algorithm in each environment.

A.6 ADDITIONAL ABLATION STUDIES ON TOPOPE AND ENHANCED-TCA

We provide additional ablation studies on our proposed TopoPE and Enhanced-TCA to provide more insights, which is demonstrated in Figure 11 and Figure 12.

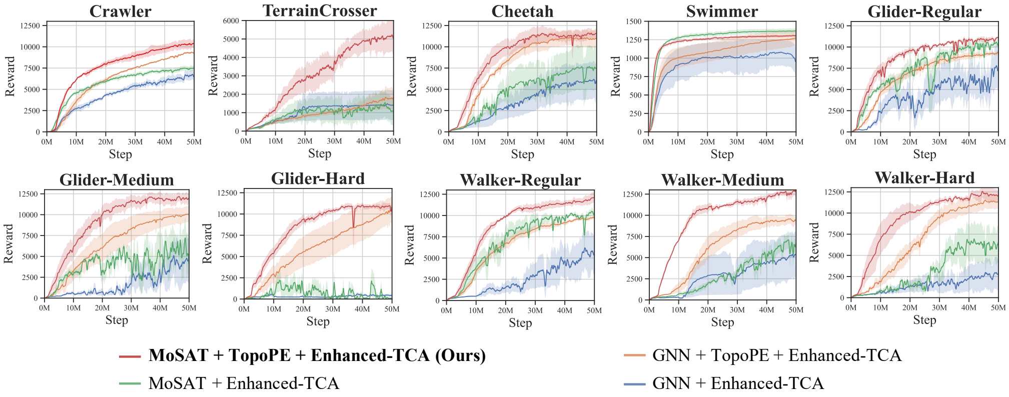


Figure 11: Extensive experiments on our proposed simple-yet-effective Topology Position Encoding (TopoPE) across different architectures of MoSAT and GNN, validating TopoPE as an efficient and general method for morphology representation. (1) MoSAT: w/o TopoPE → with TopoPE; (2) GNN: w/o TopoPE → with TopoPE; Both sets demonstrated the obvious performance improvements brought by TopoPE.

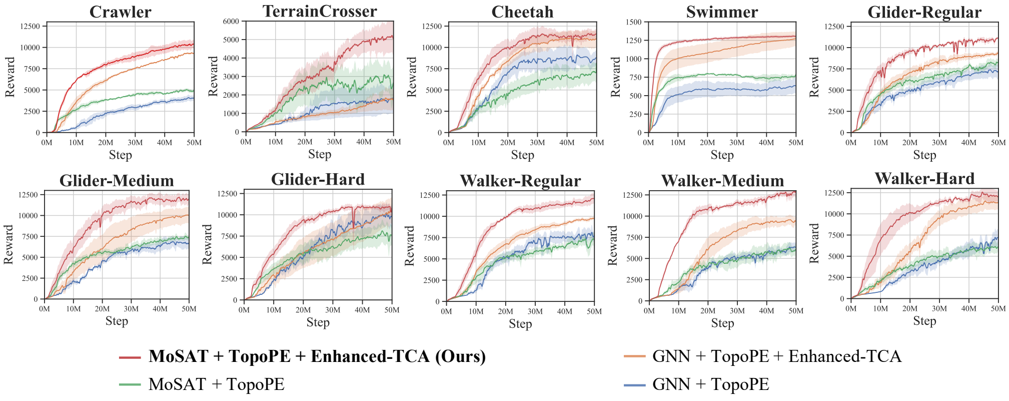
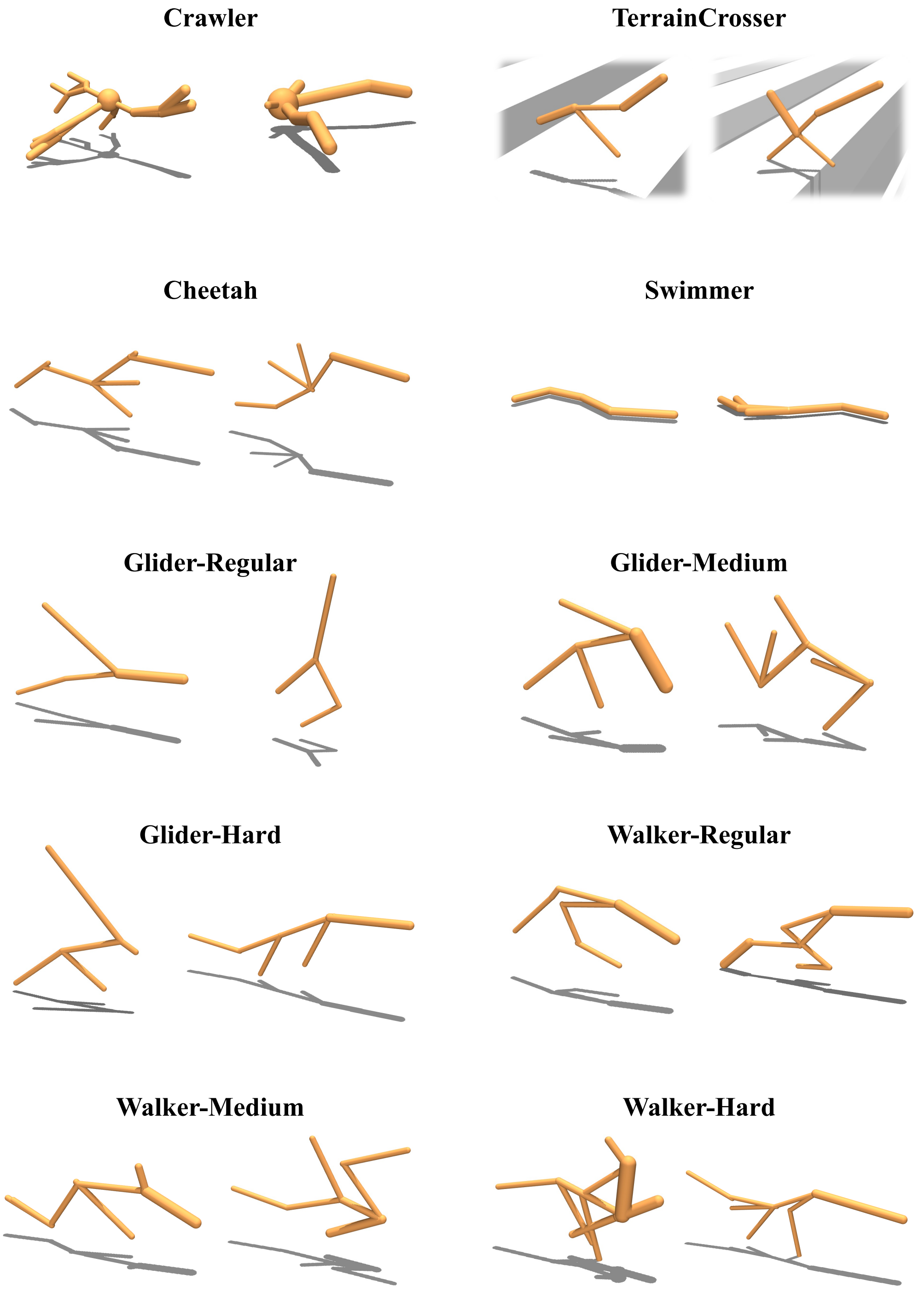


Figure 12: Extensive experiments on our proposed Temporal Credit Assignment Mechanism (Enhanced-TCA) across different architectures of MoSAT and GNN, validating Enhanced-TCA mechanism as an efficient method for enhancing bi-level optimization. (1) MoSAT: w/o Enhanced-TCA → with Enhanced-TCA; (2) GNN: w/o Enhanced-TCA → with Enhanced-TCA; Both sets demonstrated the obvious performance improvements brought by our Enhanced-TCA mechanism.

A.7 MORE VISUALIZATION RESULTS

In this section, we provide additional visualization results for embodied agents generated by Genesis across ten co-design environments, presenting their poses in motion. All the agents are randomly captured from the simulator.



Crawler

TerrainCrosser

Cheetah

Swimmer

Glider-Regular

Glider-Medium

Glider-Hard

Walker-Regular

Walker-Medium

Walker-Hard

Figure 13: Visualization of embodied agents generated by Genesis on different environments.

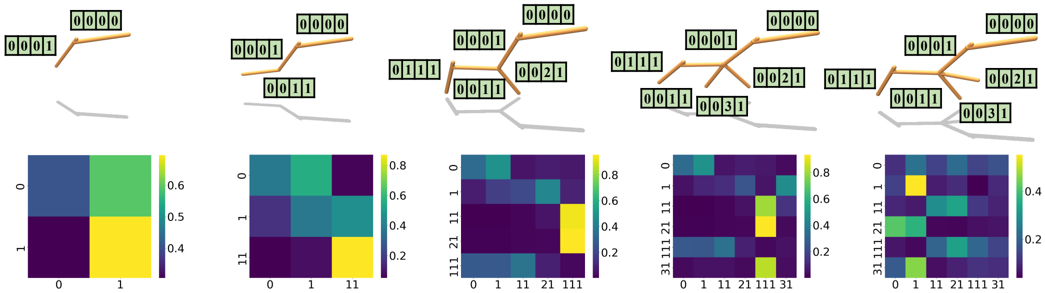


Figure 14: Visualization for Genesis's design process on Cheetah.

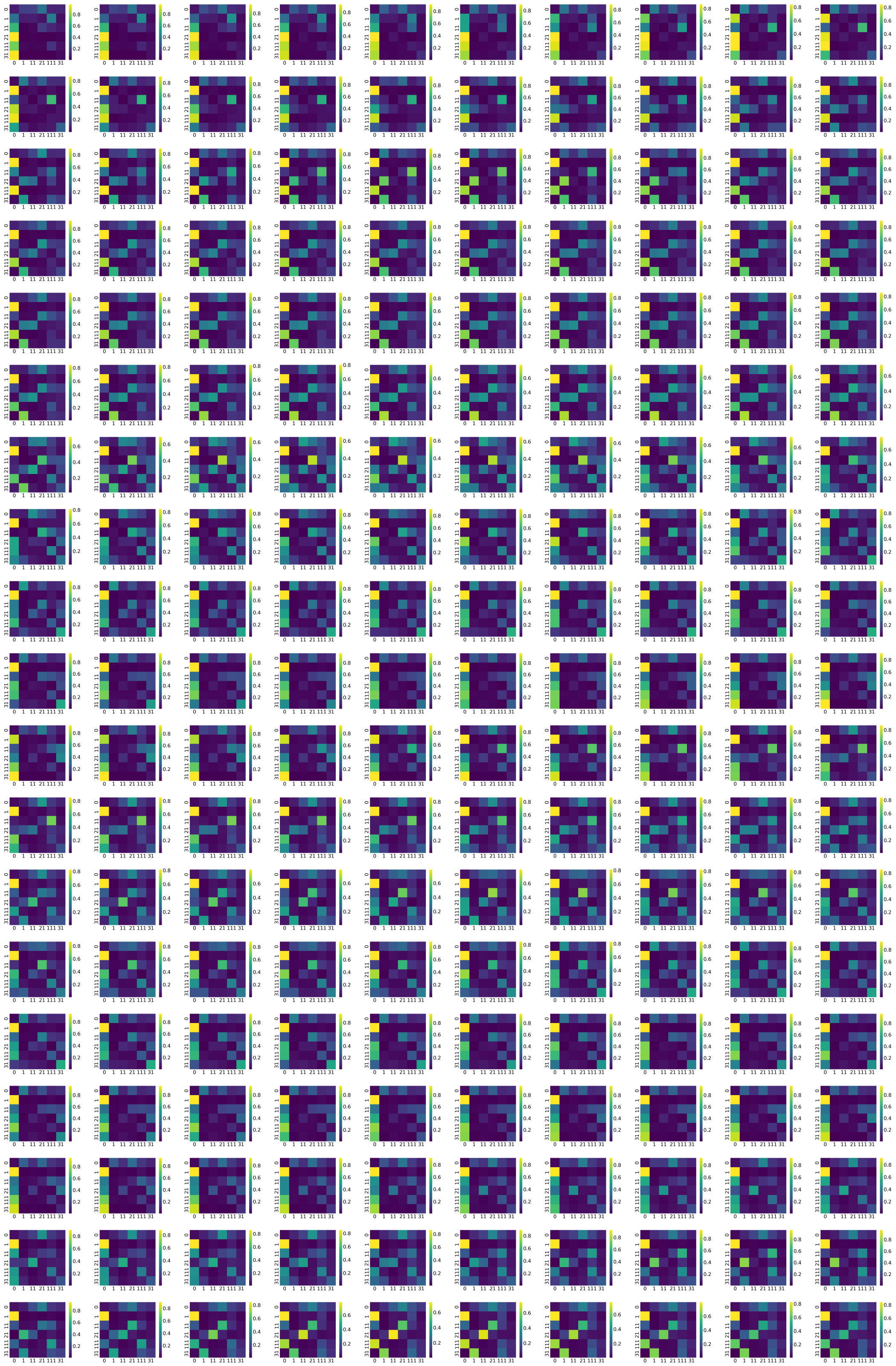


Figure 15: Visualization for Genesis's attention map during the control process on Cheetah.