Towards General-Purpose Reinforcement Learning Agents

Networked Intelligence for Comprehensive Efficiency (NICE) Lab
College of Information Science and Electronic Engineering
Zhejiang University
http://nice.rongpeng.info/

Sep. 5, 2025

Content

Multi-Task Reinforcement Learning

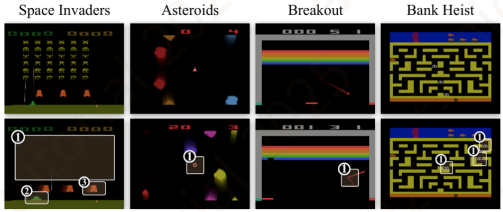
Multi-Task Reinforcement Learning

- **Single-Task Reinforcement Learning (STRL):** to train a specialized policy from scratch to achieve optimal performance on a specific task.
- **Multi-Task Reinforcement Learning (MTRL):** to learn a single, general policy that solves multiple related tasks by leveraging their commonalities to improve overall learning efficiency and generalization.

Advantages of MTRL

- Improved Learning Efficiency and Performance
- Enhanced Generalization and Transfer Learning
- Learning of Effective Shared Representations

Multi-Task Pretraining RL



- **Environment Setup:** Atari 2600 games, each containing various game variants that alter gameplay. And the underlying dynamics of the environment do not change much.

Key Challenges in MTRL

- **High Variance in Importance Sampling:** In asynchronous MTRL, significant policy lag between actors and the learner causes the importance sampling correction ratios to explode or vanish.
- **Conflicting Gradients:** Different tasks often suggest opposing gradient updates to shared network parameters, making simultaneous optimization difficult.

Taiga A A, Agarwal R, Farebrother J, et al. Investigating multi-task pretraining and generalization in reinforcement learning[C]. 2023.

Importance Sampling: Single-Step vs. Trajectory Correction

- **PPO: Single-Step-Level Importance Sampling**
  - The difference between the target policy π_θ and the behavior policy $\pi_{\theta_{old}}$ is small, it's only necessary to correct the single-step update
  - Applies a **clipped, single-step importance sample ratio** to the objective function.

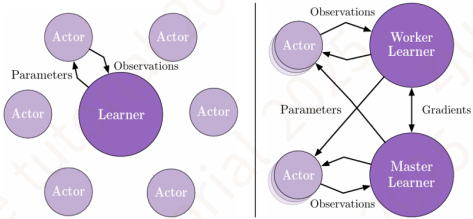
$$L^{CLIP}(\theta) = -\hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \quad \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right], r_t(\theta) = \frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)}$$

- **Trajectory-Level Importance Sampling**
  - To correct a trajectory's value across large policy differences, Importance Sampling requires re-weighting it by the ratio of its probabilities under the target π and behavior μ policies.

$$\frac{P(\tau|\pi)}{P(\tau|\mu)} = \frac{P(s_0)\prod_t \pi(a_t|s_t)P(s_{t+1}|s_t,a_t)}{P(s_0)\prod_t \mu(a_t|s_t)P(s_{t+1}|s_t,a_t)} = \prod_{t=0}^{T-1} \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}$$

  - **Issue:** This **product of importance ratios** over many time steps (e.g., $T-1$ steps) leads to extremely high variance (exploding or vanishing values), making learning unstable and often impossible.

IMPALA: Importance Weighted Actor-Learner Architectures



- **Pre-training:** A single, shared **IMPALA** agent is trained on N variants from the training set.
- **IMPALA:** Consists of a set of actors, repeatedly generating trajectories of experience, and one or more learners that use the experiences sent from actors.

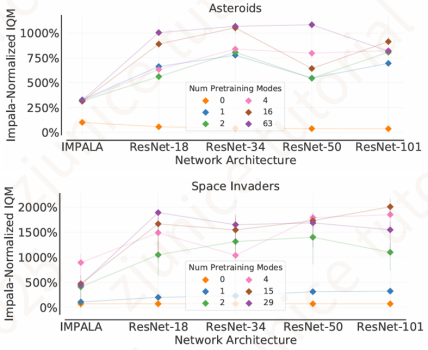
- **V-trace controls the variance through double clipping:** $v_s = V(x_s) + \sum_{t=s}^{T-1} \gamma^{t-s} \left(\prod_{i=s}^{t-1} c_i \right) \delta_t V$
- TD-error: $\delta_t V = \rho_t (r_t + \gamma V(x_{t+1}) - V(x_t))$, clipped weight for the TD error: $\rho_t = \min \left(\bar{\rho}, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)} \right)$, clipped weight for the temporal-cumulative error: $c_t = \min \left(\bar{c}, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)} \right)$.
- **Critic update:** $(v_s - V_\theta(x_s)) \nabla_\theta V(x_s)$
- **Actor update:** $\rho_s \nabla_\omega \log \pi_\omega(a_s|x_s)(r_s + \gamma v_{s+1} - V_\theta(x_s))$

Espeholt, Lasse, et al. "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. "International conference on machine learning. PMLR, 2018

Multi-Task Pretraining RL

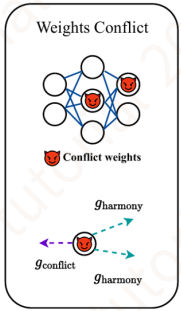
- **Fine-tuning:** The parameters from the pre-trained model are used as initial weights for a new agent. This agent is then fine-tuned online for previously unseen variants.

Multi-task pretraining effectively improves generalization.



The advantages of high-capacity networks are prominent in MTRL.

Gradient Conflict in MTRL



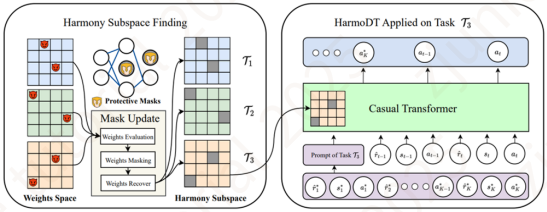
- In MTRL, the problem of gradient conflict arises when multiple different but related tasks share a single policy/value network.

  optimal policy: $\pi^* = \arg\max_\pi \mathbb{E}_{\tau\sim p(\tau)}\mathbb{E}_{a_t\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r^t\right]$

  aggregated gradient: $\hat{g} = \mathbb{E}_{\tau\sim p(\tau)}\nabla\mathcal{L}_T(\theta) = \frac{1}{N}\sum_{i=1}^{N} g_i(\theta)$
- **Harmony Score on a Single Weight:** $(g_i \odot \hat{g})_j = g_{i,j} \times \hat{g}_j$
  Meaning: The consistency between a single task's gradient direction and the average gradient direction for a given weight j.
- **Average Harmony Score:** $\frac{1}{NK}\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{(g_i\odot\hat{g})_j}{|g_{i,j}||\hat{g}_j|}$
  Meaning: A single value that measures the overall degree of harmony across all weights and tasks.
- **Task masks (composed of 0 and 1):** $\bar{g}_i = \nabla\mathcal{L}_{\tau_i}(\theta \odot M^{\tau_i}) \odot M^{\tau_i}, \quad i = 1, 2, \ldots, N$
  Meaning: applied to the global parameters and gradients to derive the task-specific parameter gradient.

How to determine the task masks?

Scaling Decision Transformer for Multi-Task RL



$$\max_{\mathbb{M}} \quad \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}^{\mathcal{T}_i} \left(s_t, \pi \left(\tau_{i,t}^{\text{input}} \mid \theta^{*\mathcal{T}_i} \right) \right) \right], \tag{1}$$

$$\text{s.t.} \quad \theta^* = \arg\min_{\theta} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{DT}(\theta, \mathbb{M}), \tag{2}$$

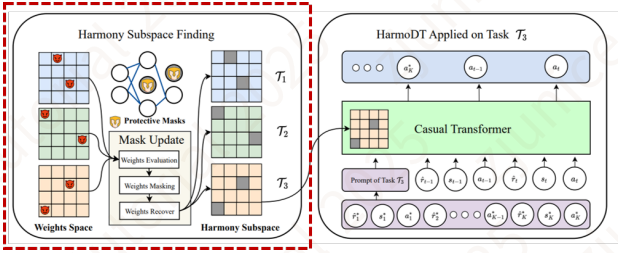
$$\text{where} \quad \theta^{*\mathcal{T}_i} = \theta^* \odot \boldsymbol{M}^{\mathcal{T}_i}, \mathbb{M} = \left\{ \boldsymbol{M}^{\mathcal{T}_i} \right\}_{\mathcal{T}_i \sim p(\mathcal{T})}, \tag{3}$$

Bi-level Optimization Framework

- **Upper-level Optimization (Eq. 1):** Learns the optimal set of task masks.
- **Inner-level Optimization (Eq. 2):** Learns the optimal model parameters, conditioned on the set of masks M provided by the upper level.

Hu, Shengchao, et al. "HarmoDT: Harmony Multi-Task Decision Transformer for Offline Reinforcement Learning." International Conference on Machine Learning. PMLR, 2024.

Upper-level: Find Harmony Subspace

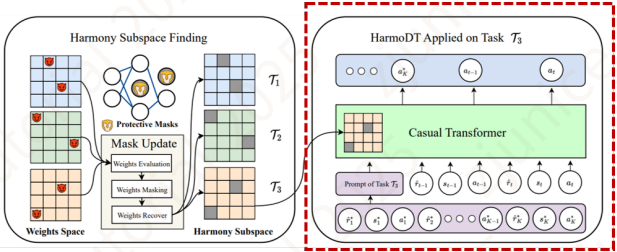


Algorithm 2 Mask Update

Input: A set of tasks $\mathbb{T}$, a set of task masks $\mathbb{M}$, trainable weights vector θ_t and hyper-parameters $\{\lambda, \alpha_t\}$.
for $i = 1, \ldots, N$ **do**
  $\bar{\mathbf{g}}_i = \nabla \mathcal{L}_{\mathcal{T}_i}(\theta \odot \boldsymbol{M}^{\mathcal{T}_i}) \odot \boldsymbol{M}^{\mathcal{T}_i}$.
  $\mathbf{g}_i = \nabla \mathcal{L}_{\mathcal{T}_i}(\theta)$.
end for
$\bar{\mathbf{g}} = \frac{1}{N} \sum_{i=1}^{N} \bar{\mathbf{g}}_i$.
for $i = 1, \ldots, N$ **do**
  // Weights Evaluation
  Calculate $H(\mathcal{T}_i)$ with $\boldsymbol{\lambda}$, $\mathbf{g}_i$ and $\bar{\mathbf{g}}$ as Sec. 4.1.
  // Weights Masking
  $\boldsymbol{M}^{\mathcal{T}_i} = \boldsymbol{M}^{\mathcal{T}_i} - \mathrm{ArgBtmK}_{\alpha_t}(H(\mathcal{T}_i))$.
  // Weights Recovery
  $\boldsymbol{M}^{\mathcal{T}_i} = \boldsymbol{M}^{\mathcal{T}_i} + \mathrm{ArgTopK}_{\alpha_t}(\mathbf{g}_i \odot \bar{\mathbf{g}})$.
end for
Output: $\mathbb{M}$.

- **Weights Evaluation:** Calculate Agreement Score, Importance Score and Harmony Score.
  Agreement Score: $A(\tau_i) = \bar{g}_i \odot \frac{1}{N} \sum_{i=1}^{N} \bar{g}_i$; **Importance Score:** $I_F(\tau_i) = (\nabla \log \mathcal{L}_{\tau_i}(\theta^{\tau_i}) \odot M^{\tau_i})^2$.
  Harmony Score: $H(\tau_i) = \begin{cases} A(\tau_i) + \lambda I_F(\tau_i)/j, & (M^{\tau_i})_j = 1, \\ \infty, & (M^{\tau_i})_j = 0. \end{cases}$
- **Weights Masking:** $\boldsymbol{M}^{\mathcal{T}_i} = \boldsymbol{M}^{\mathcal{T}_i} - \mathrm{ArgBtmK}_{\alpha_t}(H(\mathcal{T}_i))$
- **Weights Recover:** $\boldsymbol{M}^{\mathcal{T}_i} = \boldsymbol{M}^{\mathcal{T}_i} + \mathrm{ArgTopK}_{\alpha_t}\left(\mathbf{g}_i \odot \frac{1}{N} \sum_{i=1}^{N} \bar{\mathbf{g}}_i\right)$

Inner-level: Prompt Decision Transformer



Method	Meta-World 50 Tasks		
#Partition	Near-optimal	Sub-optimal	Params
CARE (online)	$46.12_{\pm 1.30}$	-	1.26 M
PaCo (online)	$54.31_{\pm 1.32}$	-	3.39 M
Soft-M (online)	$53.41_{\pm 0.72}$	-	1.62 M
D2R (online)	$63.55_{\pm 1.22}$	-	1.40 M
MTBC	$60.39_{\pm 0.86}$	$34.53_{\pm 1.25}$	1.74 M
MTIQL	$56.21_{\pm 1.39}$	$43.28_{\pm 0.90}$	1.74 M
MTDIFF-P	$59.53_{\pm 1.12}$	$48.67_{\pm 1.32}$	5.32 M
MTDIFF-P-ONEHOT	$61.32_{\pm 0.89}$	$\mathbf{48.94}_{\pm 0.95}$	5.32 M
MTDT	$20.99_{\pm 2.66}$	$20.63_{\pm 2.21}$	0.87 M
MTDT*	$65.80_{\pm 1.02}$	$42.33_{\pm 1.89}$	1.47 M
Prompt-DT	$45.68_{\pm 1.84}$	$39.76_{\pm 2.79}$	0.87 M
Prompt-DT*	$\mathbf{69.33}_{\pm 0.89}$	$48.40_{\pm 0.16}$	1.47 M
HarmoDT-R (ours)	$\mathbf{75.39}_{\pm 1.18}$	$\mathbf{53.80}_{\pm 1.07}$	1.47 M
HarmoDT-M(ours)	$\mathbf{80.33}_{\pm 0.97}$	$\mathbf{57.20}_{\pm 0.73}$	1.47 M
HarmoDT-F(ours)	$\mathbf{82.20}_{\pm 0.40}$	$\mathbf{57.20}_{\pm 0.68}$	1.47 M

- The inner-loop for task parameter optimization uses the **Prompt Decision Transformer** algorithm.

$$\mathcal{L}_{DT} = \mathbb{E}_{\tau_{i,t}^{input} \sim \mathcal{D}_i}\left[\frac{1}{K}\sum_{m=t-K+1}^{t}\left(\mathbf{a}_{i,m} - \pi\left(\tau_i^*, \tau_{i,m}\right)\right)^2\right]$$

$$\tau_{i,t}^{input} = \left(\hat{r}_{i,1}^*, \mathbf{s}_{i,1}^*, \mathbf{a}_{i,1}^*, \ldots, \hat{r}_{i,K^*}^*, \mathbf{s}_{i,K^*}^*, \mathbf{a}_{i,K^*}^*\right.$$
$$\left.\hat{r}_{i,t-K+1}, \mathbf{s}_{i,t-K+1}, \mathbf{a}_{i,t-K+1}, \ldots, \hat{r}_{i,t}, \mathbf{s}_{i,t}, \mathbf{a}_{i,t}\right)$$

$$\theta_{t+1} = \theta_t - \eta \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \nabla \mathcal{L}_{\mathcal{T}_i}\left(\theta \odot \boldsymbol{M}^{\mathcal{T}_i}\right) \odot \boldsymbol{M}^{\mathcal{T}_i}$$

- For unseen tasks that differ from training tasks but share identical states and transitions:

$$\hat{M}_j = \begin{cases} 0, & \sum_{i=1}^{N} M_{i,j} \leq \text{ thresh} \\ 1, & \sum_{i=1}^{N} M_{i,j} > \text{ thresh} \end{cases}$$

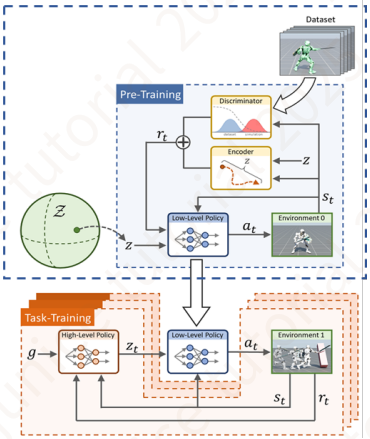
Generative Pre-trained Reinforcement Learning

Generative Pre-trained RL

Features of Generative Pre-trained RL

- **Generative Skill Representation:** skills are encoded into a latent space, where sampling a point can generate a complete, coherent behavior.
- **Hierarchical and Modular Architecture:** uses a high-level policy to call and combine a pre-trained, low-level skill policy to solve complex tasks.
- **Knowledge Freezing and Reuse:** the pre-trained low-level skill model is frozen, and the high-level policy adapts to new tasks by learning how to command it.

GAN-based Pre-trained RL



1. Pre-Training Stage: Learning Low-Level Skills

- train a **low-level skill-conditioned policy** $\pi(a|s,z)$ that maps latent variables z to behaviors resembling those in the dataset. Training objective combines **imitation** and **skill discovery**:

$$\max_D -D_{JS}(d^\pi(s,s')||d^M(s,s')) + \beta I(s,s';z|\pi)$$

- **Imitation Objective:** achieved through a **GAN-like discriminator** $D(s,s')$

$$\min_D -\mathbb{E}_{d^M(s,s')}[\log D(s,s')] - \mathbb{E}_{d^\pi(s,s')}[\log(1-D(s,s'))]$$

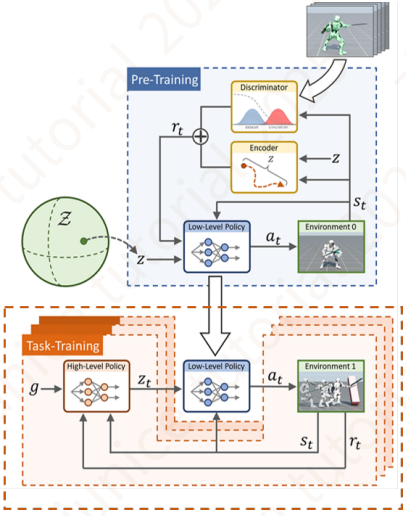
- **Skill Discovery Objective:** develop a diverse set of skills by maximizing the mutual information between the latent skill variable z and the resulting behaviors:

$$\max \beta I(s,s';z|\pi) \Rightarrow I(s,s';z|\pi) = \mathcal{H}(z) - \mathcal{H}(z|s,s',\pi)$$

$$-\mathcal{H}(z|s,s',\pi) = \mathbb{E}_{p(z)p(s,s'|\pi,z)}[\log p(z|s,s',\pi)]$$

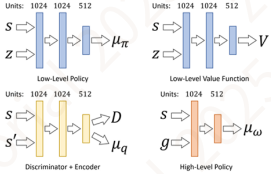
Peng, Xue Bin, et al. "Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters." ACM Transactions On Graphics (TOG) 41.4 (2022).

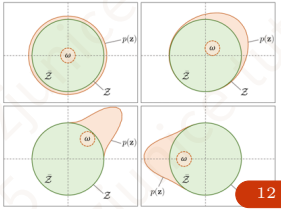
GAN-based Pre-trained RL



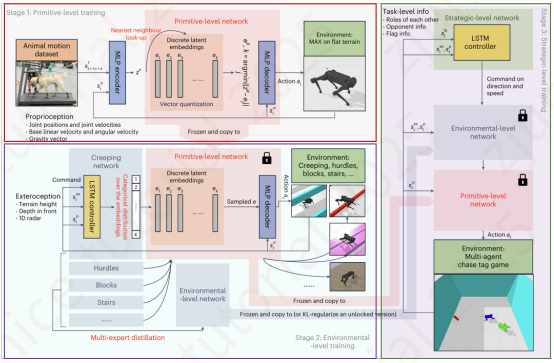
2. Task-Training Stage: Reusing Learned Skills

- **Hierarchical Control:** after pre-training, the low-level policy $\pi(a|s,z)$ is **fixed** and reused as a low-level controller.
- **High-Level Policy:** for each new downstream task, a task-specific **high-level policy** $z = \omega(z|s,g)$ is trained.
- **High-Level Action Space:** the high-level policy's action space is the **unnormalized latent space** $\bar{z}$, allowing it to balance exploration and exploitation by adjusting the mean of its sampling distribution: $\omega(\bar{z}|s,g) = \mathcal{N}(\mu_\omega(s,g), \Sigma_\omega)$





VAE-based Pre-trained RL



- **Primitive-level training (PMC)**
  Using the VQ-PMC pre-training method, the robot learns lifelike basic movements—like walking, running, and jumping—by imitating animal motion data via RL.
- **Environmental-level training (EPMC)**
  A new network is trained on top of the pre-trained primitive model to handle environmental inputs. It uses multi-expert distillation to merge various expert skills into a single model for complex traversal tasks.
- **Strategic-level training (SEPMC)**
  Reuse the PMC and EPMC networks to train a multi-agent reinforcement learning control algorithm.

Han, Lei, et al. "Lifelike agility and play in quadrupedal robots using reinforcement learning and generative pre-trained models." Nature Machine Intelligence 6.7 (2024): 787-79

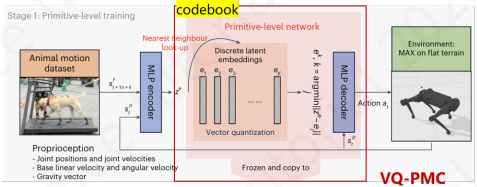
VQ-PMC

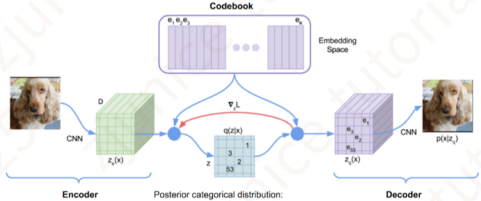
- VQ-PMC Borrows from VQ-VAE: Using Discrete Latent Space Embeddings to Generate Actions
- **VAE:** Uses a neural network to find a non-linear latent representation of a given data distribution. The network consists of an encoder and a decoder.

$$L = ||x - \text{decoder}(z_e(x))||_2^2 + \text{KL}(z_e(x)||p(z))$$

- **VQ-VAE:** A discrete latent space representation; consists of an encoder, a decoder, and a codebook.
  - It extends the standard **autoencoder** by adding a discrete **codebook** component.

$$L = \left\|x - \text{decoder}\left(z_e(x) + \text{sg}\left(z_q(x) - z_e(x)\right)\right)\right\| + \alpha \left\|\text{sg}\left(z_e(x)\right) - z_q(x)\right\|_2^2 + \beta \left\|z_e(x) - \text{sg}\left(z_q(x)\right)\right\|_2^2$$

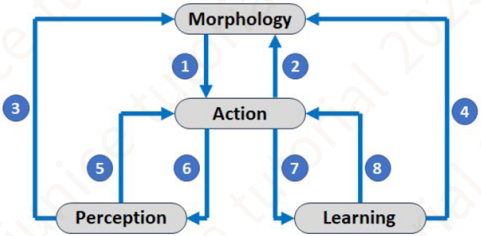




Embodied Intelligence

Embodied Intelligence

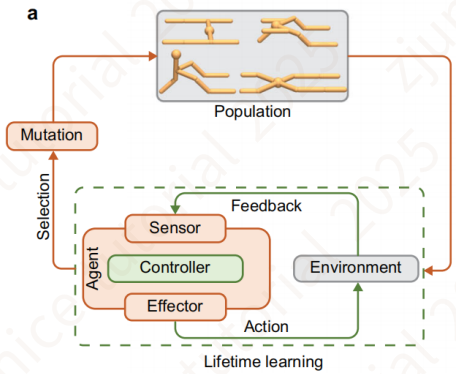
- Embodied intelligence is the computational approach to the design and understanding of intelligent behavior in embodied and situated agents through the consideration of the strict coupling between the agent and its environment, mediated by the constraints of the agent's own body, perceptual and motor system, and brain.
- This technology is considered an important pathway to Artificial General Intelligence (AGI).



① Action Generation Based on Morphology
② Morphological Control Based on Action
③ Perception-Driven Morphology Transformation
④ Learning-Driven Morphology Optimization
⑤ Perception-Driven Action Generation
⑥ Action-Driven Perception Improvement
⑦ Action-Driven Embodied Learning
⑧ Action Generation Based on Learning

Liu H, Guo D, Cangelosi A. Embodied intelligence: A synergy of morphology, action, perception and learning[J]. ACM Computing Surveys, 2025, 57(7): 1-36.

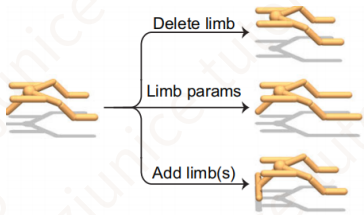
Deep Evolutionary RL: Outer Loop - Evolution



DERL framework

Outer Loop: Evolution

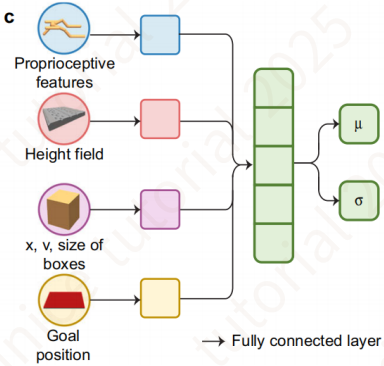
- **Mutation Operations:** The morphology is altered through a set of mutation operations, including adding or deleting limbs and modifying physical parameters.



Morphology Mutations

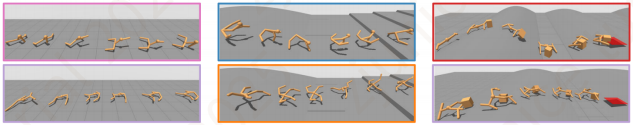
Gupta A, Savarese S, Ganguli S, et al. Embodied intelligence via learning and evolution[J]. Nature communications, 2021, 12(1): 5721.

Deep Evolutionary RL: Inner Loop - RL



Inner Loop: Reinforcement Learning

- At each timestep, the agent individually encodes internal and external sensory data. This is passed to a fully connected layer that outputs two parameters for the neural controller. The controller's policy is then optimized via PPO in a lifelong learning framework.
- Use three progressively complex environments to evolve morphologies and learn agent controllers: (1) Flat Terrain (FT), (2) Variable Terrain (VT), and (3) Non-prehensile Manipulation in Variable Terrain (MVT).



Conclusion

Conclusion

I have talked about

- **Multi-Task RL (MTRL):**
  Improves generalization by training on diverse tasks but must resolve the core challenge of high variance in importance sampling and gradient conflict. Advanced methods solve these by using IMPALA and identifying a shared ”harmony subspace” for parameter reuse.
- **Generative Pre-trained RL:**
  Boosts generalization by first creating a reusable library of low-level skills with generative models (GANs/VAEs). A high-level policy then learns to combine these skills to solve new, complex tasks.
- **Embodied Intelligence:**
  As a key pathway to AGI, this approach integrates an agent’s physical morphology with its control policy. Techniques like Deep Evolutionary Reinforcement Learning (DERL) co-optimize the agent’s body via evolution and its ”brain” via RL, creating more adaptive agents.

Thank you

Networked Intelligence for Comprehensive Efficiency (NICE) Lab
College of Information Science and Electronic Engineering
Zhejiang University
https://nice.rongpeng.info