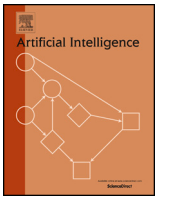
Artificial Intelligence 348 (2025) 104421

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Learngene: Inheritable "genes" in intelligent agents

Fu Feng [a,b], Jing Wang [a,b,*], Xu Yang [a,b], Xin Geng [a,b,*]

[a] School of Computer Science and Engineering, Southeast University, Nanjing, China
[b] Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China

ARTICLE INFO

Keywords:
Knowledge transfer
Genetic mechanism
Intelligent agents
Machine evolution
Gene

ABSTRACT

Biological intelligence has driven significant progress in artificial intelligence (AI), but a critical gap remains: biological systems inherit innate abilities from genes, with brains initialized by blueprints refined over 3.5 billion years of evolution, while machines rely heavily on inefficient, data-driven learning from scratch. This gap arises from the lack of a genetic mechanism in machines to transfer and accumulate inheritable knowledge across generations. To bridge this gap, we propose learngenes, network fragments that act as inheritable "genes" for machines. Unlike conventional knowledge transfer methods, learngenes enable efficient and universal knowledge transfer by selectively encapsulating task-agnostic knowledge. To facilitate the transfer and accumulation of task-agnostic knowledge across generations, we introduce Genetic Reinforcement Learning (GRL), a framework that simulates the learning and evolution of organisms in intelligent agents following Lamarckian principles. Through GRL, we identify learngenes as network fragments within agents' policy networks, equipping newborn agents with innate abilities for rapid adaptation to novel tasks. We demonstrate the advantages of learngene-based knowledge transfer over evolution-based search and traditional pre-trained models, and show how learngenes evolve through the accumulation of task-agnostic knowledge. Overall, this work establishes a novel paradigm for knowledge transfer and model initialization in AI, offering new possibilities for more adaptive, efficient, and scalable learning systems.

1. Introduction

Biological intelligence [1,2] has driven significant advancements in artificial intelligence (AI) [3]. For example, reinforcement learning (RL), inspired by animal learning processes [4,5], underpins AlphaGo's success against world-class Go players [6,7]. Similarly, artificial neural networks (ANNs), modeled after human brains' information-processing mechanisms [8,9], enable GPT-4's near-human performance in natural language understanding and generation [10,11], and Stable Diffusion's ability to generate artwork rivaling that of human artists [12–14].

Despite remarkable advancements, a notable gap persists between the learning mechanisms of living organisms and intelligent machines. For instance, spiders are born with the innate ability to spin webs [15], and newborn colts can walk shortly after birth [16]. These innate behaviors, which equip animals with the ability to survive and adapt efficiently to their environments [17,18], have been shaped by evolutionary processes over 3.5 billion years [19,20]. In contrast, machines typically learn from scratch and depend on vast amounts of data [21–23], resulting in a time-consuming and inefficient learning process [24–26].

* Corresponding authors at: School of Computer Science and Engineering, Southeast University, Nanjing, China.
E-mail addresses: wangjing91@seu.edu.cn (J. Wang), xgeng@seu.edu.cn (X. Geng).

https://doi.org/10.1016/j.artint.2025.104421
Received 1 March 2024; Received in revised form 11 September 2025; Accepted 17 September 2025

Available online 22 September 2025

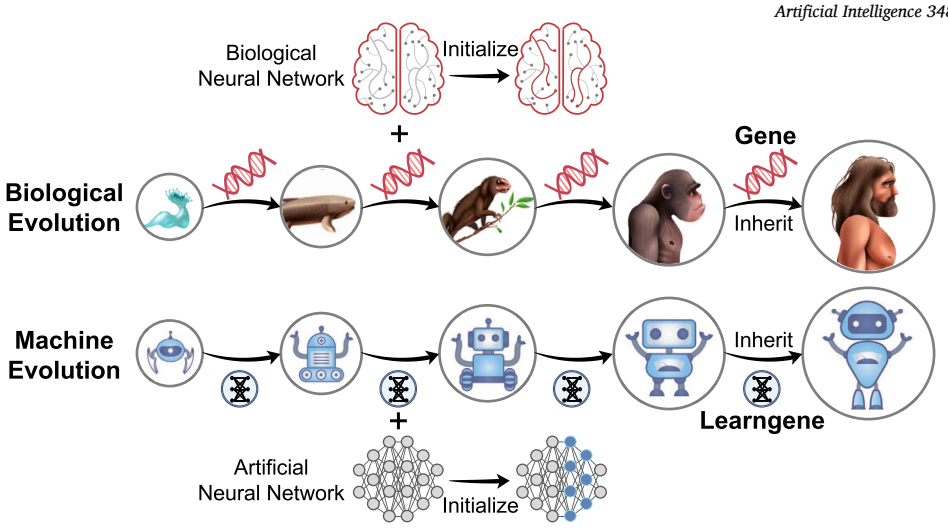


Fig. 1. Biological Evolution and Machine Evolution. In nature, genes have evolved over 3.5 billion years, endowing descendants with innate abilities by initializing their brains with evolution-shaped blueprints. Similarly, we construct a genetic mechanism for machines to identify learngenes that evolve as carriers of inheritable knowledge. By encapsulating task-agnostic knowledge, learngenes equip machines with innate abilities, enabling efficient adaptation to new tasks.

The lack of innate capabilities in machines stems from the absence of a genetic mechanism comparable to that in living organisms. In biological systems, genes transfer environment-agnostic knowledge across generations through inheritance, effectively initializing the brains of descendants with blueprints shaped by evolution [27,28]. While natural evolution has inspired evolutionary learning (EL) in AI [29–31], current EL-based approaches primarily focus on searching for optimal solutions through population-based optimization, neglecting the critical role of genes in facilitating the transfer and accumulation of knowledge across generations [32,33].

To bridge this gap, we construct a genetic mechanism for machines, centered on knowledge transfer (see Fig. 1). At its core is the concept of the "learngene", defined as network fragments that serve as carriers of inheritable knowledge within machines, akin to genes in biological systems. Unlike conventional transfer learning methods, which typically indiscriminately transfer extensive knowledge and may result in redundancy [34], bias [35], or even negative effects [36,37], learngenes offer a more targeted and efficient solution. By selectively compressing inheritable knowledge into compact fragments through a "genomic bottleneck" (Section 3.2.2), learngenes enable the efficient and universal transfer of knowledge to descendant models, establishing a robust mechanism for knowledge accumulation across generations.

In this study, we focus on RL-trained intelligent agents and consider inheritable knowledge as task-agnostic knowledge within neural networks, with the remainder classified as task-specific. The notion of "task-agnostic knowledge", adopted from the meta-learning literature [38,39], refers to knowledge that generalizes across diverse tasks without being restricted to any particular instance. Here, it specifically encompasses fundamental control principles, generalized coordination patterns, and core locomotion strategies that are broadly applicable to control tasks. By selectively compressing and transferring this knowledge, learngenes enhance transferability and universality while minimizing capacity. We term this process knowledge condensation (Section 3.2.2), where task-agnostic knowledge is encapsulated within learngenes, while task-specific knowledge remains independently preserved in non-learngene components.

To achieve the condensation of task-agnostic knowledge, as well as facilitate its transfer and accumulation across machines, we propose Genetic Reinforcement Learning (GRL), a computational framework inspired by natural genetic processes (see Fig. 2). GRL integrates knowledge transfer and condensation into an evolutionary paradigm, where agents tackle variable terrain tasks simulating diverse environmental challenges [40,41]. Aligning with Lamarckian principles, agents inherit learngenes encapsulating accumulated task-agnostic knowledge from their ancestors and adapt to their respective environments through RL, refining the parameters of inherited learngenes to further accumulate task-agnostic knowledge. Across generations, superior learngenes, encapsulating the most transferable and generalized knowledge, are selectively passed on. This iterative process ensures the continuous refinement and accumulation of task-agnostic knowledge across diverse agents and tasks, enabling learngenes to serve as efficient and universal carriers of inheritable knowledge.

We make a breakthrough in constructing a genetic mechanism for machines, with the following key contributions:

- We introduce the concept of "learngenes", inheritable "genes" in intelligent agents, represented as network fragments. Learngenes enable the transfer of task-agnostic knowledge, providing a novel mechanism for efficient and universal knowledge transfer in machines.
- We propose Genetic Reinforcement Learning (GRL), a pioneering genetic framework that facilitates the condensation and accumulation of task-agnostic knowledge within learngenes, marking the first effort to establish genetic mechanisms in machines.

- Extensive experiments demonstrate that learngenes equip newborn agents with innate abilities, enabling efficient adaptation to novel tasks. Additionally, we demonstrate that the evolution of agents follows Lamarckian principles, with the accumulation of task-agnostic knowledge driving the continual evolution of learngenes.

2. Related work

The concept of "learngene" is introduced to facilitate the transfer of task-agnostic knowledge among machines (or artificial neural networks), inspired by the role of genes in transferring environment-agnostic knowledge in biological evolution [42]. To model this process, we propose Genetic Reinforcement Learning (GRL), a framework that establishes genetic mechanisms in RL agents, simulating biological evolution. The following sections differentiate learngenes from conventional transfer learning methods [43] (Section 2.1) and contrast GRL with traditional evolutionary algorithms (EAs) (Section 2.2) and their applications, including neuroevolution [44] (Section 2.2.1) and evolutionary reinforcement learning (ERL) [45,46] (Section 2.2.2).

2.1. Transfer learning

Transfer learning (TL), also known as knowledge transfer, enhances neural network learning in a target domain by leveraging external knowledge [43]. In supervised learning, TL is typically achieved by pre-training models on a source domain (e.g., ImageNet [47]) and fine-tuning them on the target domain [48]. While straightforward and effective, this approach may introduce redundant knowledge and risk negative transfer [34,36,37], especially in RL, where fine-tuning is ill-defined [49] and transferring knowledge across Markov decision process remains challenging [43].

Transfer techniques in RL primarily fall into policy transfer and representation transfer, distinguished by the type of knowledge being transferred [43]. Policy transfer reuses policies from source domains via policy reuse [50,51] or policy distillation [52–54]. Representation transfer exploits knowledge encoded in neural networks, either by reusing pre-trained representations [49,55] or disentangling them to extract reusable components [56,57]. Learngenes fall under representation transfer, condensing task-agnostic knowledge into neural network fragments and reusing these representations to facilitate adaptation to novel tasks.

Typical representation transfer methods, such as progressive neural networks [55], retain all trained policy networks and integrate them into new networks via collateral connections. However, this approach incurs high computational overhead and memory consumption due to the continuously growing network structure. Other approaches, such as those proposed in [49], decompose policy networks into task-specific and agent-specific modules for recombination, yet their effectiveness declines when applied to unseen task-agent pairs. Similarly, GRUSM [55] reuses existing neural modules while introducing new structures for target tasks. However, these methods rely on task-specific architectures, requiring recombination or structural expansion when encountering new tasks. This reliance complicates transfer, reduces efficiency, and limits generalization to unseen scenarios.

In contrast, learngenes provide a more universal and efficient knowledge transfer mechanism by encapsulating task-agnostic knowledge into compact, inheritable neural fragments. These fragments (i.e., learngenes) can be seamlessly integrated into target policy networks, facilitating the initialization of new agents across diverse tasks. This approach minimizes task-specific dependencies and structural complexity, offering a scalable, efficient, and generalizable method for knowledge transfer in RL.

2.2. Evolutionary algorithms

Evolutionary algorithms (EAs) are gradient-free optimization methods inspired by natural evolution [29–31]. Rooted in Darwinian principles, they refine a population of candidate solutions through fitness-based selection, mutation, crossover, and survival of the fittest [58], enabling effective solutions to complex optimization problems [45,59].

Thus, EAs inherently introduce concepts analogous to "genes", represented by "genomes" or "chromosomes" [60–65]. However, in EAs, these "genomes" or "chromosomes" are merely ***encodings of candidate solutions*** for optimization rather than carriers of inheritable knowledge. Typically, they are expressed as real-valued vectors in evolution strategies (ES) [66] and differential evolution (DE) [67], or as binary strings in genetic algorithms (GA) [68].

In contrast, the "learngenes" introduced in this study fundamentally differ from the "genomes" in traditional EAs. Rather than encoding candidate solutions, learngenes function as ***carriers of inheritable knowledge*** among agents, aligning with the principles of transfer learning. Their evolution follows Lamarckian principles, where task-agnostic knowledge is inherited and accumulated across generations, enabling efficient adaptation to new tasks.

2.2.1. Neuroevolution

Neuroevolution applies EAs to optimize neural networks, including parameters, hyperparameters, and architectures [69–72]. These components to be optimized are encoded as "genomes", using direct representations like binary strings in NEAT [61] or indirect representations such as pattern-producing networks in CPPN [73] and DPPN [74]. Despite applying evolution to neural networks, neuroevolution remains a search-driven approach focused on network optimization rather than enabling knowledge transfer [55,74–76]. In contrast, GRL constructs a genetic mechanism within agents, condensing task-agnostic knowledge into learngenes to facilitate efficient knowledge transfer.

Methods such as [77] and [78] integrate transfer learning with EAs to select transferable layers in pre-trained models. However, they rely solely on binary encodings for layer selection, with pre-trained model parameters remaining frozen throughout the process. In contrast, GRL follows Lamarckian evolution, applying the principle of survival of the fittest to ensure that superior learngenes are

inherited and refined over generations. This enables the progressive accumulation of task-agnostic knowledge in learngenes, with parameters continuously updated via gradient descent [79] rather than EAs, ensuring a more effective and adaptable knowledge transfer mechanism.

2.2.2. Evolutionary reinforcement learning (ERL)

RL faces persistent challenges, including limited exploration [80,81], poor convergence [82,83], and sensitivity to hyperparameters [84,85], which hinder its application to complex problems [45,46]. To overcome these limitations, researchers integrate EAs with RL, leveraging their robust exploration and global optimization capabilities for efficient parameter search [86–88], action selection [89–92], and hyperparameter tuning [93–95].

Among these methods, ERL [96] pioneers the integration of EAs for policy optimization, leveraging a population of agents to enhance exploration and improve experience diversity, particularly in offline training. Several works have extended ERL to enhance its effectiveness. CERL [97] integrates multiple RL agents with diverse hyperparameters to enrich the learning process. PDERL [98] mitigates catastrophic forgetting via Proximal-Distilled GA, while CHDRL [99] enhances sample efficiency using global agents. Other extensions, including SC [100], CSPS [99], and T-ERL [101], similarly focus on diversifying experiences to improve exploration.

Beyond enhancing exploration through agent populations, EAs are also employed for the direct optimization of network parameters in RL. SupeRL [87] enhances policy optimization by perturbing the current RL policy to initialize a population of policies. Similarly, VFS [88] applies this approach to value networks, generating a population of perturbed value networks for more accurate approximations.

Instead of optimizing the ***training process*** of policy networks through population-driven exploration or global search, learngenes ***directly initialize*** agents' policy networks with accumulated task-agnostic knowledge. Unlike the inheritance of building blocks in Genetic Programming [102,103], this mechanism enables direct knowledge transfer without relying on evolutionary operators such as crossover or mutation, positioning GRL as an initialization-driven rather than search-driven approach. As a result, evolution in GRL is a ***one-time process***—once learngenes have fully encapsulated task-agnostic knowledge, they provide a reusable initialization for agents across different tasks, unlike ERL-based methods, which necessitate ***repeated evolution*** to optimize policy training for each new task.

3. Methods

In this section, we first review reinforcement learning and the PPO algorithm as the foundation for agent training. We then introduce learngene, outlining its definition, structure, and mechanism for facilitating knowledge transfer. Finally, we present the Genetic Reinforcement Learning (GRL) framework, illustrating how genetic mechanisms are integrated into agents to promote the encapsulation of task-agnostic knowledge within learngenes.

3.1. Preliminary

3.1.1. Reinforcement learning

Reinforcement learning (RL) trains an agent to interact with an environment and maximize a cumulative reward [104–106]. This process is typically formalized as a Markov decision process (MDP) [107], represented by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively. At each time step t, the agent selects an action $a_t \sim \pi(s_t) \in \mathcal{A}$ according to a policy π based on the current state $s_t \in \mathcal{S}$. The environment then transitions to a new state s_{t+1} following the transition probability $P(s_{t+1}|s_t, a_t)$, and the agent receives a reward $r_t = R(s_t, a_t)$.

The objective of RL is to learn an optimal policy π^* that maximizes the expected return G_t, defined as the discounted cumulative reward:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor that determines the significance of future rewards.

3.1.2. Proximal policy optimization (PPO)

PPO [108] is a widely used on-policy Actor-Critic algorithm [109,110] that simultaneously trains two neural network models: a policy model π_θ (i.e., actor) and a value model V_ϕ (i.e., critic).

The **policy model** $\pi_\theta(a|s)$, parameterized by θ, represents the agent's stochastic policy and is optimized using a surrogate objective function based on the advantage function $\hat{A}_t$. The **value model** $V_\phi(s)$, parameterized by ϕ, approximates the expected return from a given state and serves as *a baseline for evaluating actions*. The advantage function $\hat{A}_t$, which measures the relative improvement of an action over this baseline, is typically estimated as:

$$\hat{A}_t = G_t - V_\phi(s_t) \tag{2}$$

where G_t is the empirical return. To balance exploration and exploitation while reducing variance and improving stability, PPO estimates the advantage function $\hat{A}_t$ using Generalized Advantage Estimation (GAE), defined as:

$$\hat{A}_t = \sum_{k=0}^{\infty} (\gamma\lambda)^k \delta_{t+k} \tag{3}$$

where $\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$ is the temporal difference (TD) error, γ is the discount factor (as in Eq. (1)), and λ is a hyperparameter controlling the trade-off between bias and variance.

Thus, the PPO algorithm optimizes two complementary objectives: the policy objective and the value objective. The **policy objective** maximizes a clipped surrogate function to constrain policy updates within a stable range, ensuring training stability by limiting overly large policy updates, which is defined as:

$$\mathcal{J}_\pi(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{4}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies, and ϵ defines the clipping threshold. This mechanism prevents excessively large updates to the policy, ensuring stable learning.

The **value objective** aims to train the value model $V_\phi(s_t)$ to accurately predict the empirical return G_t, using a mean squared error (MSE) objective:

$$\mathcal{J}_V(\phi) = \hat{\mathbb{E}}_t[-(V_\phi(s_t) - G_t)^2] \tag{5}$$

PPO alternates between optimizing the policy and value functions to improve action selection and ensure accurate state-value estimates, thereby promoting stability and efficiency in training.

3.2. Learngene

Inspired by genes in nature, learngene functions as a carrier of inheritable knowledge in neural networks. We first present its general definition and structural properties, and then describe its instantiation for agents in reinforcement learning control tasks.

3.2.1. Definition of the learngene

Genes in nature encode and transfer inheritable traits or survival knowledge that is environmental-agnostic, enhancing the adaptation and survival of descendants across diverse living environments. Similarly, neural networks possess analogous task-agnostic knowledge, typically represented as foundational principles or general patterns that can be shared across diverse domains or tasks [50,52,111–113]. Researches have further shown the transferability of specific network modules across domains [49,55,76,114,115], highlighting the potential for network fragments to encapsulate and transfer such task-agnostic knowledge. This paves the way for designing a "genetic mechanism" in machines that leverages shared task-agnostic knowledge and transferable network fragments to enhance generalization and efficiency in novel domains or tasks.

To construct such a "genetic mechanism" in machines, we introduce the concept of the "learngene", a network fragment specifically designed to encapsulate task-agnostic knowledge across diverse tasks. For a neural network with parameters Φ, a learngene is defined as compact fragments $\Phi_{\mathcal{F}}$, where $\mathcal{F}$ represents the structural form of learngenes. These fragments typically consist of fundamental building blocks in neural networks, such as layers in fully connected networks, kernels in convolutional neural networks, or blocks in transformers.

Learngenes enable networks to inherit encapsulated task-agnostic knowledge, thereby enhancing their learning performance on novel tasks and domains. Specifically, the inheritance of learngenes in a newly initialized network with parameters Φ is expressed as:

$$\Phi = \Phi_{\mathcal{F}} + \Phi_\Delta \tag{6}$$

Here, $\Phi_{\mathcal{F}}$ represents the learngenes, encapsulating task-agnostic knowledge to facilitate efficient adaptation to new tasks, while Φ_Δ refers to the randomly initialized components, which capture task-specific knowledge tailored to individual task requirements, thereby maintaining a balance between generalization and specialization.

3.2.2. Learngenes in intelligent agents

In RL, an agent is controlled by a policy network π_θ, parameterized by θ, which is a mapping from state spaces $\mathcal{S}$ to action spaces $\mathcal{A}$ [116], serving as the agent's "brain". In this study, we extract learngenes from the agent's policy network (i.e., actor),[1] formally represented as:

$$g = \langle \alpha, \theta_{\mathcal{F}} \rangle \tag{7}$$

where g is the learngene extracted from the agent α, and $\mathcal{F}$ denotes the structural form of the learngene.

In this work, the policy network π_θ is modeled as a fully connected network with n_L layers. Thus, learngenes in agents are formally defined in terms of layers, where $\mathcal{F}$ represents the layer IDs that form the learngenes. For example, when $\mathcal{F} = \{i, j\}$, the learngene corresponds to the i-th and j-th layers of the agent's policy network, with parameters $\theta_{\{i,j\}}$.

[1] Experimental evidence in Section 5.1 demonstrates that the agent's value network (i.e., critic) lacks transferable task-agnostic knowledge.

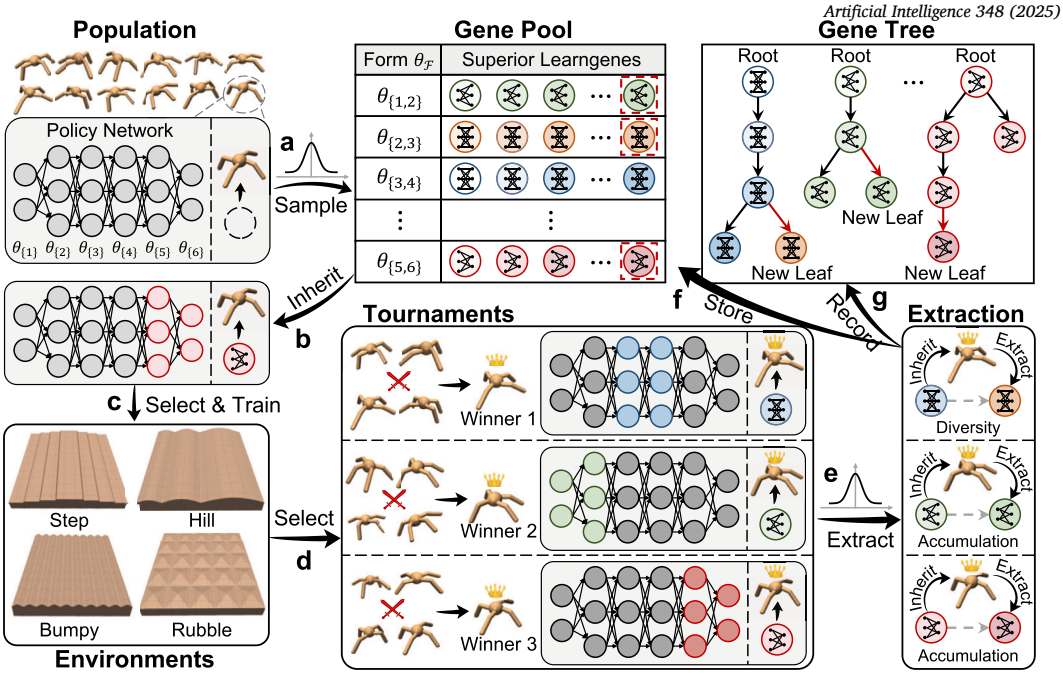


Fig. 2. GRL Overview. GRL is a computational framework designed to train agents and evolve learngenes across generations. In each generation, newborn agents select (**a**) and inherit (**b**) ancestral learngenes from the Gene Pool. Each agent is then assigned a specific training environment and undergoes reinforcement learning (**c**). After training, tournaments (**d**) are conducted to identify superior agents. Learngenes are then extracted from these superior agents (**e**) and stored in the Gene Pool (**f**), with their evolutionary relationships recorded in the Gene Tree (**g**). After updating the Gene Pool and Gene Tree, the generation concludes, and the process repeats with the initialization of a new population, enabling continuous refinement and accumulation of task-agnostic knowledge in learngenes.

In nature, innate processes are compressed into genes through a "genomic bottleneck" due to limited capacity [32]. Inspired by this, we introduce a similar bottleneck for learngenes in agents, compressing knowledge to retain only task-agnostic information. This compression is reflected in the number of layers n_l comprised in the learngenes, where $n_l < n_L$. We define this process of extracting and compressing task-agnostic knowledge into learngenes as ***knowledge condensation***.

Through knowledge condensation, learngenes encapsulate a compact set of task-agnostic knowledge that is highly transferable, enabling efficient adaptation to novel tasks and domains.

3.3. Genetic reinforcement learning (GRL)

To effectively perform knowledge condensation for encapsulating task-agnostic knowledge within learngenes, we propose Genetic Reinforcement Learning (GRL), a computational framework designed to construct a "genetic mechanism" within agents.

Unlike biological evolution and traditional evolutionary learning, both grounded in ***Darwinism***, GRL models the evolution of intelligent agents and the inheritance of learngenes based on ***Lamarckism***, as illustrated in Fig. 2. This Lamarckian genetic mechanism enables agents to inherit the task-agnostic knowledge from their ancestors, refine it on their tasks, and subsequently transfer the enhanced learngenes to their descendants for further optimization. Through generational inheritance of learngenes across diverse agents and tasks, GRL facilitates the progressive accumulation and refinement of task-agnostic knowledge, thereby improving the adaptability of agents over time.

3.3.1. Overview of GRL

Algorithm 1 outlines the GRL pipeline. Evolution starts with an initial population $\mathcal{P}$ of n_p agents, each trained on a randomly assigned task $\mathcal{T}_i$ (Section 3.3.2). After training all agents, superior agents $\alpha^\star$ are selected via a tournament-based process (Section 3.3.3). The scores of ancestral learngenes, stored in the Gene Pool and with kinship recorded in the Gene Tree (Section 3.3.4), are updated to ensure the continuity of learngenes associated with the superior agents.

The learngenes of the current generation are then extracted from the superior agents (Section 3.3.5), with the Gene Pool and Gene Tree updated accordingly to support their inheritance in future generations. In subsequent generations, a new population of agents is initialized by inheriting learngenes from the Gene Pool (Section 3.3.6).

This iterative evolutionary process facilitates the continuous transfer and refinement of learngenes, promoting the progressive accumulation of task-agnostic knowledge across generations. We next detail the training process for a single generation of agents across diverse tasks, which serves as the foundation of GRL and the starting point of evolution.

Algorithm 1 Genetic Reinforcement Learning (GRL).

Input: Training environments with m obstacles $\mathcal{E} = \{o_1, o_2, ..., o_m\}$, Total generations n_G, Population size n_p, Number of layers comprising learngenes n_l, Agents in a tournament ω

for $i_{\text{gen}} = 0$ to n_G **do**
  # Initialize population
  Generate a population of n_p agents $\mathcal{P} = \{\alpha_1, \alpha_2, \cdots, \alpha_{n_p}\}$.
  Initialize network parameters for each agent $\alpha_i \in \mathcal{P}$ randomly.
  for each agent $\alpha_i \in \mathcal{P}$ **do**
    if $i_{\text{gen}} \neq 0$ **then**
      # Inherit ancestral learngenes
      Sample an ancestral learngene from the Gene Pool based on the probability distribution defined in Eq. (22).
      Initialize α_i by inheriting the selected learngene.
    end if
    # Train agents
    Randomly select an obstacle from $\mathcal{E}$ to form a task $\mathcal{T}_i$.
    Train α_i on the task $\mathcal{T}_i$.
  end for
  # Extract learngenes from superior agents
  Perform a tournament-based competition and select superior agents $\mathcal{P}^\star = \{\alpha_1^\star, \alpha_2^\star, ..., \alpha_\kappa^\star\}$, where $\kappa = \lceil \frac{n_p}{\omega} \rceil$.
  Update scores of relevant ancestral learngenes using Eq. (16).
  Extract learngenes $g_i = \langle \alpha_i^\star, \theta_{\mathcal{F}} \rangle$ from each superior agent $\alpha_i^\star \in \mathcal{P}^\star$ based on Eq. (21).
  # Update Gene Pool and Gene Tree
  Add extracted learngenes to the Gene Tree as new leaf nodes.
  Update the Gene Pool by adding newly extracted learngenes and removing learngenes with the lowest scores.
end for

3.3.2. Training a generation of agents

The agents' living environment $\mathcal{E}$ is constructed using TRAVERSE tasks (Fig. 4a), incorporating m diverse training obstacles, $\mathcal{E} = \{o_1, o_2, \cdots, o_m\}$. Each generation includes a population of n_p agents, denoted as $\mathcal{P} = \{\alpha_1, \alpha_2, \cdots, \alpha_{n_p}\}$. Each agent α_i randomly selects an obstacle from $\mathcal{E}$ to form its task environment $\mathcal{T}_i$, in which it must traverse through the obstacles to reach the finishing line (Fig. 4c).

At each time step t, the velocity vector $\mathbf{v}_t$ can be decomposed as $\mathbf{v}_t = \mathbf{v}_t^{\text{tgt}} + \mathbf{v}_t^{\text{oth}}$, where $\mathbf{v}_t^{\text{tgt}}$ is the component directed toward the finishing line, and $\mathbf{v}_t^{\text{oth}}$ represents lateral components. In TRAVERSE, only $\mathbf{v}_t^{\text{tgt}}$ contributes to the reward, as lateral movements (i.e., $\mathbf{v}_t^{\text{oth}}$) are irrelevant to task progress. Thus, at each time step t, the reward r_t is calculated as:

$$r_t = \mu v_t^{\text{tgt}} - \nu ||a_t||^2 \tag{8}$$

where $v_t^{\text{tgt}} = |\mathbf{v}_t^{\text{tgt}}|$, a_t is the action taken by the agent (see Section 4.1 and Appendix D for details), and μ and ν are balance weights. The second term penalizes excessively large actions to encourage efficiency.

The total reward for an episode, r_{ep}, is the cumulative sum of rewards r_t over all time steps within an episode T_{ep}[2]:

$$r_{\text{ep}} = \sum_{t=0}^{T_{\text{ep}}} r_t + \lambda r_{\top} \tag{9}$$

where $r_{\top}$ is the terminal reward, and $\lambda = 1$ if the agent reaches the finishing line, and $\lambda = 0$ otherwise.

Similarly, the reward r_{it} for a training iteration, which includes one complete round of experience collection and policy updates, is computed over T_{it} time steps[3]:

$$r_{\text{it}} = \sum_{t=0}^{T_{\text{it}}} r_t + n_{\top} r_{\top} \tag{10}$$

where $n_{\top}$ is the number of agents reaching the finishing line within T_{it} time steps. Unless otherwise specified, "rewards" in all experiments refer to r_{ep}. After all agents complete training on their respective tasks, GRL evaluates the performance of each agent to identify superior ones whose learngenes are subsequently inherited by future generations.

3.3.3. Selecting superior agents

GRL selects superior agents $\alpha^\star$ based on their fitness, under the premise that higher fitness indicates greater adaptability, derived from more effective task-agnostic knowledge inherited from their ancestors. That is, such a selection process targets the identification

[2] T_{ep} is influenced by the agent's policy—typically, a better policy leads to fewer T_{ep}.
[3] T_{it} is fixed for each training iteration to ensure consistency in the duration of experience collection and policy updates.

of higher-quality ancestral learngenes, thereby facilitating their continual refinement and progressive accumulation of task-agnostic knowledge across generations.

For each agent α_i, GRL calculates its fitness f_i as the average reward r_{it} over all training iterations, reflecting its overall performance:

$$f_i = \frac{\sum_{\text{it}=1}^{n_{\text{it}}} r_{\text{it}}}{n_{\text{it}}} + \zeta \tag{11}$$

where n_{it} is the total number of training iterations, and ζ is a constant ensuring that fitness remains positive.

To ensure fair comparison across tasks with different obstacles, agent fitness is normalized based on obstacle difficulty, determined by the fitness of agents assigned the same obstacle in the current generation. Let $\mathcal{I}$ be the fitness set of all agents in $\mathcal{P}$, and $\mathcal{I}^{(o)}$ be the subset for agents assigned obstacle o. For an agent α_i with fitness f_i on a task involving obstacle o, the normalized fitness $\tilde{f}_i$ is calculated using min-max normalization within $\mathcal{I}^{(o)}$, scaled by the mean fitness of all agents:

$$\tilde{f}_i = \frac{f_i - f_{\min}^{(o)}}{f_{\max}^{(o)} - f_{\min}^{(o)}} \cdot \bar{f} \tag{12}$$

where $f_{\min}^{(o)}$ and $f_{\max}^{(o)}$ are the minimum and maximum fitness in $\mathcal{I}^{(o)}$, respectively. $\bar{f} = \frac{1}{n_p}\Sigma_{f\in\mathcal{I}} f$ represents the mean fitness of all agents in the current generation.

GRL selects superior agents through a tournament-based process [30,117]. In each tournament, ω agents are randomly chosen from $\mathcal{P}$ (without replacement), and the one with the highest normalized fitness wins. The set of superior agents, $\mathcal{P}^\star = \{\alpha_1^\star, \alpha_2^\star, ..., \alpha_\kappa^\star\}$, consists of $\kappa = \lceil \frac{n_p}{\omega} \rceil$ tournament winners, where $\lceil\cdot\rceil$ is the ceiling function, ensuring that all agents are included, even if n_p is not divisible by ω. Once superior agents are identified, GRL enhances the continuity of their ancestral learngenes, a process detailed in the following section.

3.3.4. Promoting survival of superior ancestral learngenes

GRL preserves the continuity of corresponding ancestral learngenes by increasing the probability of their inheritance in subsequent generations. This mechanism ensures the continuous accumulation of task-agnostic knowledge, akin to how acquired traits are preserved and refined across generations in Lamarckian evolution [118,119].

To facilitate this process, GRL introduces two key structures essential for learngene evolution: the **Gene Pool**, which stores superior ancestral learngenes, and the **Gene Tree**, which tracks kinship relationships among learngenes throughout evolution.

- **Gene Pool (GP)** is a data structure designed to store superior learngenes, categorized by their structural forms $\mathcal{F}$, as shown in Fig. 2. Each form $\mathcal{F}$ corresponds to a specific combination of n_l layers, with $C_{n_L}^{n_l}$ possible combinations in the GP,[4] and each form contains ρ learngenes, resulting in a total of $C_{n_L}^{n_l} \cdot \rho$ learngenes. Specifically, when $n_l = 2$ and $n_L = 6$, the GP can be represented as $\text{GP} = \{\mathcal{G}_{\{1,2\}}^{\text{GP}}, \mathcal{G}_{\{1,3\}}^{\text{GP}}, ..., \mathcal{G}_{\{i,j\}}^{\text{GP}}, ..., \mathcal{G}_{\{5,6\}}^{\text{GP}}\}$, where $\mathcal{G}_{\{i,j\}}^{\text{GP}}$ is the set of learngenes with the same form $\mathcal{F} = \{i, j\}$:

$$\mathcal{G}_{\{i,j\}}^{\text{GP}} = \{g_k | g_k = \langle \cdot, \theta_{\{i,j\}} \rangle, k \in [1, \rho]\} \tag{13}$$

- **Gene Tree (GT)** is a data structure that tracks kinship relationships among learngenes throughout evolution, as shown in Fig. 2. Each GT originates from a root node representing a learngene extracted in the initial generation and expands over subsequent generations by appending newly extracted learngenes as leaf nodes. Similar to GP, there are $n_s = C_{n_L}^{n_l} \cdot \rho$ GTs in total, forming a gene forest $\text{GF} = \{\mathcal{G}_1^{\text{GT}}, \mathcal{G}_2^{\text{GT}}, ..., \mathcal{G}_i^{\text{GT}}, ..., \mathcal{G}_{n_s}^{\text{GT}}\}$. Each GT $\mathcal{G}_i^{\text{GT}}$ is structured as binary parent-child relationships $(g^{(p)}, g^{(c)})$:

$$\mathcal{G}_i^{\text{GT}} = \{(g^{(p)}, g^{(c)}) \mid g^{(p)} = \langle \alpha^{(p)}, \theta_{\mathcal{F}^{(p)}} \rangle, g^{(c)} = \langle \alpha^{(c)}, \theta_{\mathcal{F}^{(c)}} \rangle, \alpha^{(c)} \text{ inherits } g^{(p)}\} \tag{14}$$

  The path length between two nodes indicates the number of generations between learngenes, reflecting their evolutionary relationship.

To quantify the effectiveness of each learngene, a score s_g is assigned to every learngene in the GP. This score evaluates learngene quality based on two key factors: the ***initial value***, reflecting the performance of the superior agent from which it is extracted, and the ***continuity value***, assessing its impact on their descendant agents.

For a learngene $g = \langle \alpha^\star, \theta_{\mathcal{F}} \rangle$, the initial value of s_g incorporates both its structural efficiency and the performance of the agent $\alpha^\star$ from which it is extracted, defined as:

$$s_g = \frac{f}{\sum_{i\in\mathcal{F}} \mathcal{W}_{\text{eff}}(\theta_{\{i\}})} \tag{15}$$

where f is the fitness of $\alpha^\star$, and $\mathcal{W}_{\text{eff}}(\theta_{\{i\}}) = \sqrt{|\theta_{\{i\}}|}$ calculates the effective layer width of the i-th layer, with $|\cdot|$ denoting the number of parameters.

[4] Here, $C_{n_L}^{n_l}$ is the binomial coefficient, calculated as $C_{n_L}^{n_l} = \frac{n_L!}{n_l!(n_L - n_l)!}$.

The continuity value of ancestral learngenes associated with $\alpha^\star$ is then updated via the GT. Specifically, this update process starts from the node of the learngene g_* inherited by $\alpha^\star$ and backtracks along its lineage to the root node (Fig. 3a). For each parent-child pair $(g^{(p)}, g^{(c)})$ in the update path, the score of the ancestral learngene $g^{(p)}$ is incremented by the continuity value:

$$s_{g^{(p)}} \leftarrow s_{g^{(p)}} + \sigma(g^{(p)}, g^{(c)})\eta^{\epsilon+1}\tilde{f} \tag{16}$$

where $\tilde{f}$ is the normalized fitness of $\alpha^\star$ defined in Eq. (12), η is the parental decay coefficient, and ϵ is the path length between $g^{(p)}$ and g_*. $\sigma(\cdot,\cdot)$ quantifies the similarity between the structural forms of two learngenes and is defined as:

$$\sigma(g^{(p)}, g^{(c)}) = \frac{\sum_{i \in \mathcal{F}^{(p)} \cap \mathcal{F}^{(c)}} \mathcal{W}_{\text{eff}}(\theta_{\{i\}})}{\sum_{i' \in \mathcal{F}^{(p)} \cup \mathcal{F}^{(c)}} \mathcal{W}_{\text{eff}}(\theta_{\{i'\}})} \tag{17}$$

where $\sigma(g_*, \varnothing) = 1$.

To gradually eliminate early ancestral learngenes and promote learngene diversity [120–122] as well as task-agnostic knowledge accumulation, learngene scores decay proportionally by β after each generation:

$$s_g \leftarrow \beta \cdot s_g \tag{18}$$

After updating the scores of ancestral learngenes to reflect their evolutionary advantages and continuity, the next step focuses on extracting new learngenes from the superior agents of the current generation.

3.3.5. Extracting learngenes from the current generation

New learngenes are extracted from the superior agents of the current generation $\mathcal{P}^\star$, with the GP and GT updated to support their inheritance in future generations. For a superior agent $\alpha^\star$ inheriting the ancestral learngene with the structural form $\mathcal{F}^{(p)}$, the most direct way to ensure task-agnostic knowledge accumulation is to extract new learngenes with the same structural form as the inherited one.

However, to promote diversity, the learngene extraction process incorporates randomness by sampling structural forms based on form probability, allowing inferior structural forms a chance to contribute to evolution (Fig. 3b). The form probability $p_{\mathcal{F}}$ for a structural form $\mathcal{F}$ is defined as:

$$p_{\mathcal{F}} = \frac{s_{\mathcal{F}}}{\sum_{\mathcal{F}' \in \text{GP}} s_{\mathcal{F}'}} \tag{19}$$

where $s_{\mathcal{F}}$ is the total score of learngenes with structural form $\mathcal{F}$ in $\mathcal{G}^{\text{GP}}_{\mathcal{F}}$:

$$s_{\mathcal{F}} = \sum_{g \in \mathcal{G}^{\text{GP}}_{\mathcal{F}}} s_g \tag{20}$$

To balance continuity and diversity, the inherited structural form $\mathcal{F}^{(p)}$ is also incorporated into the extraction process by adjusting the final extraction probability $\tilde{p}_{\mathcal{F}}$ as:

$$\tilde{p}_{\mathcal{F}} = \frac{\mathbb{1}_{\mathcal{F}^{(p)}}(p_{\mathcal{F}})}{\sum_{\mathcal{F}' \in \text{GP}} \mathbb{1}_{\mathcal{F}^{(p)}}(p_{\mathcal{F}'})} \tag{21}$$

where $\mathbb{1}_{\mathcal{F}^{(p)}}(p_{\mathcal{F}}) = 1$ if $\mathcal{F} = \mathcal{F}^{(p)}$, and $\mathbb{1}_{\mathcal{F}^{(p)}}(p_{\mathcal{F}}) = p_{\mathcal{F}}$ otherwise. This adjustment favors the selection of learngenes with the same structural form as $\mathcal{F}^{(p)}$, thereby encouraging the accumulation of task-agnostic knowledge while preserving diversity through the exploration of alternative forms (Fig. 3c).

Then, the newly extracted learngenes replace lower-scoring ones in the GP and are added to the GT as leaf nodes. To maintain the stability of GP, at most two learngenes are replaced per structural form in each generation.

3.3.6. Inheriting learngenes for a new generation

After extracting learngenes and updating the GP and GT, the current generation concludes. The next generation starts by initializing a new population, where each agent selects and inherits an ancestral learngene from the GP.

The learngene to be inherited is selected via weighted sampling based on both form probability and individual learngene scores, with the selection probability of $g = \langle \cdot, \theta_{\mathcal{F}} \rangle$ in the GP given by:

$$p_g = p_{\mathcal{F}} \cdot \frac{s_g}{s_{\mathcal{F}}} \tag{22}$$

This approach preserves diversity by ensuring that learngenes with lower scores or inferior structural forms still have a chance to contribute to the next generation.

Then, the selected learngenes are copied to the corresponding network layers of the agents, while the non-learngene parts remain randomly initialized (see Fig. 2 and Eq. (6)). Once all agents have inherited their learngenes, they undergo generational training (see Section 3.3.2), marking the start of a new evolutionary cycle.

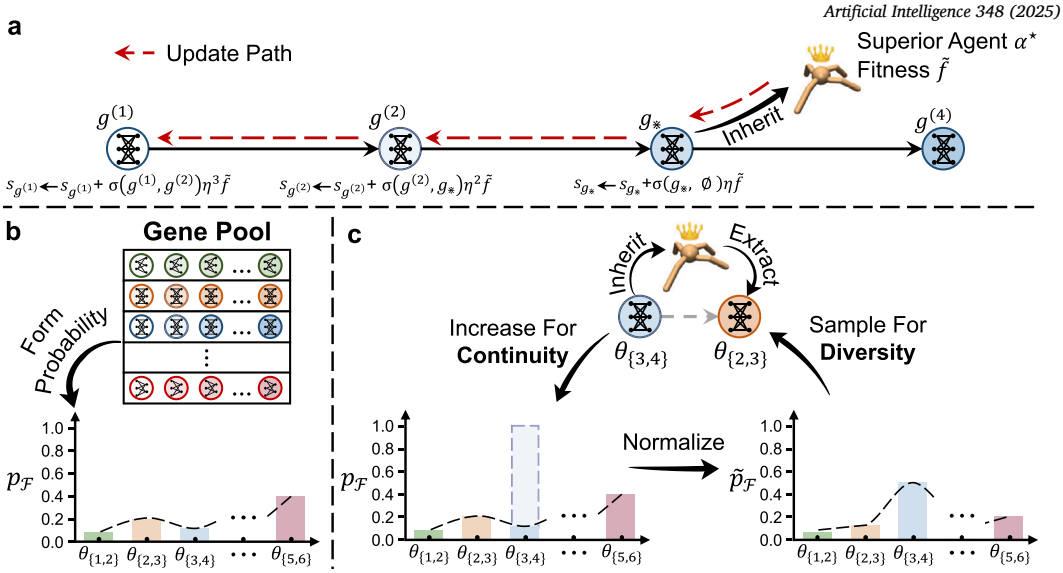


Fig. 3. Mechanism for Promoting the Survival of Ancestral Learngenes. a The Gene Tree tracks the kinship among learngenes, where $(g^{(i)}, g^{(i+1)})$ represents a binary parent-child relationship (Eq. (14)), and $s_{g^{(i)}}$ indicates the learngene score. After tournaments, the scores of ancestral learngenes associated with superior agents are updated to preserve their advantages. This update starts from the learngene inherited by each superior agent and backtracks to the root node, adjusting ancestral learngene scores along this path according to Eq. (16). **b** The Gene Pool stores superior learngenes for each structural form, promoting diversity. The form probability $p_{\mathcal{F}}$ for learngene extraction is calculated by Eq. (19). **c** The continuity of ancestral learngenes is reinforced during extraction. For example, when an agent α inherits a learngene of the form $\theta_{3,4}$, the corresponding form is consequently assigned a higher sampling probability to preserve continuity (Eq. (21)). Learngenes are then extracted via random sampling based on $\tilde{p}_{\mathcal{F}}$, as exemplified by extracting a learngene with form $\theta_{\{2,3\}}$ from $\alpha^\star$.

4. Experiments

4.1. Details of training agents

Training environments are constructed using the MuJoCo simulator [123]. The agent is modeled as a standard ant robot with a central torso and four legs, each consisting of two links connected by eight hinge joints. By applying torques to these hinges, the ant robot effectively coordinates its limbs to walk in a designated direction.

Agents are trained within an RL framework, optimized using Proximal Policy Optimization (PPO) [108] that concurrently updates a policy model (i.e., actor) and a value model (i.e., critic) (see Section 3.1.2 for details). Both the policy and value models are fully connected networks with six layers ($n_L = 6$) in our experiments, each hidden layer containing 48 neurons.

The policy and value models receive proprioceptive inputs, including position, orientation, joint angles, and velocities, from the MuJoCo simulator [123]. Observations are normalized using a running mean and standard deviation [108] to ensure training stability.[5] The policy model outputs torques applied to the agent's eight hinge joints (8 dimensions), determining its actions (see Appendix D for details on the action space). And the value model outputs a scalar estimate (1 dimension) of the expected return from a given state. Hyperparameters for training agents and evolving learngenes are listed in Table C.1 and C.2.

4.2. Task environments

TRAVERSE is the primary task, featuring eight obstacle types—four used for training agents and evolving learngenes, and four novel ones for learngene evaluation (Fig. 4a,b). Each task involves obstacles of the same kind covering over half of a 120×40 square meters (m^2) environment, with the rest remaining flat (Fig. 4c). Agents start at the starting line and complete the task by traversing the obstacles to reach the finishing line. These obstacles are designed with progressively increasing difficulty, with variations in height and slope introducing greater challenges as agents approach the finish.

When designing obstacles, we ensure that training obstacles (i.e., Step, Bumpy, Hill, and Rubble) maintain comparable difficulty while balancing similarity and diversity to effectively condense task-agnostic knowledge. For novel obstacles (i.e., Channel, Incline, Dune, and Square) used for evaluation, we maximize differences in obstacle structure and task difficulty from training obstacles to highlight the task-agnostic knowledge encapsulated in learngenes. To visualize obstacle relationships, we introduce the *knowledge transfer rate* (see Appendix B for details), which quantifies knowledge similarity across obstacles, as shown in heat maps of Fig. 4.

[5] For each newly initialized agent, the parameters for normalizing observations in PPO are reset, with the mean vector reinitialized to **0** and variance vector to **1**. This prevents shared normalization across tasks, eliminating potential biases or inter-task dependencies. Additionally, it ensures that the inherited knowledge comes exclusively from learngenes, allowing a more robust evaluation of policy transferability and generalization across diverse tasks.

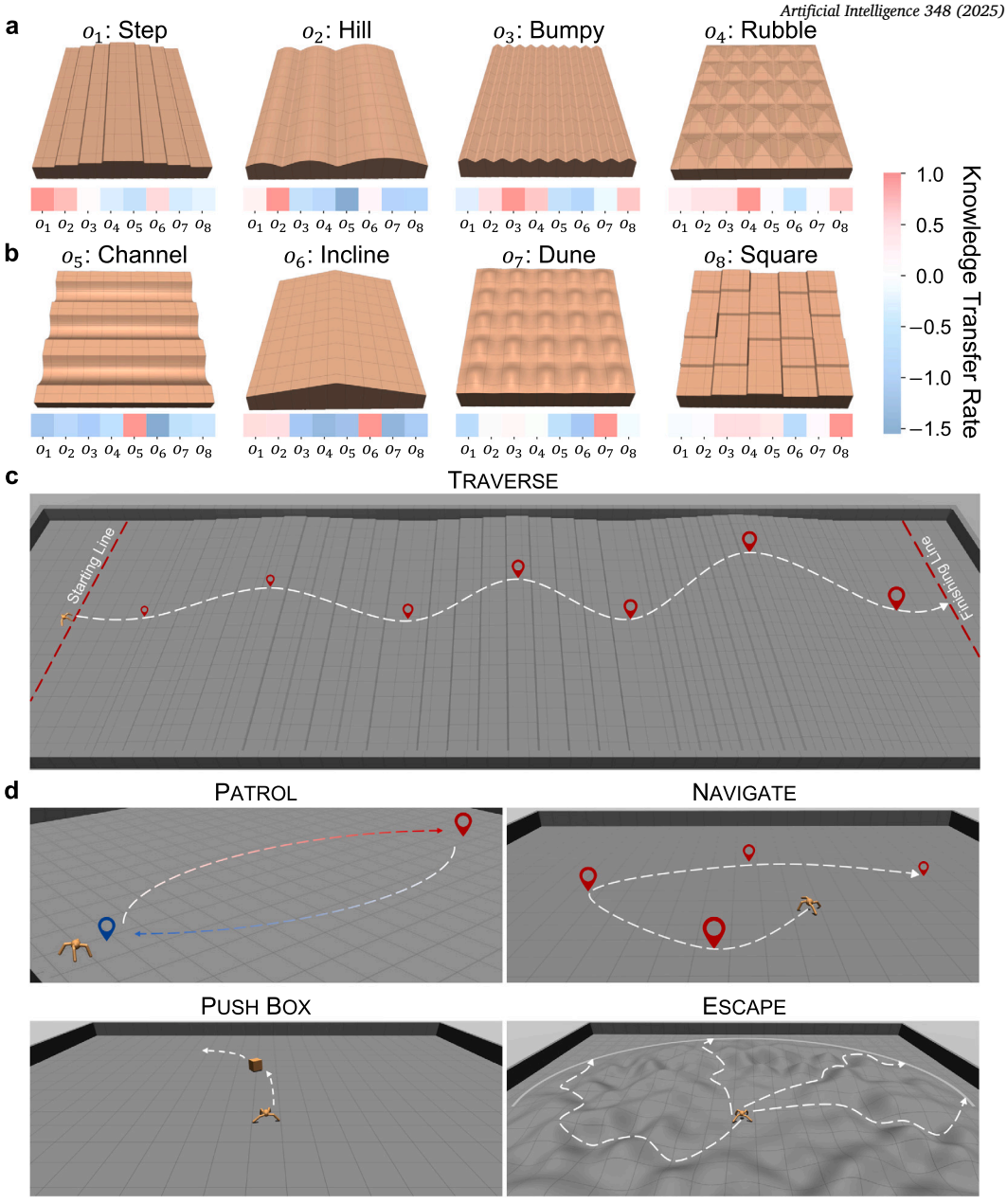


Fig. 4. Tasks for Training Agents and Evaluating Learngenes. TRAVERSE includes four training obstacles (**a**) for agent training and learngene evolution within GRL and four novel obstacles (**b**) for evaluating learngene transferability. The heat map for obstacle o_i depicts the knowledge transfer rate $\Omega_{j\to i}$, quantifying the extent to which knowledge acquired from o_j can be transferred to o_i, as defined in Eq. (B.1). Positive values (red) denote beneficial knowledge transfer from o_j to o_i, while negative values (blue) indicate counterproductive transfer. **c** Overview of TRAVERSE with Steps (i.e., o_1), where agents start from the starting line, traverse obstacles and reach the finishing line. **d** Beyond TRAVERSE, four additional tasks (i.e., PATROL, NAVIGATE, PUSH BOX, and ESCAPE) are introduced to evaluate agents' agility, stability, and manipulation capabilities, further highlighting the task-agnostic knowledge encapsulated in learngenes. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

In our experimental setup, task-agnostic knowledge includes foundational principles such as movement dynamics, reward acquisition, and avoidance of harmful factors, along with general patterns like torso-limbs coordination and basic locomotion strategies shared across obstacles.

To further illustrate the task-agnostic knowledge inherited from learngenes, we introduce four additional diverse tasks—PATROL, NAVIGATE, PUSH BOX, and ESCAPE [40] (see Fig. 4d for details). These tasks extend beyond TRAVERSE, challenging the agent's capabilities by altering its perception (i.e., inputs), objectives, and environmental interactions, which comprehensively evaluate the agent's agility, stability, and manipulation skills across a broader range of scenarios. Detailed task descriptions are provided in Appendix A.

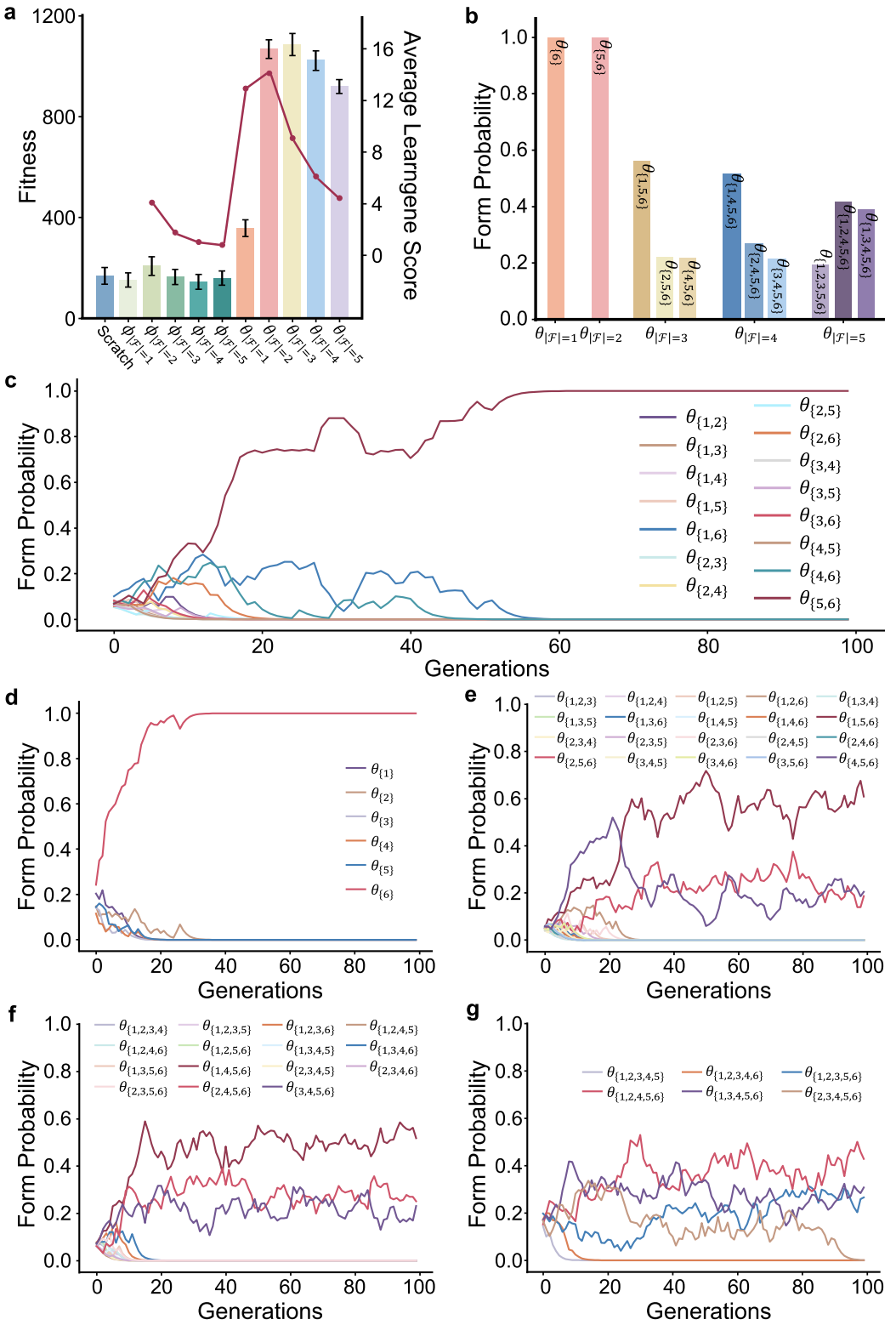


Fig. 5. Learngenes Extracted by GRL. a The mean and 95% bootstrapped confidence intervals of population fitness and corresponding learngene scores at the end of evolution (i.e., 100th generation), with learngenes composed of n_l layers from policy ($\theta_{|\mathcal{F}|=n_l}$) or value networks ($\phi_{|\mathcal{F}|=n_l}$). **b** Non-zero form probability of each learngene form at the end of evolution. **c** Change curves of form probabilities for all learngene forms in $\theta_{|\mathcal{F}|=2}$, along with $\theta_{|\mathcal{F}|=1}$ (**d**), $\theta_{|\mathcal{F}|=3}$ (**e**), $\theta_{|\mathcal{F}|=4}$ (**f**), and $\theta_{|\mathcal{F}|=5}$ (**g**).

5. Results

5.1. Evolving optimal learngenes in intelligent agents

In our experiments, each agent's policy network (i.e., actor) and value network (i.e., critic) consist of six layers ($n_L = 6$). Constrained by the "genomic bottleneck" for knowledge compression (Section 3.2.1), the number of layers forming learngenes must satisfy $n_l < n_L$ (i.e., $n_l \in [1, 5]$). Fig. 5a illustrates the average population fitness at the end of evolution, with learngenes composed of different numbers of layers from policy ($\theta_{|\mathcal{F}|=n_l}$) or value networks ($\phi_{|\mathcal{F}|=n_l}$).

Notably, transferring fragments of policy networks as learngenes significantly improves the overall fitness of the population, whereas value network fragments show negligible impact. This indicates that the policy network, which maps observations to actions [116], contains transferable "genetic materials" (i.e., inheritable knowledge). Accordingly, learngenes are identified as specific fragments of policy networks.

Furthermore, GRL dynamically extracts learngenes based on a distribution over various structural forms (Eq. (19)). As shown in Fig. 5b,c,d, only $\theta_{\{6\}}$ and $\theta_{\{5,6\}}$ exhibit stable inheritance throughout evolution, with their form probabilities reaching 1 while others drop to 0. This indicates that the knowledge encapsulated in $\theta_{\{6\}}$ and $\theta_{\{5,6\}}$ is ***essential*** across tasks. This essentiality is further evidenced by the observation that, when learngenes include additional layers ($n_l \in \{3, 4, 5\}$), only those containing $\theta_{\{5,6\}}$ retain their reproductive capability at the end of evolution (e.g., $\theta_{\{1,\mathbf{5,6}\}}$ and $\theta_{\{1,4,\mathbf{5,6}\}}$ in Fig. 5b).

In addition, incorporating additional layers inevitably introduces task-specific knowledge, thereby reducing overall commonality and increasing transfer costs. This is evident in Fig. 5a, where populations with $\theta_{|\mathcal{F}|=4}$ and $\theta_{|\mathcal{F}|=5}$ as learngenes perform worse than those with $\theta_{|\mathcal{F}|=2}$ and $\theta_{|\mathcal{F}|=3}$. Notably, $\theta_{\{5,6\}}$ achieves the highest average score (see $\theta_{|\mathcal{F}|=2}$ vs. $\theta_{|\mathcal{F}|=1}$ in Fig. 5a), striking the best balance between transfer effectiveness and cost.

Consequently, $\theta_{\{5,6\}}$ is identified as the optimal structural form of learngene. The learngene from the final generation's Gene Pool, with the form $\theta_{\{5,6\}}$ (i.e., $\mathcal{G}^{\mathrm{GP}}_{\{5,6\}}$), is designated as the optimal learngene $g^\star$, formally expressed as $g^\star = \langle \alpha^\star, \theta_{\{5,6\}} \rangle$, where $\alpha^\star$ represents the superior agent carrying $g^\star$. Hereafter, the term "learngene" refers exclusively to $g^\star$.

5.2. Inheriting innate abilities from learngenes

Innate behaviors in animals are encoded in their genes through evolution [124], enabling rapid adaptation to their environments with minimal interaction [17,18]. For instance, spiders inherently possess the ability to spin webs [15], and newborn colts can walk shortly after birth [16]. Inspired by this, we explore similar innate abilities in agents enabled by inheritable knowledge in evolved learngenes.

Fig. 6 visualizes the motion trajectories of newly initialized agents with and without learngenes. As observed in Fig. 6a, randomly initialized agents exhibit aimless exploration with large, uncoordinated actions, leading to substantial control costs without effective progress. These inefficiencies arise from the absence of prior knowledge, forcing agents to learn entirely from scratch.

In contrast, newborn agents inheriting learngenes (i.e., $g^\star$) exhibit markedly distinct behaviors. As shown in Fig. 6b and Video in Appendix E, these agents perform smaller, more efficient actions, incurring only 7% of the control costs of randomly initialized agents, while exhibiting straighter trajectories and greater forward movement toward the finishing line.

These innate abilities stem from the inheritable task-agnostic knowledge encapsulated within learngenes, condensed through evolution. As shown in Fig. 6c, learngenes constrain unnecessary exploration, allowing agents to achieve rewards immediately after initialization and adopt advantageous strategies early in training. Meanwhile, non-learngene components retain flexibility for exploration, enabling efficient task-specific adaptation while mitigating inductive bias from parameter transfer.

5.3. Enhanced learning ability via learngenes

Inheriting task-agnostic knowledge from learngenes strengthens agents' learning capabilities, particularly when encountering novel environments. Fig. 7 compares training curves of agents with learngenes against those learning from scratch across various tasks.

Agents inheriting learngenes demonstrate accelerated and superior learning performance on TRAVERSE across all four training obstacles from the outset of training (i.e., innate ability), reducing the time to an equivalent performance by at least 100 episodes. Even on novel obstacles unseen by ancestral agents, learngenes enable rapid adaptation, significantly outperforming agents learning from scratch. Notably, in TRAVERSE DUNE, agents with learngenes achieve higher rewards at birth than those learning from scratch throughout training.

To further validate the general learning capability enabled by learngenes, we transfer them to four tasks distinct from TRAVERSE, each featuring unique perceptions (i.e., inputs), objectives, and environmental interactions (details in Appendix A). Notably, learngenes consistently provide significant advantages across these tasks, confirming that the knowledge encapsulated in learngenes is sufficiently task-agnostic. For PUSH BOX and ESCAPE, where environmental feedback is modified to maximize divergence (e.g., encouraging large actions by reducing ν in Eq. (8)), agents with learngenes do not exhibit immediate advantages at the onset of training. However, they facilitate rapid adaptation to the environment, with their benefits becoming increasingly evident as training progresses.

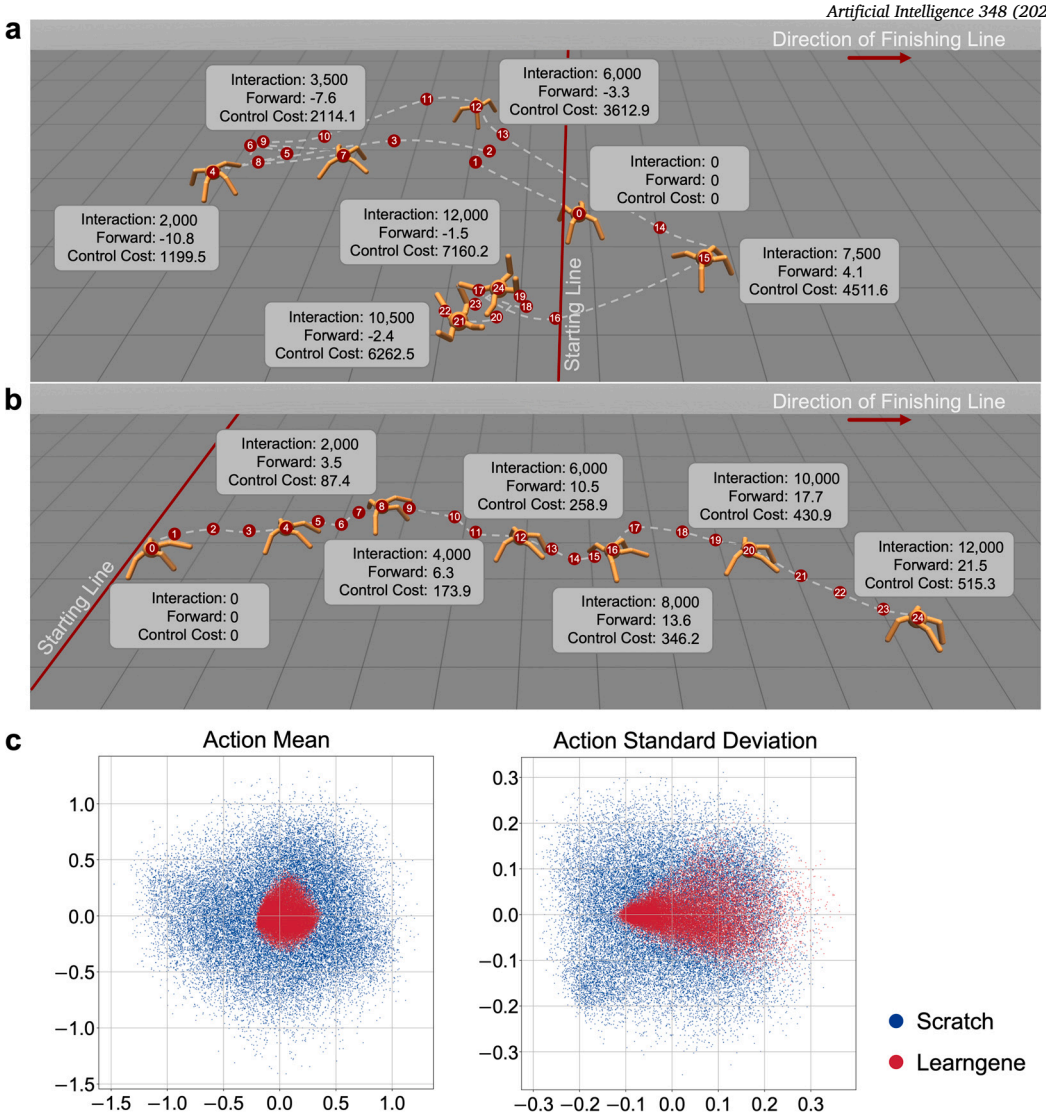


Fig. 6. Innate Abilities Enabled by Learngenes. Motion trajectories of newborn agents with (**a**) random initialization and (**b**) inheriting learngenes. Agents' locations are recorded every 500 interactions, shown as red dots labeled 0-24 in chronological order. Key metrics, including forward distance and control costs, are recorded at selected points. Notably, no parameter updates occur during these interactions. **c** We compare the actions (i.e., outputs of the policy network π_θ) of randomly initialized agents and those inheriting learngenes. A total of 30,000 actions are collected from agents after initialization and visualized via PCA. The inheritable knowledge encapsulated in learngenes effectively reduces unnecessary exploration, facilitating more efficient learning in the early stages.

5.4. Superiority of knowledge transfer in evolution

Evolutionary reinforcement learning (ERL) [96–98] expands the policy search space by concurrently evolving a population of agents, facilitating the discovery of diverse strategies and more robust behaviors than single-agent optimization. In contrast, learngenes prioritize the inheritance and accumulation of knowledge across generations. This distinction sets GRL apart from ERL algorithms, which combine evolution with RL but focus more on the search process rather than knowledge transfer. To underscore the benefits of task-agnostic knowledge transfer via learngenes, we compare GRL with ERL-based methods, as illustrated in Fig. 8.

Agents inheriting learngenes consistently outperform ERL-based methods across all tasks, demonstrating their effectiveness in accumulating task-agnostic knowledge through evolution and enabling efficient adaptation via inheritance. Unlike methods that rely solely on an evolutionary search to optimize actions or network parameters, learngenes encapsulate a higher-order form of "experience" beyond direct phenotypic expressions (e.g., actions). By condensing this experience into compact neural network fragments, learngenes facilitate more direct and universal knowledge transfer across diverse tasks.

Moreover, this transfer-based approach is more computationally efficient than search-based methods. Fig. 8d compares the computational costs of learngene and ERL-based methods across tasks. GRL's evolution process incurs a ***one-time*** cost, requiring only a single evolutionary run to condense knowledge, regardless of the number of downstream tasks. Once extracted, learngenes can be

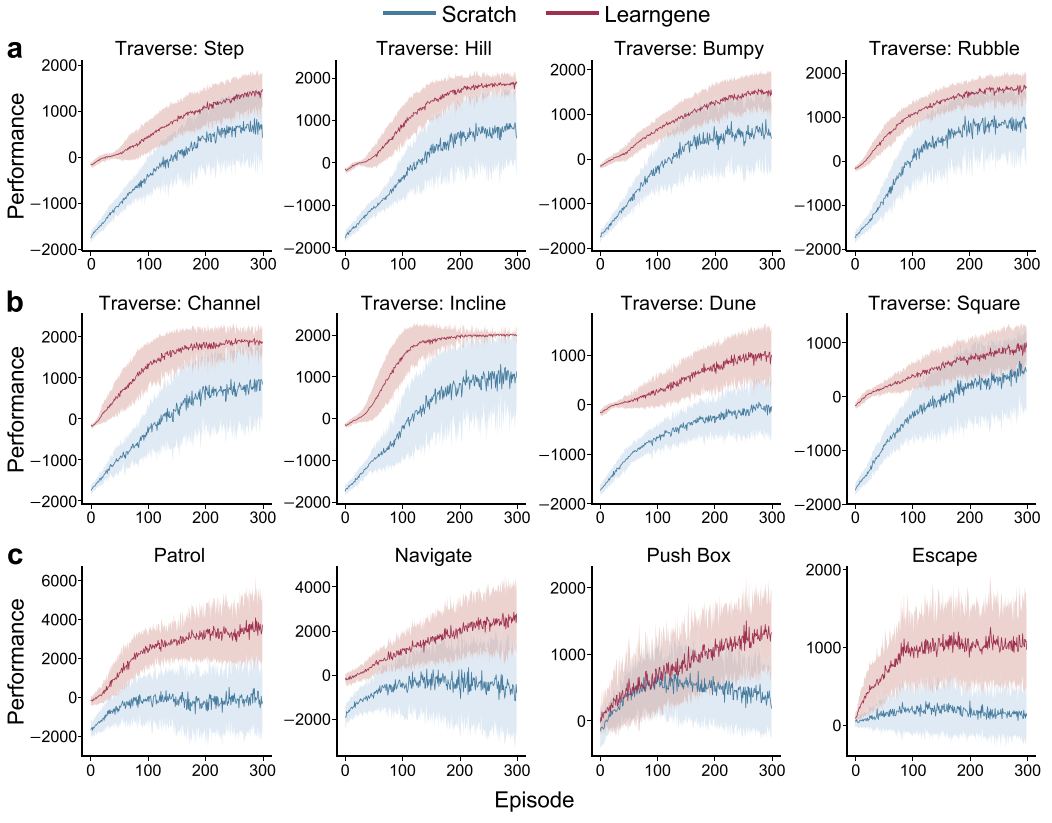


Fig. 7. Comparison with Learning from Scratch. Mean and standard deviation of rewards for agents trained on TRAVERSE with **(a)** training obstacles, **(b)** novel obstacles, and **(c)** other novel tasks, including PATROL, NAVIGATE, PUSH BOX and ESCAPE. Agents inheriting learngenes consistently outperform those learning from scratch across all tasks.

directly inherited, allowing agents to undergo standard training akin to traditional methods. In contrast, ERL-based algorithms must repeat the evolutionary search for each agent and task, making them significantly more time- and resource-intensive.

Consequently, the time complexity of ERL-related methods [96,98,125] is approximately $n_p^{(\mathrm{ERL})}$-times higher than that of learngene-based training or learning from scratch, where $n_p^{(\mathrm{ERL})}$ is the population size in ERL. Although Supe-RL [87] and VFS [88] employ evolution to optimize network updates rather than action search, they still incur substantial computational overhead, since each task requires an independent evolutionary process.

5.5. Essentiality of knowledge condensation through evolution

Learngenes enhance agents' learning ability by transferring condensed task-agnostic knowledge. While pre-trained models offer similar benefits and are widely explored in transfer learning [48,126,127], their application in RL remains limited. This limitation arises from task-specific variations in observations, rewards, and actions, making it challenging to develop a universal model that generalizes effectively across diverse RL tasks. To highlight the role of knowledge condensation, we pre-train an agent on TRAVERSE in a flat environment (i.e., no obstacles) to acquire knowledge, such as movement dynamics and reward acquisition. Fig. 9 compares training curves of agents inheriting learngenes versus those inheriting knowledge from pre-trained models across various tasks.

Pre-trained models transfer all policy network parameters, often leading to over-reliance on pre-trained policies for exploitation, which limits effective exploration in novel task environments. Consequently, their success depends largely on task similarity. For example, in TRAVERSE with bumpy obstacles, where reward acquisition and environmental observations align with pre-training, adaptation is rapid as the pre-trained strategy remains effective. However, in tasks with significant differences, such as NAVIGATE and PUSH BOX in Fig. 9, pre-trained policies can fail, sometimes performing worse than training from scratch. This underscores the risk of negative transfer [36,37], where prior knowledge hinders rather than aids learning.

We further explore knowledge transfer in pre-trained models using a learngene-like approach by transferring $\theta_{\{5,6\}}$ from the pre-trained model. As shown in Fig. 9, this mitigates negative transfer, allowing knowledge to remain effective even in tasks with substantial differences, such as PATROL, POINT NAVIGATE, or PUSH BOX. This indicates that in RL, performance may not improve due to excessive or inappropriate knowledge transfer. Moreover, it validates the effectiveness of the optimal structural form $\theta_{\{5,6\}}$, identified by GRL via population-based evolution. However, traditional pre-trained models lack the knowledge condensation process

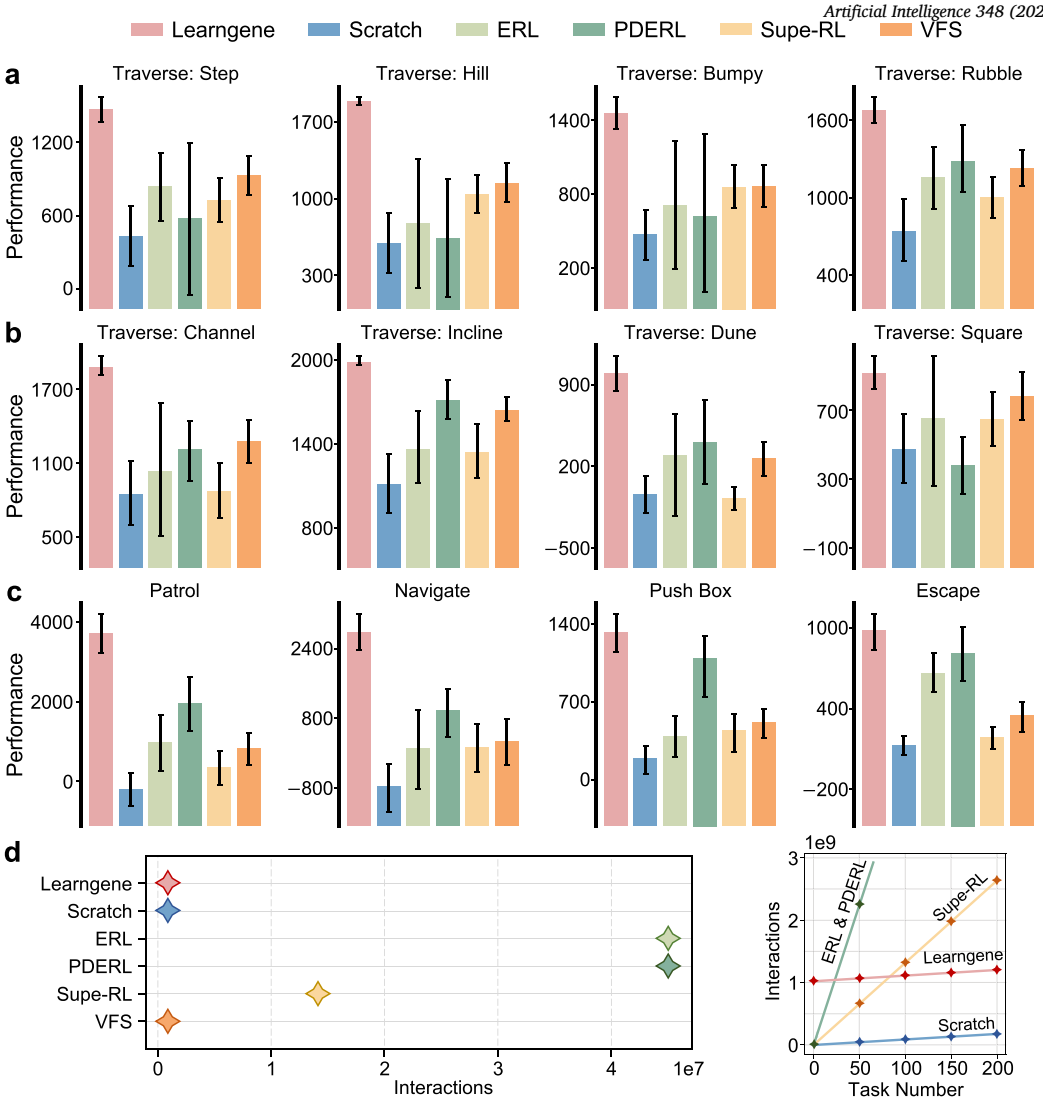


Fig. 8. Comparison with ERL-based Methods. Bars represent the average rewards at the end of training or evolution (i.e., 300th episode) on TRAVERSE (**a, b**) and other novel tasks (**c**), with error bars indicating 95% bootstrapped confidence intervals. **d** Computational cost per method, estimated by agent-environment interactions. The left shows the cost on a single downstream task, where agents initialized with learngenes incur only standard training with no additional evolutionary overhead, whereas ERL-based methods require repeated population-based searches. The right shows the total cost across multiple tasks, indicating that GRL's one-time evolutionary process yields an essentially constant cost as the number of tasks increases, whereas the cost of ERL-based methods grows linearly, underscoring GRL's efficiency in knowledge transfer and its reduced computational complexity.

in GRL, preventing $\theta_{\{5,6\}}$ from fully encapsulating task-agnostic knowledge. As a result, they still fall short of evolved learngenes, further underscoring the essential role of knowledge condensation.

5.6. Encapsulation of complete task-agnostic knowledge within learngenes

As defined in Section 3.2.2, the "genomic bottleneck" (i.e., $n_l < n_L$) establishes a "boundary" within the network, ensuring that task-agnostic knowledge is encapsulated within learngenes, while task-specific knowledge remains in non-learngene components. To validate the completeness of task-agnostic knowledge encapsulated in learngenes, we apply different initialization methods to all parameters of agents learning from scratch and the non-learngene parts of agents inheriting learngenes (Fig. 10).

Regardless of the initialization method, agents with learngenes consistently achieve comparable environmental adaptability. In contrast, agents learning from scratch are highly sensitive to initialization, with significant performance variance across different initialization methods, highlighting the strong dependency of learning and adaptability on parameter settings. The insensitivity of non-learngene parts to initialization confirms that the task-agnostic knowledge encapsulated in learngenes provides the minimal essential initialization for agents' networks. Once learngenes initialize the essential network components, the initialization of non-learngene parts (including value networks) has minimal impact on the agents' learning process.

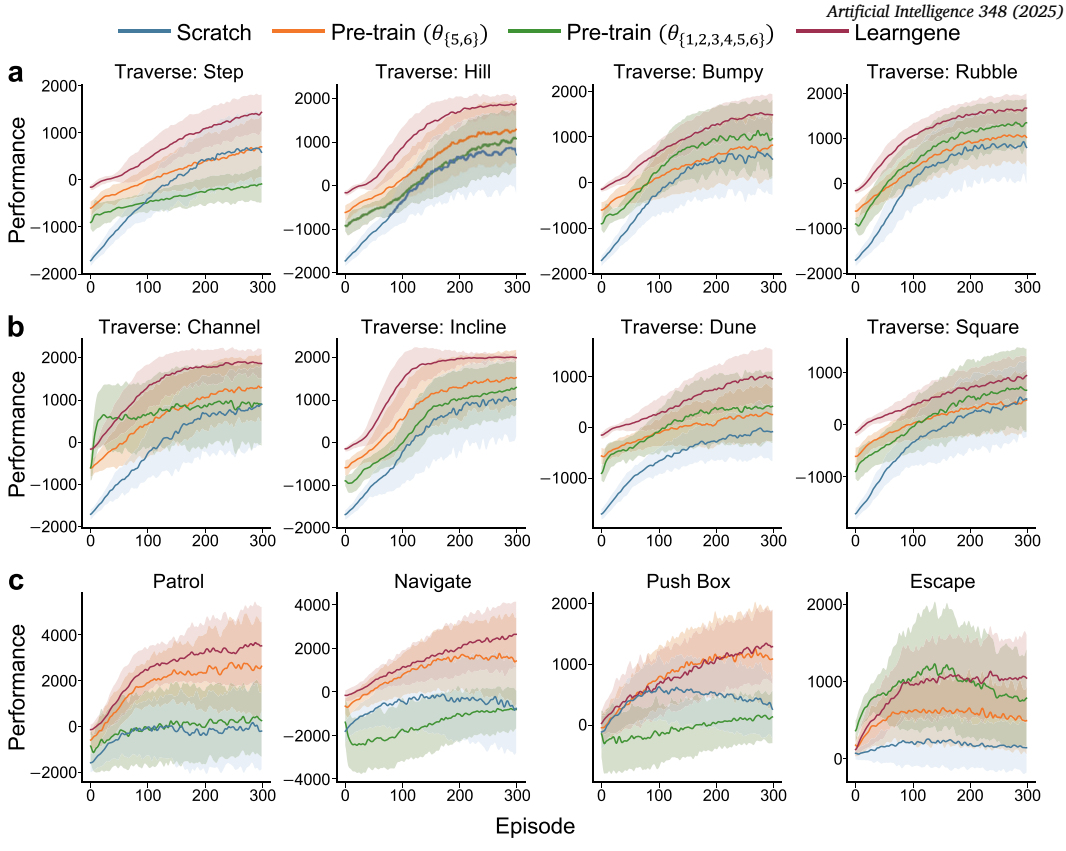


Fig. 9. Comparison with Pre-trained Agents. Mean and standard deviation of rewards for agents trained on the TRAVERSE with (**a**) training obstacles, (**b**) novel obstacles, and (**c**) other novel tasks, including PATROL, NAVIGATE, PUSH BOX and ESCAPE. To enhance clarity, we apply a Gaussian filter to smooth these curves, emphasizing their relative relationships. For pre-trained agents, knowledge transfer is conducted by transferring either the entire policy network (i.e., $\theta_{\{1,2,3,4,5,6\}}$) or only the parameters corresponding to the optimal structural form of learngene (i.e., $\theta_{\{5,6\}}$).

5.7. Evolution of learngenes through knowledge accumulation

Lamarckism suggests that acquired traits can be inherited, enabling instinctive behaviors shortly after birth [128,129]. Agent evolution aligns with this principle, with learngenes continuously refining and accumulating task-agnostic knowledge through inheritance. Specifically, agents were born with learngenes and refine task-agnostic knowledge by interacting with their environment and updating learngene parameters. These refined learngenes, carrying accumulated knowledge, are then inherited by descendants, sustaining the cycle of knowledge accumulation across generations.

Fig. 11a highlights the evolutionary path of learngenes by visualizing several main Gene Trees, with the backbone of the largest tree representing the evolution of the optimal learngene identified in Section 5.1. Along this path (marked by ⟶), agents inheriting later-generation learngenes exhibit higher performance, benefiting from accumulated task-agnostic knowledge (Fig. 11b). A more detailed view of the training process (Fig. 11c) shows that learngenes evolved over more generations provide a higher starting point and enable faster knowledge acquisition.

5.8. Diversity and stability of learngenes during evolution

GRL preserves learngene diversity using the Gene Pool to store superior learngenes (Section 3.3.4) and probability-based sampling for learngene extraction (Section 3.3.5). In Fig. 11a, ancestral learngenes with lighter colors are often surrounded by darker nodes, indicating that lower-fitness learngenes can evolve into superior ones (with the path marked by ⇢) by accumulating task-agnostic knowledge over generations. This highlights the necessity of preserving suboptimal learngenes to promote diversity and long-term improvement.

Furthermore, learngenes with the optimal structural form $\theta_{\{5,6\}}$ exhibit remarkable stability throughout evolution. Fig. 11d visualizes the average parameter changes for each structural form across evolution. The optimal form $\theta_{\{5,6\}}$ remains persistent throughout evolution with minimal parameter changes, and these changes progressively decrease over time, highlighting its stability. This demonstrates that the knowledge encapsulated in $\theta_{\{5,6\}}$ is highly inheritable, allowing agents to adapt to new tasks with minimal adjustments.

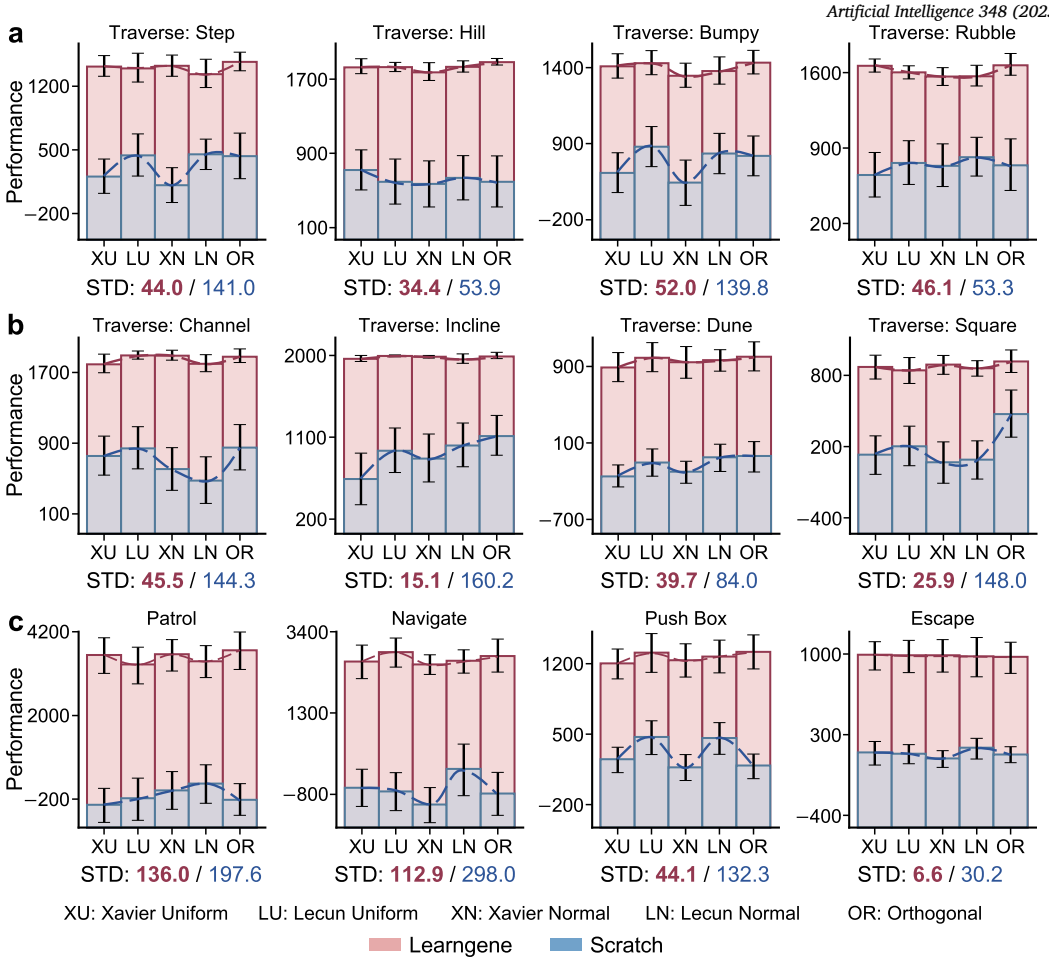


Fig. 10. Training with Different Initialization Methods. We apply five common initialization methods to non-learngene parts of agents inheriting learngenes and all parameters of agents learning from scratch. Bars represent the average rewards at the end of training (i.e., 300th episode) on TRAVERSE with (**a**) training obstacles, (**b**) novel obstacles, and (**c**) other novel tasks, including PATROL, NAVIGATE, PUSH BOX and ESCAPE, with the error bars denoting the 95% bootstrapped confidence intervals. We additionally report the standard deviations of performance across different initialization methods to quantify the impact of initialization on an agent's adaptability. Agents inheriting learngenes exhibit stable learning performance regardless of the initialization of non-learngene components, whereas agents trained from scratch remain highly sensitive to initialization methods.

6. Conclusion

In this paper, we introduce learngenes, neural network fragments that serve as carriers of task-agnostic knowledge across agents. To condense task-agnostic knowledge into learngenes, we propose Genetic Reinforcement Learning (GRL), which establishes a genetic mechanism for agent training and evolution. Following Lamarckian inheritance, learngenes continuously refine and accumulate task-agnostic knowledge across agents and tasks over generations. Experiments demonstrate that agents inheriting learngenes develop innate abilities to acquire rewards, leveraging ancestral knowledge for efficient learning while maintaining exploration flexibility. Additionally, we highlight the advantages of learngene-based knowledge transfer over evolution-based search and traditional pre-trained models, underscoring the necessity of condensing task-agnostic knowledge.

In conclusion, inspired by genetic inheritance in nature, our work introduces a novel paradigm for knowledge transfer and model initialization in AI, offering new possibilities for more adaptive, efficient, and scalable learning systems.

7. Discussion and future works

In this work, we extract learngenes from the ant agent using Genetic Reinforcement Learning (GRL), which condenses task-agnostic knowledge into learngenes through the TRAVERSE task with four types of training obstacles. The task-agnostic knowledge encapsulated in learngenes encompasses fundamental principles of agent control (e.g., movement dynamics, reward optimization, harm avoidance), as well as generalized patterns of coordination (e.g., torso-limb interaction) and core locomotion strategies. This enables efficient policy learning across a broad spectrum of tasks, including those involving **complex locomotion**, **dynamic navigation**, **balance control**, **object manipulation**, and so on. To verify this generalizability, we assess learngenes across TRAVERSE with novel obstacles,

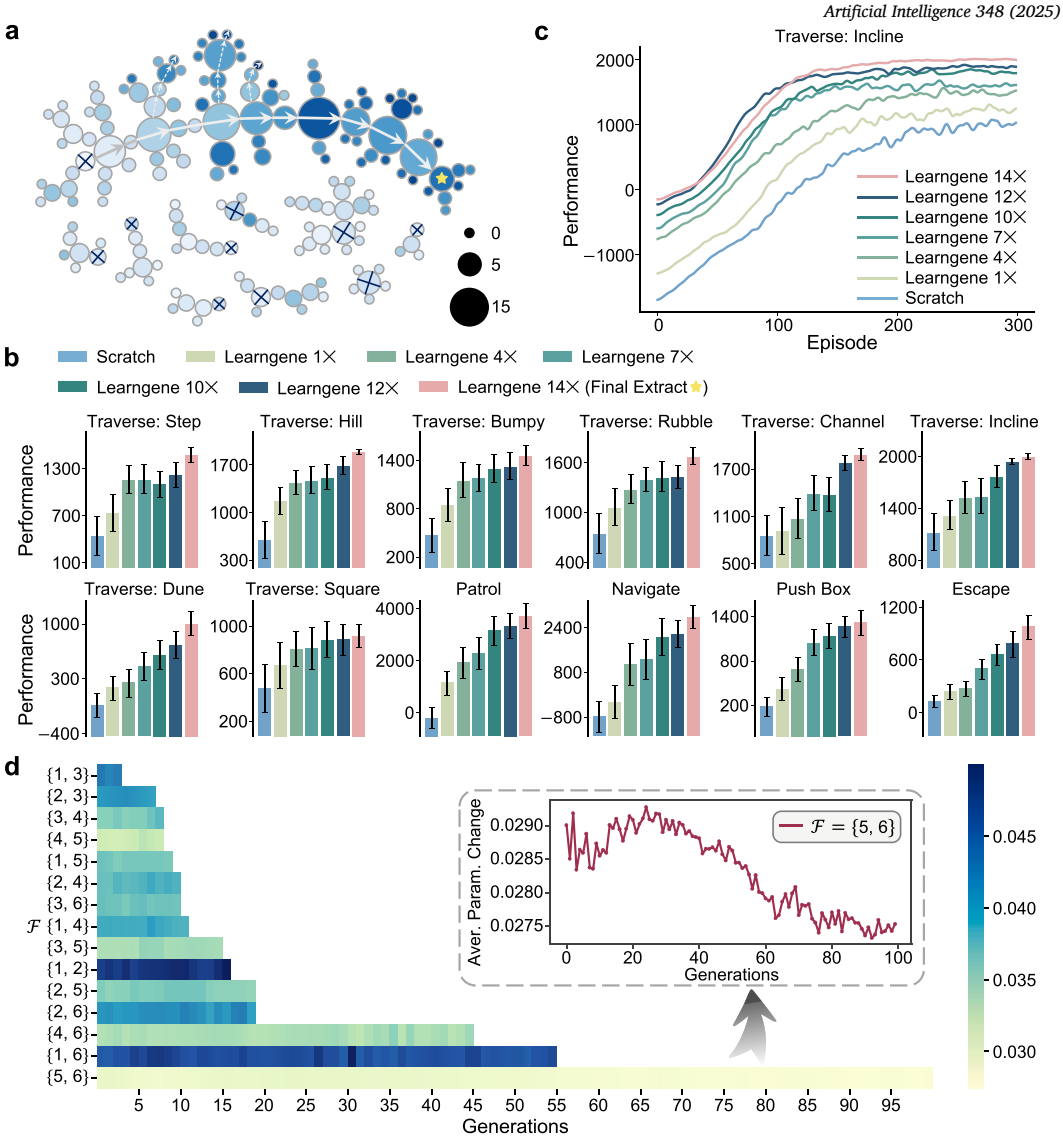


Fig. 11. Evolution of Learngenes. a Visualization of the Gene Tree ($\theta_{|\mathcal{F}|=2}$). Each dot represents a learngene, with its size indicating the number of descendants and opacity reflecting the corresponding score (darker indicates higher scores). Leaf nodes (i.e., learngenes with zero descendants) are partially omitted for clarity. The largest Gene Tree captures the evolution of learngenes with the form $\theta_{\mathcal{F}=\{5,6\}}$, with ⟶ marking its backbone. The optimal learngene $g^\star$, extracted in Section 5.1, is marked with ★. Root learngenes are marked with ✕, and ⇢ highlights representative branches illustrating learngene diversity. **b** Each bar represents the average rewards r_{ep} achieved at the end of training, with 95% bootstrapped confidence intervals, for agents inheriting learngenes from different generations along the largest Gene Tree backbone (marked by ⟶). The notation $i\times$ denotes the path length from the root node, indicating the number of evolutionary generations undergone by the initial learngene (i.e., root node). **c** Detailed training curves of agents inheriting learngenes from different generations on the TRAVERSE INCLINE. **d** Heat map of average parameter changes in learngenes of each structural form ($\theta_{|\mathcal{F}|=2}$) throughout evolution, measured by Manhattan distance. The length of each heat map represents the survival duration of the corresponding form. A convergence curve for $\theta_{\mathcal{F}=\{5,6\}}$ highlights its stabilization trend over generations.

as well as distinct tasks such as PATROL, NAVIGATE, PUSH BOX, and ESCAPE, and further demonstrate strong robustness of learngenes to variations in perception input, task objectives, environmental conditions, and terrain features.

The effectiveness of the task-agnostic knowledge encapsulated in learngenes may decrease when faced with tasks requiring highly specialized strategies (e.g., complex decision-making in unpredictable environments) or tasks with extreme physical constraints (e.g., zero-gravity scenarios). In such cases, relearning or extracting new learngenes may be necessary. However, even under these conditions, previous learngenes **do not lead to negative effects**; instead, they provide a valuable foundation for further adaptation to the specific task. A key limitation of the current approach is that learngenes are architecture-specific. As a result, they cannot be directly transferred between agents with different morphologies, since the interpretation of control signals (e.g., hinge movements) in the output space varies across agent types, such as between ant and humanoid agents. This highlights an important direction for future research: exploring task-agnostic knowledge shared across diverse architectures. Specifically, replacing the direct use of neural

network parameters with indirect encoding methods (e.g., hypernetworks [130–132]) could enhance the transferability of learngenes across diverse agent morphologies.

Learngenes can also extend beyond fully connected networks and RL to mainstream models like CNNs and Transformers, where supervised learning could be utilized to condense task-agnostic knowledge into specific convolutional kernels or transformer blocks as learngenes, enabling efficient knowledge transfer and flexible model initialization. Additionally, learngenes can enhance real-world robotics by reducing potential damage from unsafe behaviors and improving training efficiency in simulators to optimize strategy development.

CRediT authorship contribution statement

Fu Feng: Writing – review & editing, Writing – original draft, Methodology, Formal analysis. **Jing Wang:** Writing – original draft, Methodology, Formal analysis. **Xu Yang:** Supervision. **Xin Geng:** Writing – review & editing, Supervision, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Xin Geng reports financial support was provided by Southeast University. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

We sincerely thank Ruixiao Shi, Wenqian Li, Congzhi Zhang, Yucheng Xie and Jianlu Shen for their insightful discussions on this work. We also appreciate Freepik for contributing to the figure design. This research was supported by the Jiangsu Science Foundation (BG2024036, BK20243012), the National Natural Science Foundation of China (62125602, U24A20324, 92464301, 62306073), China Postdoctoral Science Foundation (2022M720028, 2025T180432), the Xplorer Prize, and the Fundamental Research Funds for the Central Universities (2242025K30024).

Appendix A. Detailed task information

The task and obstacle designs in this paper are inspired by [40]. Detailed descriptions and parameters for each task are provided below:

Traverse: This task takes place in a rectangular environment of 120×40 square meters (m^2), where more than half of the area is occupied by obstacles, with the remaining portion as flat terrain. Agents begin at the starting line and complete the task by traversing obstacles to reach the finishing line. The reward r_t at each time step follows Eq. (8), with $\mu = 1$ and $\nu = 0.2$.

The parameters for each obstacle are as follows:

- **Step** is a cuboid obstacle with a height ranging from $[0.15, 0.20]$ m and a length ranging from $[2.00, 3.00]$ m. Steps are arranged in a sequence of $\tau-$steps up followed by $\tau-$steps down, where $\tau \in [3, 6]$.
- **Bumpy** is a trapezoidal obstacle with a height ranging from $[-0.30, 0.36]$ m and a width of 2.00 m.
- **Hill** is a semi-cylindrical obstacle shaped by a sine wave, with a height ranging from $[0.23, 0.90]$ m and a width ranging from $[4.00, 10.00]$ m.
- **Rubble** is composed of right square pyramids, with a height ranging from $[0.38, 1.50]$ m and a width ranging from $[2.00, 4.00]$ m.
- **Channel** is a semi-cylindrical depression formed by a sine wave, with a depth of -0.30 m and a width of 2.00 m.
- **Incline** is a sequence of planes inclined upwards and downwards, with a height of 1.5 m. The width of the downward plane is 10.00 m, while the upward plane measures 20.00 m.
- **Dune** is an obstacle created using a 2D Gaussian function within a 4×4 m^2 square area, with a height of approximately 0.84 m.
- **Square** is a flat square with dimensions of 4×4 m^2, and a height ranging from $[-0.15, 0.15]$ m.

Patrol: The agent operates in a 15×15 m^2 environment containing two target points. PATROL requires the agent to travel from one target to the other, and immediately reverse direction upon arrival, which challenges its agility and directional control. The agent perceives the target's position relative to its reference frame, and the reward at each time step is given by:

$$r_t = \mu(d_t^{\text{tgt}} - d_{t-1}^{\text{tgt}}) - \nu||a_t||^2 \tag{A.1}$$

where d_t^{tgt} represents the distance to the target at time step t. For PATROL, we set $\mu = 100$ and $\nu = 0.2$ to encourage efficient and controlled movements.

Navigate: The agent navigates in a 15×15 m^2 area with multiple target points. Upon reaching a target, a new one will be randomly generated, requiring the agent to navigate efficiently in various directions. The agent's observations and reward function follow the same formulation as in PATROL.

Push Box: This task introduces a box placed at a random location within a 15×15 m^2 environment. The agent must locate the box, push it as far as possible, and then approach it again to repeat this process. The agent perceives the box's position relative to its own, and the reward at each time step is defined as:

$$r_t = \mu_1(d_t^{\text{agt}} - d_{t-1}^{\text{agt}}) + \mu_2(d_t^{\text{box}} - d_{t-1}^{\text{box}}) - \nu||a_t||^2 \tag{A.2}$$

where d_t^{agt} is the distance between the agent and the box, while d_t^{box} is the distance between the box and the environment center at time step t. For PUSH BOX, we set $\mu_1 = 100$, $\mu_2 = 200$ and $\nu = 0.01$ to emphasize the box-pushing objective.

Escape: The agent begins at the center of a bowl-shaped terrain (15×15 m^2) surrounded by small hills, aiming to maximize its geodesic distance from the starting point and escape the hilly region. This task evaluates the agent's ability to maintain balance while navigating uneven terrain. The reward at each time step is computed as:

$$r_t = \mu(d_t^{\text{agt}} - d_{t-1}^{\text{agt}}) - \nu||a_t||^2 \tag{A.3}$$

where d_t^{agt} is the distance between the agent and the environment center at time step t. For ESCAPE, we set $\mu = 100$ and $\nu = 0.01$ to encourage dynamic movement while allowing flexibility for larger actions to traverse challenging terrain effectively.

Appendix B. Calculation of knowledge transfer rate

The knowledge transfer rate $\Omega_{j\to i}$ quantifies the transferability of knowledge from obstacle o_j to obstacle o_i, which is defined as:

$$\Omega_{j\to i} = \frac{\mathcal{R}_{j\to i} - r_{\text{ep}}^{i\text{-base}}}{\mathcal{R}_{i\to i} - r_{\text{ep}}^{i\text{-base}}} \tag{B.1}$$

where $\mathcal{R}_{j\to i}$ represents the reward achieved on TRAVERSE with obstacle o_i by agents pre-trained on o_j (direct transfer without fine-tuning). $r_{\text{ep}}^{i\text{-base}}$ is the baseline reward for obstacle o_i, marking the stage where agents successfully stand and begin acquiring task-specific knowledge. In Fig. 4, agents are pre-trained from scratch on each task for 300 episodes, with $r_{\text{ep}}^{i\text{-base}}$ computed as the reward at the 150th episode, denoted as $r_{\text{ep-150th}}^{i\text{-base}}$.

Appendix C. Hyperparameters in training and evolution

Table C.1 and Table C.2 list the hyperparameters used for agent training with the PPO algorithm and learngene evolution within the GRL framework, respectively.

Table C.1
Hyperparameters in PPO.

Hyperparameter	Value
Discount	0.99
GAE parameter	0.95
PPO clipping parameter	0.2
Policy epochs	10
Batch size	64
Layer number	6
Hidden dimensions	48
Entropy coefficient	0.01
Reward scaling	0.1
Observation normalization	Yes
Observation clipping	[−5, 5]
Timesteps per rollout	4096
Timesteps per episode	3000
Training iterations	50
Optimizer	Adam
Initial learning rate	0.0003
Gradient clipping	0.5
Clipped value function	Yes
Value loss coefficient	0.5

Table C.2
Hyperparameters in GRL.

Hyperparameter	Value
Training task number	4
Novel task number	4
Population size	50
Gene Pool size (each form)	7
Obsolete number	2
Generational decay coefficient	0.6
Parental decay coefficient	0.46
Agents number in competition	3
Generation number	100
Learngene layer number	2

Appendix D. Details of the action space of agents

The action space of the ant agent comprises eight continuous control variables, each corresponding to the torque applied to one of its eight hinges connecting the ant's body parts. The agent's structural design is illustrated in Fig. D.12.

Each torque value range from −1 to 1, enabling precise joint control and coordinated limb movement for locomotion and environmental interaction. The torque values are represented as 32-bit floating-point numbers (float32). Additional specifications are provided in Table D.1.

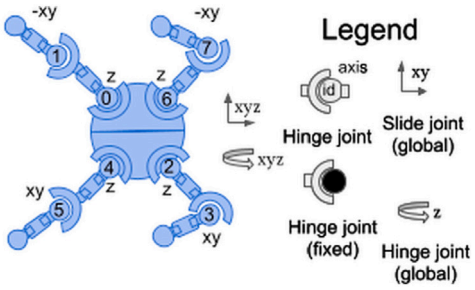


Fig. D.12. The architecture of the ant agent, with this image sourced from the Gymnasium documentation.

Table D.1
The detailed action space of ant agents. Each action corresponds to the torque applied to a hinge between two connected parts, denoted as (part1, part2). The "Num" column corresponds to the number in Fig. D.12.

Num	Action	Control	Joint	Type
0	(torso, back right hip)	[−1, 1]	hinge	torque
1	(back right link1, back right link2)	[−1, 1]	hinge	torque
2	(torso, front left hip)	[−1, 1]	hinge	torque
3	(front left link1, front left link2)	[−1, 1]	hinge	torque
4	(torso, front right hip)	[−1, 1]	hinge	torque
5	(front right link1, front right link2)	[−1, 1]	hinge	torque
6	(torso, back left hip)	[−1, 1]	hinge	torque
7	(back left link1, back left link2)	[−1, 1]	hinge	torque

Appendix E. Video demonstration of learngenes

To visually illustrate the evolution of learngenes and their impact on intelligent agents, we provide a video comprising four sections.

The first section presents the evolution of learngenes, demonstrating that agents inheriting learngenes from later generations exhibit improved adaptability to their environments. The subsequent sections highlight the advantages provided by learngenes, including (1) innate abilities, where agents with learngenes perform controlled actions and move progressively toward the finishing line, and (2,3) enhanced learning efficiency, where agents inheriting learngenes acquire superior strategies more rapidly in TRAVERSE with both training and novel obstacles.

The video is available at Google Drive (https://drive.google.com/file/d/1AMoiHlTkPGqDwLWYZHKW6wivBunc0f3W/view?usp=sharing).

Appendix F. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.artint.2025.104421.

Data availability

Data will be made available on request.

References

[1] E. Danchin, S. Nöbel, A. Pocheville, A.-C. Dagaeff, L. Demay, M. Alphand, S. Ranty-Roby, L. Van Renssen, M. Monier, E. Gazagne, et al., Cultural flies: conformist social learning in fruitflies predicts long-lasting mate-choice traditions, Science 362 (6418) (2018) 1025–1030.
[2] S.R. Howard, A. Avarguès-Weber, J.E. Garcia, A.D. Greentree, A.G. Dyer, Numerical cognition in honeybees enables addition and subtraction, Sci. Adv. 5 (2) (2019) eaav0961.
[3] D. Hassabis, D. Kumaran, C. Summerfield, M. Botvinick, Neuroscience-inspired artificial intelligence, Neuron 95 (2) (2017) 245–258.
[4] S. Gronauer, K. Diepold, Multi-agent deep reinforcement learning: a survey, Artif. Intell. Rev. 55 (2) (2022) 895–943.
[5] T.T. Nguyen, N.D. Nguyen, S. Nahavandi, Deep reinforcement learning for multiagent systems: a review of challenges, solutions, and applications, IEEE Trans. Cybern. 50 (9) (2020) 3826–3839.
[6] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489.
[7] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, Nature 550 (7676) (2017) 354–359.
[8] Y. Matsuo, Y. LeCun, M. Sahani, D. Precup, D. Silver, M. Sugiyama, E. Uchibe, J. Morimoto, Deep learning, reinforcement learning, and world models, Neural Netw. 152 (2022) 267–275.
[9] N. Kriegeskorte, Deep neural networks: a new framework for modeling biological vision and brain information processing, Annu. Rev. Vis. Sci. 1 (1) (2015) 417–446.
[10] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F.L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al., Gpt-4 technical report, arXiv preprint, arXiv:2303.08774, 2023.
[11] E.A. Van Dis, J. Bollen, W. Zuidema, R. van Rooij, C.L. Bockting, ChatGPT: five priorities for research, Nature 614 (7947) (2023) 224–226.
[12] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, et al., Scaling rectified flow transformers for high-resolution image synthesis, in: Proceedings of International Conference on Machine Learning (ICML'24), 2024, pp. 1–13.
[13] E. Richardson, K. Goldberg, Y. Alaluf, D. Cohen-Or, Conceptlab: creative concept generation using vlm-guided diffusion prior constraints, ACM Trans. Graph. 43 (3) (2024) 1–14.
[14] F. Feng, Y. Xie, J. Wang, X. Geng, Redefining <creative> in dictionary: towards an enhanced semantic understanding of creative generation, arXiv preprint, arXiv:2410.24160, 2024.
[15] T. Krink, F. Vollrath, Analysing spider web-building behaviour with rule-based simulations and genetic algorithms, J. Theor. Biol. 185 (3) (1997) 321–331.
[16] B. Gorissen, C. Wolschrijn, F. Serra Bragança, A. Geerts, W. Leenders, W. Back, P. Van Weeren, The development of locomotor kinetics in the foal and the effect of osteochondrosis, Equine Vet. J. 49 (4) (2017) 467–474.
[17] B.B. Wong, U. Candolin, Behavioral responses to changing environments, Behav. Ecol. 26 (3) (2015) 665–673.
[18] A. Sih, M.C. Ferrari, D.J. Harris, Evolution and behavioural responses to human-induced rapid environmental change, Evol. Appl. 4 (2) (2011) 367–387.
[19] A. Braga, R.K. Logan, The emperor of strong AI has no clothes: limits to artificial intelligence, Information 8 (4) (2017) 156.
[20] J.J. Oró, Evolution of the brain: from behavior to consciousness in 3.4 billion years, Neurosurgery 54 (6) (2004) 1287–1297.
[21] Y. Tan, P. Hu, L. Pan, J. Huang, L. Huang, Rlx2: training a sparse deep reinforcement learning model from scratch, in: Proceedings of the International Conference on Learning Representations (ICLR'22), 2022, pp. 1–13.
[22] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, J.T. Springenberg, Learning by playing solving sparse reward tasks from scratch, in: Proceedings of International Conference on Machine Learning (ICML'18), 2018, pp. 4344–4353.
[23] A. Bakhtin, D. Wu, A. Lerer, N. Brown, No-press diplomacy from scratch, in: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'21), 2021, pp. 18063–18074.
[24] R. Agarwal, M. Schwarzer, P.S. Castro, A.C. Courville, M. Bellemare, Reincarnating reinforcement learning: reusing prior computation to accelerate progress, in: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'22), 2022, pp. 28955–28971.
[25] D. Li, H. Wu, J. Zhang, K. Huang, A2-rl: aesthetics aware reinforcement learning for image cropping, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'18), 2018, pp. 8193–8201.
[26] Z. Wang, T. Hong, Reinforcement learning for building controls: the opportunities and challenges, Appl. Energy 269 (2020) 115036.
[27] J. Bohacek, I.M. Mansuy, Molecular insights into transgenerational non-genetic inheritance of acquired behaviours, Nat. Rev., Genet. 16 (11) (2015) 641–652.
[28] C.H. Waddington, Canalization of development and the inheritance of acquired characters, Nature 150 (3811) (1942) 563–565.
[29] T. Bäck, H.-P. Schwefel, An overview of evolutionary algorithms for parameter optimization, Evol. Comput. 1 (1) (1993) 1–23.
[30] D. Whitley, S. Rana, J. Dzubera, K.E. Mathias, Evaluating evolutionary algorithms, Artif. Intell. 85 (1–2) (1996) 245–276.
[31] L. Spector, Evolution of artificial intelligence, Artif. Intell. 170 (18) (2006) 1251–1253.
[32] A.M. Zador, A critique of pure learning and what artificial neural networks can learn from animal brains, Nat. Commun. 10 (2019) 3770.
[33] D.L. Barabási, T. Beynon, Á. Katona, N. Perez-Nieves, Complex computation from developmental priors, Nat. Commun. 14 (1) (2023) 2226.
[34] F. Feng, J. Wang, X. Geng, Transferring core knowledge via learngenes, arXiv preprint, arXiv:2401.08139, 2024.
[35] H. Ren, J. Materzynska, R. Gandikota, D. Bau, A. Torralba, Art-free generative models: art creation without graphic art knowledge, arXiv preprint, arXiv: 2412.00176, 2024.
[36] Z. Wang, Z. Dai, B. Póczos, J. Carbonell, Characterizing and avoiding negative transfer, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19), 2019, pp. 11293–11302.
[37] M.T. Rosenstein, Z. Marx, L.P. Kaelbling, T.G. Dietterich, To transfer or not to transfer, in: Proceedings of NIPS 2005 Workshop on Transfer Learning, 2005, pp. 1–4.
[38] M.A. Jamal, G.-J. Qi, Task agnostic meta-learning for few-shot learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19), 2019, pp. 11719–11727.
[39] J. Rajasegaran, S. Khan, M. Hayat, F.S. Khan, M. Shah, Itaml: an incremental task-agnostic meta-learning approach, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20), 2020, pp. 13588–13597.
[40] A. Gupta, S. Savarese, S. Ganguli, L. Fei-Fei, Embodied intelligence via learning and evolution, Nat. Commun. 12 (1) (2021) 5721.

[41] A.A. Hoffmann, C.M. Sgrò, Climate change and evolutionary adaptation, Nature 470 (7335) (2011) 479–485.
[42] H. Moravec, Mind Children: The Future of Robot and Human Intelligence, Harvard University Press, 1988.
[43] Z. Zhu, K. Lin, A.K. Jain, J. Zhou, Transfer learning in deep reinforcement learning: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 45 (11) (2023) 13344–13362.
[44] K.O. Stanley, J. Clune, J. Lehman, R. Miikkulainen, Designing neural networks through neuroevolution, Nat. Mach. Intell. 1 (2019) 24–35.
[45] P. Li, J. Hao, H. Tang, X. Fu, Y. Zhen, K. Tang, Bridging evolutionary algorithms and reinforcement learning: a comprehensive survey on hybrid algorithms, IEEE Trans. Evol. Comput. (2024).
[46] O. Sigaud, Combining evolution and deep reinforcement learning for policy search: a survey, ACM Trans. Evolut. Learn. 3 (3) (2023) 1–20.
[47] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'09), 2009, pp. 248–255.
[48] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, Q. He, A comprehensive survey on transfer learning, Proc. IEEE 109 (1) (2020) 43–76.
[49] C. Devin, A. Gupta, T. Darrell, P. Abbeel, S. Levine, Learning modular neural network policies for multi-task and multi-robot transfer, in: Proceedings of IEEE International Conference on Robotics and Automation, 2017, pp. 2169–2176.
[50] F. Fernández, M. Veloso, Probabilistic policy reuse in a reinforcement learning agent, in: Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems, 2006, pp. 720–727.
[51] A. Barreto, W. Dabney, R. Munos, J.J. Hunt, T. Schaul, H.P. van Hasselt, D. Silver, Successor features for transfer in reinforcement learning, in: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'17), 2017, pp. 1–11.
[52] A.A. Rusu, S.G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, R. Hadsell, Policy distillation, arXiv preprint, arXiv:1511.06295, 2015.
[53] W.M. Czarnecki, R. Pascanu, S. Osindero, S. Jayakumar, G. Swirszcz, M. Jaderberg, Distilling policy distillation, in: Proceedings of International Conference on Artificial Intelligence and Statistics, 2019, pp. 1331–1340.
[54] E. Parisotto, J.L. Ba, R. Salakhutdinov, Actor-mimic: deep multitask and transfer reinforcement learning, arXiv preprint, arXiv:1511.06342, 2015.
[55] A. Braylan, M. Hollenbeck, E. Meyerson, R. Miikkulainen, Reuse of neural modules for general video game playing, in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'16), 2016, pp. 353–359.
[56] P. Dayan, Improving generalization for temporal difference learning: the successor representation, Neural Comput. 5 (4) (1993) 613–624.
[57] T.D. Kulkarni, A. Saeedi, S. Gautam, S.J. Gershman, Deep successor reinforcement learning, arXiv preprint, arXiv:1606.02396, 2016.
[58] A. Telikani, A. Tahmassebi, W. Banzhaf, A.H. Gandomi, Evolutionary machine learning: a survey, ACM Comput. Surv. 54 (8) (2021) 1–35.
[59] P.A. Vikhar, Evolutionary algorithms: a critical review and its future prospects, in: Proceedings of the International Conference on Global Trends in Signal Processing, Information Computing and Communication, 2016, pp. 261–265.
[60] Z.-H. Zhou, Y. Yu, C. Qian, Evolutionary Learning: Advances in Theories and Algorithms, Springer, 2019.
[61] K.O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, Evol. Comput. 10 (2) (2002) 99–127.
[62] K.O. Stanley, D.B. D'Ambrosio, J. Gauci, A hypercube-based encoding for evolving large-scale neural networks, Artif. Life 15 (2) (2009) 185–212.
[63] S. Mirjalili, S. Mirjalili, Genetic algorithm, in: Evolutionary Algorithms and Neural Networks: Theory and Applications, 2019, pp. 43–55.
[64] S. Sivanandam, S. Deepa, S. Sivanandam, S. Deepa, Genetic Algorithms, Springer, 2008.
[65] S. Mirjalili, J. Song Dong, A.S. Sadiq, H. Faris, Genetic algorithm: theory, literature review, and application in image reconstruction, in: Nature-Inspired Optimizers: Theories, Literature Reviews and Applications, 2020, pp. 69–85.
[66] N. Hansen, The cma evolution strategy: a comparing review, in: Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms, 2006, pp. 75–102.
[67] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2010) 4–31.
[68] S. Katoch, S.S. Chauhan, V. Kumar, A review on genetic algorithm: past, present, and future, Multimed. Tools Appl. 80 (2021) 8091–8126.
[69] H.T. Ünal, F. Başçiftçi, Evolutionary design of neural network architectures: a review of three decades of research, Artif. Intell. Rev. 55 (3) (2022) 1723–1802.
[70] V. Mishra, L. Kane, A survey of designing convolutional neural network using evolutionary algorithms, Artif. Intell. Rev. 56 (6) (2023) 5095–5132.
[71] A. Darwish, A.E. Hassanien, S. Das, A survey of swarm and evolutionary computing approaches for deep learning, Artif. Intell. Rev. 53 (3) (2020) 1767–1812.
[72] X. Zhou, A.K. Qin, M. Gong, K.C. Tan, A survey on evolutionary construction of deep neural networks, IEEE Trans. Evol. Comput. 25 (5) (2021) 894–912.
[73] K.O. Stanley, Compositional pattern producing networks: a novel abstraction of development, Genet. Program. Evol. Mach. 8 (2007) 131–162.
[74] C. Fernando, D. Banarse, M. Reynolds, F. Besse, D. Pfau, M. Jaderberg, M. Lanctot, D. Wierstra, Convolution by evolution: differentiable pattern producing networks, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2016, pp. 109–116.
[75] A. Soltoggio, K.O. Stanley, S. Risi, Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks, Neural Netw. 108 (2018) 48–67.
[76] S. Swarup, S.R. Ray, Cross-domain knowledge transfer using structured representations, in: Proceedings of National Conference on Artificial Intelligence, 2006, pp. 506–511.
[77] S. Nagae, S. Kawai, H. Nobuhara, Transfer learning layer selection using genetic algorithm, in: Proceedings of IEEE Congress on Evolutionary Computation, 2020, pp. 1–6.
[78] R. de Lima Mendes, A.H. da Silva Alves, M. de Souza Gomes, P.L.L. Bertarini, L.R. do Amaral, Many layer transfer learning genetic algorithm (mltlga): a new evolutionary transfer learning approach applied to pneumonia classification, in: Proceedings of IEEE Congress on Evolutionary Computation, 2021, pp. 2476–2482.
[79] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint, arXiv:1609.04747, 2016.
[80] P. Ladosz, L. Weng, M. Kim, H. Oh, Exploration in deep reinforcement learning: a survey, Inf. Fusion 85 (2022) 1–22.
[81] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. Xi Chen, Y. Duan, J. Schulman, F. DeTurck, P. Abbeel, # exploration: a study of count-based exploration for deep reinforcement learning, in: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'17), vol. 30, 2017.
[82] A.W. Beggs, On the convergence of reinforcement learning, J. Econ. Theory 122 (1) (2005) 1–36.
[83] A. Gosavi, A reinforcement learning algorithm based on policy iteration for average reward: empirical results with yield management and convergence analysis, Mach. Learn. 55 (2004) 5–29.
[84] T. Eimer, M. Lindauer, R. Raileanu, Hyperparameters in reinforcement learning and how to tune them, in: Proceedings of International Conference on Machine Learning (ICML'23), 2023, pp. 9104–9149.
[85] B. Zhang, R. Rajan, L. Pineda, N. Lambert, A. Biedenkapp, K. Chua, F. Hutter, R. Calandra, On the importance of hyperparameter optimization for model-based reinforcement learning, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2021, pp. 4015–4023.
[86] A. Leite, M. Candadai, E.J. Izquierdo, Reinforcement learning beyond the Bellman equation: exploring critic objectives using evolution, in: Artificial Life Conference Proceedings, 2020, pp. 441–449.
[87] E. Marchesini, D. Corsi, A. Farinelli, Genetic soft updates for policy evolution in deep reinforcement learning, in: Proceedings of the International Conference on Learning Representations (ICLR'21), 2021, pp. 1–12.
[88] E. Marchesini, C. Amato, Improving deep policy gradients with value function search, in: Proceedings of the International Conference on Learning Representations (ICLR'23), 2023, pp. 1–12.
[89] Y. Ma, T. Liu, B. Wei, Y. Liu, K. Xu, W. Li, Evolutionary action selection for gradient-based policy learning, in: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'22), 2022, pp. 579–590.
[90] Z. Shi, S.P. Singh, Soft actor-critic with cross-entropy policy optimization, arXiv preprint, arXiv:2112.11115, 2021.

[91] L. Shao, Y. You, M. Yan, S. Yuan, Q. Sun, J. Bohg, Grac: self-guided and self-regularized actor-critic, in: Proceedings of Conference on Robot Learning, 2022, pp. 267–276.
[92] L. Pan, L. Huang, T. Ma, H. Xu, Plan better amid conservatism: offline multi-agent reinforcement learning with actor rectification, in: Proceedings of International Conference on Machine Learning (ICML'22), 2022, pp. 17221–17237.
[93] S. Elfwing, E. Uchibe, K. Doya, Online meta-learning by parallel algorithm competition, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2018, pp. 426–433.
[94] J.K. Franke, G. Koehler, A. Biedenkapp, F. Hutter, Sample-efficient automated deep reinforcement learning, in: Proceedings of the International Conference on Learning Representations (ICLR'21), 2021, pp. 1–12.
[95] Y. Tang, K. Choromanski, Online hyper-parameter tuning in off-policy learning via evolutionary strategies, arXiv preprint, arXiv:2006.07554, 2020.
[96] S. Khadka, K. Tumer, Evolution-guided policy gradient in reinforcement learning, in: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'18), 2018, pp. 1–13.
[97] S. Khadka, S. Majumdar, T. Nassar, Z. Dwiel, E. Tumer, S. Miret, Y. Liu, K. Tumer, Collaborative evolutionary reinforcement learning, in: Proceedings of International Conference on Machine Learning (ICML'19), 2019, pp. 3341–3350.
[98] C. Bodnar, B. Day, P. Lió, Proximal distilled evolutionary reinforcement learning, in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'20), 2020, pp. 3283–3290.
[99] H. Zheng, P. Wei, J. Jiang, G. Long, Q. Lu, C. Zhang, Cooperative heterogeneous deep reinforcement learning, in: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'20), vol. 33, 2020, pp. 17455–17465.
[100] Y. Wang, T. Zhang, Y. Chang, X. Wang, B. Liang, B. Yuan, A surrogate-assisted controller for expensive evolutionary reinforcement learning, Inf. Sci. 616 (2022) 539–557.
[101] B. Zheng, R. Cheng, Rethinking population-assisted off-policy reinforcement learning, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2023, pp. 624–632.
[102] J.R. Koza, Genetic Programming II: Automatic Disovery of Reusable Programs, MIT Press, 1994.
[103] J.R. Koza, F.H. Bennett, D. Andre, M.A. Keane, F. Dunlap, Automated synthesis of analog electrical circuits by means of genetic programming, IEEE Trans. Evol. Comput. 1 (2) (2002) 109–128.
[104] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: a survey, J. Artif. Intell. Res. 4 (1996) 237–285.
[105] S. Arora, P. Doshi, A survey of inverse reinforcement learning: challenges, methods and progress, Artif. Intell. 297 (2021) 103500.
[106] P. Tadepalli, D. Ok, Model-based average reward reinforcement learning, Artif. Intell. 100 (1–2) (1998) 177–224.
[107] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, John Wiley & Sons, 2014.
[108] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint, arXiv:1707.06347, 2017.
[109] V. Konda, J. Tsitsiklis, Actor-critic algorithms, in: Proceedings of the Annual Conference on Neural Information Processing Systems, 1999, pp. 1008–1014.
[110] I. Grondman, L. Busoniu, G.A. Lopes, R. Babuska, A survey of actor-critic reinforcement learning: standard and natural policy gradients, IEEE Trans. Syst. Man Cybern., Part C, Appl. Rev. 42 (6) (2012) 1291–1307.
[111] L. Liu, L. Xie, X. Zhang, S. Yuan, X. Chen, W. Zhou, H. Li, Q. Tian, Tape: task-agnostic prior embedding for image restoration, in: Proceedings of the European Conference on Computer Vision (ECCV'22), 2022, pp. 447–464.
[112] K. Park, J. Park, H. Oh, B.-T. Zhang, Y. Lee, Learning task-agnostic representation via toddler-inspired learning, in: Proceedings of NeurIPS 2020 Workshop on BabyMind, 2021, pp. 1–6.
[113] Y.-C. Liu, C.-Y. Ma, J. Tian, Z. He, Z. Kira, Polyhistor: parameter-efficient multi-task adaptation for dense vision tasks, in: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'22), 2022, pp. 36889–36901.
[114] J. Andreas, D. Klein, S. Levine, Modular multitask reinforcement learning with policy sketches, in: Proceedings of International Conference on Machine Learning (ICML'17), 2017, pp. 166–175.
[115] A. Zhang, H. Satija, J. Pineau, Decoupling dynamics and reward for transfer learning, arXiv preprint, arXiv:1804.10689, 2018.
[116] X. Jiang, H. Saggar, S.I. Ryu, K.V. Shenoy, J.C. Kao, Structure in neural activity during observed and executed movements is shared at the neural population level, not in single neurons, Cell Rep. 32 (6) (2020) 108006.
[117] R. Poli, W.B. Langdon, Backward-chaining evolutionary algorithms, Artif. Intell. 170 (11) (2006) 953–982.
[118] G. Brandt, W. Haak, C.J. Adler, C. Roth, A. Szécsényi-Nagy, S. Karimnia, S. Möller-Rieker, H. Meller, R. Ganslmeier, S. Friederich, et al., Ancient DNA reveals key stages in the formation of central European mitochondrial genetic diversity, Science 342 (6155) (2013) 257–261.
[119] M. Achtman, M. Wagner, Microbial diversity and the genetic nature of microbial species, Nat. Rev., Microbiol. 6 (6) (2008) 431–440.
[120] C. Guijarro-Clarke, P.W. Holland, J. Paps, Widespread patterns of gene loss in the evolution of the animal kingdom, Nat. Ecol. Evol. 4 (4) (2020) 519–523.
[121] V. Sharma, N. Hecker, J.G. Roscito, L. Foerster, B.E. Langer, M. Hiller, A genomics approach reveals insights into the importance of gene losses for mammalian adaptations, Nat. Commun. 9 (2018) 1215.
[122] J.R. Xue, A. Mackay-Smith, K. Mouri, M.F. Garcia, M.X. Dong, J.F. Akers, M. Noble, X. Li, Z. Consortium, K. Lindblad-Toh, et al., The functional and evolutionary impacts of human-specific deletions in conserved elements, Science 380 (6643) (2023) eabn2253.
[123] E. Todorov, T. Erez, Y. Tassa, Mujoco: a physics engine for model-based control, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 5026–5033.
[124] S. Seung, Connectome: How the Brain's Wiring Makes Us Who We Are, HMH, 2012.
[125] H. Jianye, P. Li, H. Tang, Y. Zheng, X. Fu, Z. Meng, Erl-re^2: efficient evolutionary reinforcement learning with shared state representation and individual policy representation, in: Proceedings of the International Conference on Learning Representations (ICLR'23), 2023, pp. 1–12.
[126] L.Y. Pratt, Discriminability-based transfer between neural networks, in: Proceedings of the Annual Conference on Neural Information Processing Systems, 1992, pp. 204–211.
[127] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E.D. Cubuk, Q. Le, Rethinking pre-training and self-training, in: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'20), 2020, pp. 3833–3845.
[128] E.J. Larson, et al., Evolution: The Remarkable History of a Scientific Theory, vol. 17, Random House Digital, Inc., 2004.
[129] S.J. Gould, The Lying Stones of Marrakech: Penultimate Reflections in Natural History, Random House, 2010.
[130] A.M.D. David Ha, Q.V. Le, Hypernetworks, in: Proceedings of the International Conference on Learning Representations (ICLR'17), 2017, pp. 1–15.
[131] B. Knyazev, M. Drozdzal, G.W. Taylor, A. Romero Soriano, Parameter prediction for unseen deep architectures, in: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'21), 2021, pp. 29433–29448.
[132] B. Knyazev, D. Hwang, S. Lacoste-Julien, Can we scale transformers to predict parameters of diverse imagenet models?, in: Proceedings of International Conference on Machine Learning (ICML'23), 2023, pp. 17243–17259.