Reinforcement Learning for Improving Agent Design

David Ha*
Google Brain, Tokyo, Japan
hadavid@google.com

Keywords
Neuroevolution, deep reinforcement learning, evolution strategies, generative design

Abstract In many reinforcement learning tasks, the goal is to learn a policy to manipulate an agent, whose design is fixed, to maximize some notion of cumulative reward. The design of the agent's physical structure is rarely optimized for the task at hand. In this work, we explore the possibility of learning a version of the agent's design that is better suited for its task, jointly with the policy. We propose an alteration to the popular OpenAI Gym framework, where we parameterize parts of an environment, and allow an agent to jointly learn to modify these environment parameters along with its policy. We demonstrate that an agent can learn a better structure of its body that is not only better suited for the task, but also facilitates policy learning. Joint learning of policy and structure may even uncover design principles that are useful for assisted-design applications.

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023

1 Introduction

Embodied cognition [3, 40, 58] is the theory that an organism's cognitive abilities are shaped by its body. It is even argued that an agent's cognition extends beyond its brain, and is strongly influenced by aspects of its body and also the experiences from its various sensorimotor functions [25, 73]. Evolution plays a vital role in shaping an organism's body to adapt to its environment; the brain with its ability to learn is only one of many body components that are coevolved [11, 47]. We can observe embodiment in nature by observing that many organisms exhibit complex motor skills, such as the ability to jump [12] or swim [10], even after brain death.

While evolution shapes the overall structure of the body of a particular species, an organism can also change and adapt its body to its environment during its life (see Figure 1). For instance, professional athletes spend their lives body training while also improving specific mental skills required to master a particular sport [68]. In everyday life, regular exercise not only strengthens the body but also improves mental conditions [22, 49]. We not only learn and improve our skills and abilities during our lives, but also learn to shape our bodies for the lives we want to live.

We are interested in investigating embodied cognition within the reinforcement learning (RL) framework. Most baseline tasks [36, 66] in the RL literature test an algorithm's ability to learn a policy to control the actions of an agent, with a predetermined body design, to accomplish a given task inside an environment. The design of the agent's body is rarely optimal for the task, and sometimes even intentionally designed to make policy search challenging. In this work, we explore enabling learning versions of an agent's body that are better suited for its task, jointly with its policy.

* Corresponding author.

© 2019 by the Massachusetts Institute of Technology. Published under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

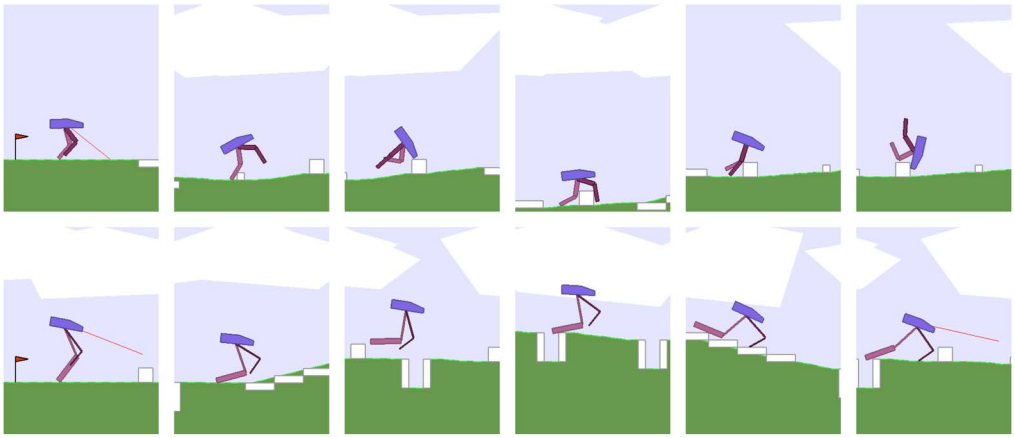

Figure 1. Learning to navigate over randomly generated terrain in `BipedalWalkerHardcore-v2` environment (top). Agent learns a better body design while jointly learning to navigate (bottom).

We demonstrate that an agent can learn a better structure of its body that not only is better for its task, but also facilitates policy learning. We can even optimize our agent's body for certain desired characteristics, such as material usage.[1] Our approach may help uncover design principles useful for assisted design.

Furthermore, we believe the ability to learn useful morphology is an important area for the advancement of AI. Although morphology learning originated in the field of evolutionary computation, there has also been great advances in RL in recent years, and we believe much of what happens in ALife should in principle be of interest to the RL community and vice versa, since learning and evolution are just two sides of the same coin.

We believe that conducting experiments using standardized simulation environments facilitates the communication of ideas across disciplines, and for this reason we design our experiments based on applying ideas from ALife, namely morphology learning, to standardized tasks in the OpenAI Gym environment, a popular testbed for conducting experiments in the RL community. We decide to use standardized Gym environments such as Ant (based on the Bullet physics engine) and Bipedal Walker (based on Box2D), not only for their simplicity, but also because their difficulty is well understood due to the large number of RL publications that use them as benchmarks. As we shall see later, the `BipedalWalkerHardcore-v2` task, while simple looking, is especially difficult to solve with modern deep RL methods. By applying simple morphology learning concepts from ALife, we are able to make a difficult task solvable with much fewer computation resources. We also made the code for augmenting OpenAI Gym for morphology learning, along with all pretrained models for reproducing results in this article, available at https://github.com/hardmaru/astool.

We hope this article can serve as a catalyst to precipitate a cultural shift in both fields and encourage researchers to open up their minds to each other. By drawing ideas from ALife and demonstrating them in the OpenAI Gym platform used by RL, we hope this work can set an example to bring both the RL and ALife communities closer together to find synergies and push the AI field forward.

2 Related Work

There is a broad literature in evolutionary computation, artificial life, and robotics devoted to studying and modeling embodied cognition [47]. In 1994, Karl Sims demonstrated that artificial evolution can produce novel morphology that resembles organisms observed in nature [59, 60]. Subsequent

1 Videos of results are at https://designrl.github.io/.

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023

```
def rollout(agent, env):
  obs = env.reset()
  done = False
  cumulative_reward = 0
  while not done:
    a = agent.action(obs)
    obs, r, done = env.step(a)
    cumulative_reward += r
  return cumulative_reward
```

```
def rollout(agent, env_params, env):
  env.augment(env_params)
  obs = env.reset()
  done = False
  cumulative_reward = 0
  while not done:
    a = agent.action(obs)
    obs, r, done = env.step(a)
    r = augment_reward(r, env_params)
    cumulative_reward += r
  return cumulative_reward
```

Figure 2. OpenAI Gym framework for rolling out an agent in an environment (left). We propose an alteration where we parameterize parts of an environment, and allow an agent to modify its environment before a rollout and also to augment its reward based on these parameters (right).

works further investigated morphology evolution [4, 8, 9, 11, 37, 44, 64, 65, 70], modular robotics [39, 45, 48, 75], and evolving soft robots [17, 20], using indirect encoding [5, 6, 7, 23, 61].

In passive dynamics studies robot designs that rely on natural swings of motion of body components instead of deploying and controlling motors at each joint [18, 19, 41, 46]. Notably, the artist Theo Jansen [33] also employed evolutionary computation to design physical *strandbeests* that can walk on their own, consuming only wind energy, to raise environmental awareness.

Recent works in robotics investigate simultaneously optimizing body design and control of a legged robot [29, 30] using constraint-based modeling, which is related to our RL-based approach. Related to our work, [1, 24] employ CMA-ES [31] to optimize over both the motion control and physical configuration of agents. A related recent work [52, 53] employs RL to learn both the policy and design parameters in an alternating fashion, where a single shared policy controls a distribution of different designs; in this work we simply treat both policy and design parameters the same way.

3 Method

In this section, we describe the method used for learning a version of the agent's design better suited for its task, jointly with its policy. In addition to the weight parameters of our agent's policy network, we will also parameterize the agent's environment, which includes the specification of the agent's body structure. This extra parameter vector, which may govern the properties of items such as the width, length, radius, mass, and orientation of an agent's body parts and their joints, will also be treated as a learnable parameter. Hence the weights w we need to learn will be the parameters of the agent's policy network combined with the environment's parameterization vector. During a rollout, an agent initialized with w will be deployed in an environment that is also parameterized with the same parameter vector w.

The goal is to learn w to maximize the expected cumulative reward, $E[R(w)]$, of an agent acting on a policy with parameters w in an environment governed by the same w. In our approach, we search for w using a population-based policy gradient method based on Section 6 of Williams' 1992 REINFORCE [72]. The next section provides an overview of this algorithm, which is shown in Figure 2.

Armed with the ability to change the design configuration of an agent's own body, we also wish to explore encouraging the agent to challenge itself by rewarding it for trying more difficult designs. For instance, carrying the same payload using smaller legs may result in a higher reward than using larger legs. Hence the reward given to the agent may also be augmented according to its parameterized environment vector. We will discuss reward augmentation to optimize for desirable design properties later on in more detail in Section 4.2.

3.1 Overview of Population-based Policy Gradient Method (REINFORCE)

In this section we provide an overview of the population-based policy gradient method described in Section 6 of Williams' REINFORCE [72] article for learning a parameter vector w in

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023

a reinforcement learning environment. In this approach, w is sampled from a probability distribution $\pi(w, \theta)$ parameterized by θ. We define the expected cumulative reward R as

$$J(\theta) = E_\theta[R(w)] = \int R(w) \ \pi(w, \theta) \ dw. \tag{1}$$

Using the log-likelihood trick allows us to write the gradient of $J(\theta)$ with respect to θ:

$$\nabla_\theta J(\theta) = E_\theta[R(w) \ \nabla_\theta \log \pi(w, \theta)]. \tag{2}$$

In a population of size N, where we have solutions w^1, w^2, …, w^N, we can estimate this as

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} R\left(w^i\right) \ \nabla_\theta \log \pi\left(w^i, \theta\right). \tag{3}$$

With this approximated gradient $\nabla_\theta J(\theta)$, we then can optimize θ using gradient ascent,

$$\theta \rightarrow \theta + \alpha \nabla_\theta J(\theta), \tag{4}$$

and sample a new set of candidate solutions w from updating the pdf using the learning rate α. We follow the approach in REINFORCE where π is modeled as a factored multivariate normal distribution. Williams derived closed-form formulas for the gradient $\nabla_\theta \log \pi(w^i, \theta)$. In this special case, θ will be the set of parameters with mean μ and standard deviation σ. Therefore, each element of a solution can be sampled from a univariate normal distribution $w_j \sim N(\mu_j, \sigma_j)$. Williams derived the closed-form formulas for $\nabla_\theta \log N(w^i, \theta)$ in Equation 3, for each individual μ and σ element of the vector θ on each solution i in the population:

$$\nabla_{\mu_j} \log N\left(w^i, \theta\right) = \frac{w_j^i - \mu_j}{\sigma_j^2}, \qquad \nabla_{\sigma_j} \log N\left(w^i, \theta\right) = \frac{\left(w_j^i - \mu_j\right)^2 - \sigma_j^2}{\sigma_j^3}. \tag{5}$$

(For clarity, we use subscript j to count across parameter space in w, and this is not to be confused with the superscript i used to count across each sampled member of the population of size N.) Combining Equation 5 with Equation 4, we can update μ_j and σ_j at each generation via a gradient update.

We note that there is a connection between population-based REINFORCE, which is a population-based policy gradient method, and particular formulations of evolution strategies [50, 56], namely ones that are not elitist. For instance, natural evolution strategies (NESs)

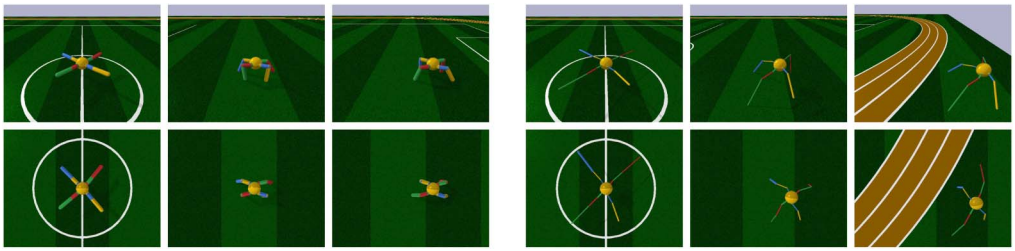

Figure 3. Agent learning a policy to navigate forward in `RoboschoolAnt-v1` environment (left). Agent develops longer, thinner legs while supporting the same body during training (right).

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023

Table 1. Learned agent body for RoboschoolAnt-v1 as a percentage of the original design specification.

	Top Left		Top Right		Bottom Left		Bottom Right	
	Length	Radius	Length	Radius	Length	Radius	Length	Radius
Top	141%	33%	141%	25%	169%	35%	84%	51%
Middle	169%	26%	164%	26%	171%	31%	140%	29%
Bottom	174%	26%	168%	50%	173%	29%	133%	38%

[57, 71] and OpenAI-ES [51] are closely based on Section 6 of REINFORCE. There is also a connection between natural gradients (computed using NESs) and CMA-ES [31]. We refer to Akimoto et al. [2] for a detailed theoretical treatment and discussion of the connection between CMA-ES and natural gradients.

4 Experiments

In this work, we experiment on the continuous control environment `RoboschoolAnt-v1` [36], based on the open source Bullet [21] physics engine, and also `BipedalWalker-v2` from the Box2D [16] section of the OpenAI Gym [13] set of environments. For simplicity, we first present results of anecdotal examples obtained over a single representative experimental run to convey qualitative results such as morphology and its relationship to performance. A more comprehensive quantitative study based on multiple runs using different random seeds will be presented in Section 4.3.

The `RoboschoolAnt-v1`[2] environment features a four-legged agent called the *Ant*. The body is supported by four legs, and each leg consists of three parts, which are controlled by two motor joints. The bottom left diagram of Figure 3 describes the initial orientation of the agent. The length of each part of a leg is controlled by the Δx and Δy distances from its joint connection. A size parameter also controls the radius of each leg part.

In our experiment, we keep the volumetric mass density of all materials, along with the parameters of the motor joints, identical to the original environment, and allow the 36 parameters (3 parameters per leg part, 3 leg parts per leg, 4 legs in total) to be learned. In particular, we allow each part to be scaled to a range of $\pm 75\%$ of its original value. This allows us to keep the sign and direction for each part to preserve the original intended structure of the design.

Figure 3 illustrates the learned agent design compared with the original design. With the exception of one leg part, it learns to develop longer, thinner legs while jointly learning to carry the body across the environment. While the original design is symmetric, the learned design (Table 1) breaks symmetry and biases towards larger rear legs while jointly learning the navigation policy using an asymmetric body. The original agent achieved an average cumulative score of 3447 ± 251 over 100 trials, compared to 5789 ± 479 for an agent that learned a better body design.

The bipedal walker series of environments is based on the Box2D [16] physics engine. Guided by lidar sensors, the agent is required to navigate across an environment of randomly generated terrain within a time limit, without falling over. The agent's payload—its head—is supported by two legs. The top and bottom parts of each leg are controlled by two motor joints. In the easier `BipedalWalker-v2` [34] environment, the agent needs to travel across small random variations of a flat terrain. The task is considered solved if an agent obtains an average score greater than 300 points over 100 rollouts.

2 A compatible version of this environment is also available in PyBullet [21], which was used for visualization.

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023

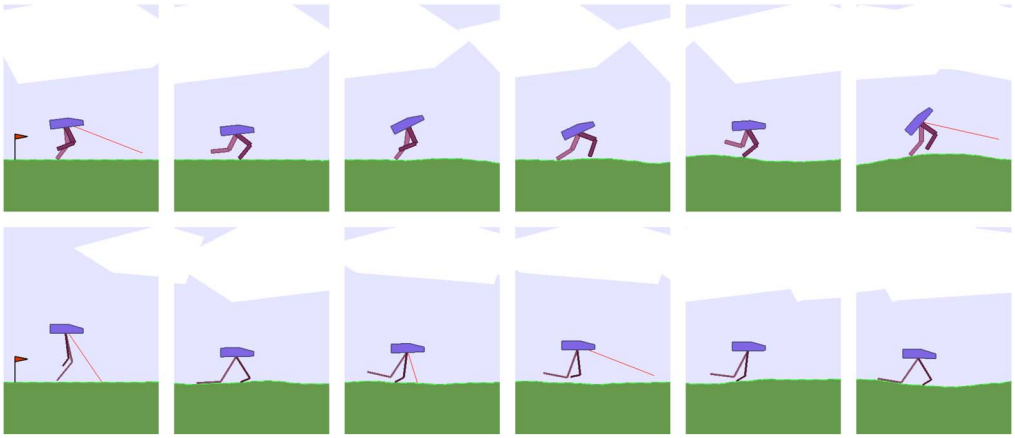

Figure 4. Agent learning a policy to navigate forward in `BipedalWalker-v2` environment (top). Agent learns a body to allow it to bounce forward efficiently (bottom).

Keeping the head payload constant, and also keeping the density of materials and the configuration of the motor joints the same as in the original environment, we only allow the lengths and widths for each of the four leg parts to be learnable, subject to the same range limit of ±75% of the original design. In the original environment in Figure 4 (top), the agent learns a policy that is reminiscent of a joyful skip across the terrain, achieving an average score of 347. In the learned version in Figure 4 (bottom), the agent's policy is to hop across the terrain using its legs as a pair of springs, achieving a score of 359.

In our experiments, all agents were implemented using three-layer fully connected networks with tanh activations. The agent in `RoboschoolAnt-v1` has 28 inputs and 8 outputs, all bounded between −1 and +1, with hidden layers of 64 and 32 units. The agents in `BipedalWalker-v2` and `BipedalWalkerHardcore-v2` have 24 inputs and 4 outputs, all bounded between −1 and +1, with two hidden layers of 40 units each.

Our population-based training experiments were conducted on 96-core CPU machines. Following the approach described in [28], we used a population size of 192, and had each agent perform the task 16 times with different initial random seeds. The agent's reward signal used by the policy gradient method is the average reward of the 16 rollouts. The most challenging BipedalWalkerHardcore agents were trained for 10,000 generations, while the easier BipedalWalker and Ant agents were trained for 5000 and 3000 generations, respectively. As done in [28], we save the parameters of the agent that achieves the best average cumulative reward during its entire training history.

4.1 Joint Learning of Body Design Facilitates Policy Learning

Learning a better version of an agent's body not only helps achieve better performance, but also enables the agent to jointly learn policies more efficiently. We demonstrate this in the much more challenging `BipedalWalkerHardcore-v2` [35] version of the task. Unlike the easier version, the agent must also learn to walk over obstacles, travel up and down hilly terrain, and even jump over pits. Figure 5 illustrates the original and learnable versions of the environment.[3]

In this environment, our agent generally learns to develop longer, thinner legs, with the exception of the rear leg, where it developed a thicker lower limb to serve as a useful stability function for

3 As of writing, two methods have been reported to solve this task. Population-based training [28] (our baseline) solves this task in 40 hours on a 96-CPU machine, using a small feedforward policy network. A3C [43], adapted for continuous control [26], solves the task in 48 hours on a 72-CPU machine, but requires an LSTM [32] policy.

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023

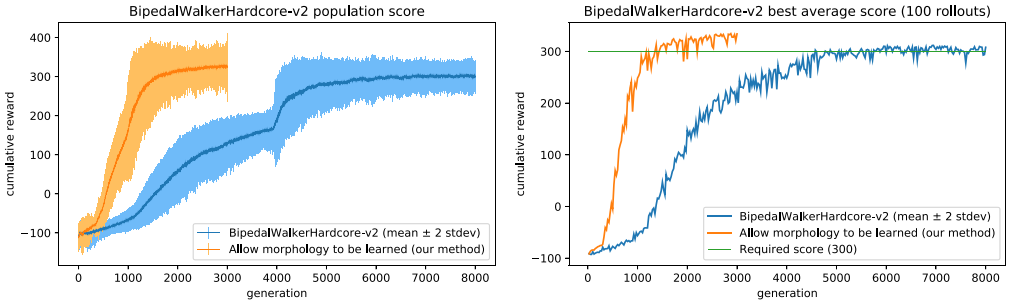

Figure 5. Population-based training curves for both versions of `BipedalWalkerHardcore-v2` (left). Plot of performance of best agent in the population over 100 random trials (right). Original version solved in under 4600 generations (40 hours). By allowing morphology to be learned, the task is solved in under 1400 generations (12 hours).

navigation. Its front legs, which are smaller and more maneuverable, also act as sensors for dangerous obstacles ahead, which complement its lidar sensors. While learning to develop this newer structure, it jointly learns a policy to solve the task in 30% of the time it took the original, static version of the environment. The average scores over 100 rollouts for the learnable version is 335 ± 37, compared to the baseline score of 313 ± 53. The full results are summarized in Table 2.

4.2 Optimize for Both the Task and the Desired Design Properties

Allowing an agent to learn a better version of its body obviously enables it to achieve better performance. But what if we want to give back some of the additional performance gains, and optimize also for desirable design properties that might not generally be beneficial for performance? For instance, we may want our agent to learn a design that utilizes the least amount of materials while still achieving satisfactory performance on the task. Here, we reward an agent for developing legs that are smaller in area, and augment its reward signal during training by scaling the rewards by a utility factor of $1 + \log(\frac{\text{orig. leg area}}{\text{new leg area}})$. Augmenting the reward encourages development of smaller legs. (See Figure 6.)

This reward augmentation resulted in a much smaller agent that is still able to support the same payload. In `BipedalWalker`, given the simplicity of the task, the agent's leg dimensions eventually shrink to near the lower bound of ~25% of the original dimensions, with the exception of the

Table 2. Summary of results for bipedal walker environments. Scaled Box2D dimensions reported.

			Top leg 1		Bottom leg 1		Top leg 2		Bottom leg 2	
`BipedalWalker-v2`	Avg. score	leg area	*w*	*h*	*w*	*h*	*w*	*h*	*w*	*h*
Original	347 ± 0.9	100%	8.0	34.0	6.4	34.0	8.0	34.0	6.4	34.0
Learnable	359 ± 0.2	33%	2.0	57.3	1.6	46.0	2.0	48.8	1.6	18.9
Reward smaller leg	323 ± 68	8%	2.0	11.5	1.6	10.6	2.0	11.4	1.6	10.2

			Top leg 1		Bottom leg 1		Top leg 2		Bottom leg 2	
`BipedalWalkerHardcore-v2`	Avg. score	leg area	*w*	*h*	*w*	*h*	*w*	*h*	*w*	*h*
Original	313 ± 53	100%	8.0	34.0	6.4	34.0	8.0	34.0	6.4	34.0
Learnable	335 ± 37	95%	2.7	59.3	10.0	58.9	2.3	55.5	1.7	34.6
Reward smaller leg	312 ± 69	27%	2.0	35.3	1.6	47.1	2.0	36.2	1.6	26.7

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023

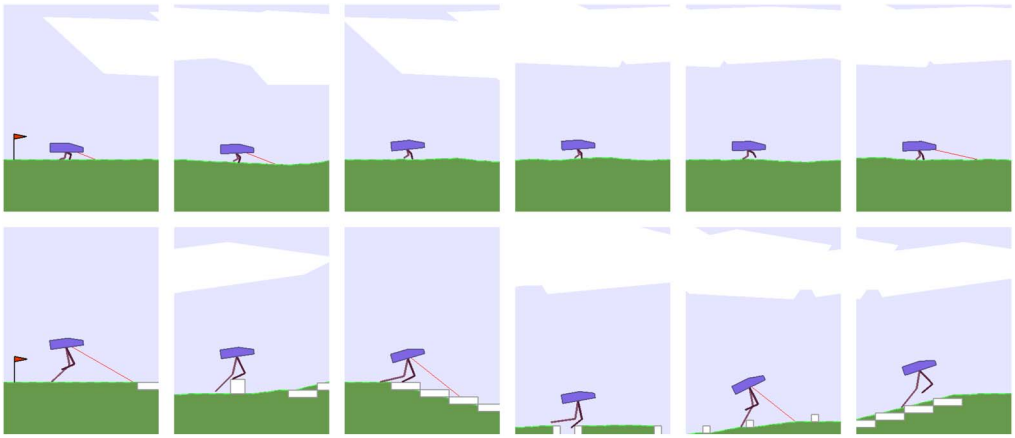

Figure 6. Agent rewarded for smaller legs for the task in `BipedalWalker-v2` environment (top). Agent learns the smallest pair of legs that still can solve `BipedalWalkerHardcore-v2` (bottom).

heights of the top leg parts, which settled at ~35% of the initial design, while still achieving an average (unaugmented) score of 323 ± 68. For this task, the leg area used is 8% of the original design.

However, the agent is unable to solve the more difficult `BipedalWalkerHardcore` task using a similar small body structure, due to the various obstacles presented. Instead, it learns to set the width of each leg part close to the lower bound, and instead to learn the shortest heights of each leg part required to navigate, achieving a score of 312 ± 69. Here, the leg area used is 27% of the original.

4.3 Results over Multiple Experimentals Runs

In the previous subsections, for simplicity, we have presented results over a single representative experimental run to convey qualitative results such as a morphology description corresponding to the average score achieved. Running the experiment from scratch with a different random seed may generate different morphology designs and different policies that lead to different performance scores. To demonstrate that morphology learning does indeed improve the performance of the agent over multiple experimental runs, we ran each experiment 10 times and report the full range

Table 3. Summary of results for each experiment over 10 independent runs.

Experiment	Statistics of average scores over 10 independent runs
(a) Ant	3139 ± 189.3
(b) Ant + morphology	5267 ± 631.4
(c) Biped	345 ± 1.3
(d) Biped + morphology	354 ± 2.2
(e) Biped + morphology + smaller Leg	330 ± 3.9
(f) Biped hardcore	300 ± 11.9
(g) Biped hardcore + morphology	326 ± 12.7
(h) Biped hardcore + morphology + smaller leg	312 ± 11.9

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023

Table 4. Full results from each of the 10 experimental trials. Each number is the average score of the trained agent over 100 rollouts in the environment.

Experiment	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
(a) Ant	3447	3180	3076	3255	3121	3223	3130	3096	3167	2693
(b) + morphology	5789	6035	5784	4457	5179	4788	4427	5253	6098	4858
(c) Biped	347	343	347	346	345	345	345	346	346	344
(d) + morphology	359	354	353	354	353	352	353	352	353	356
(e) + smaller leg	323	327	327	331	330	331	333	329	337	333
(f) Biped hardcore	313	306	300	283	311	295	307	309	292	279
(g) + morphology	335	331	330	330	332	292	327	331	316	330
(h) + smaller leg	312	320	314	318	307	314	316	281	319	324

of average scores obtained in Table 3 and Table 4. From multiple independent experimental runs, we see that morphology learning consistently produces higher scores over the normal task.

We also visualize the variations of morphology designs over different runs in Figure 7 to get a sense of the variations of morphology that can be discovered during training. As these models may take up to several days to train for a particular experiment on a powerful 96-core CPU machine, it may be costly for the reader to fully reproduce the variation of results here, especially when 10 machines running the same experiment with different random seeds are required. We also include all pretrained models from multiple independent runs in the GitHub repository containing the code to reproduce this article. The interested reader can examine the variations in more detail using the pretrained models.

5 Discussion and Future Work

We have shown that using a simple population-based policy gradient method for allowing an agent to learn not only the policy, but also a small set of parameters describing the environment, such as

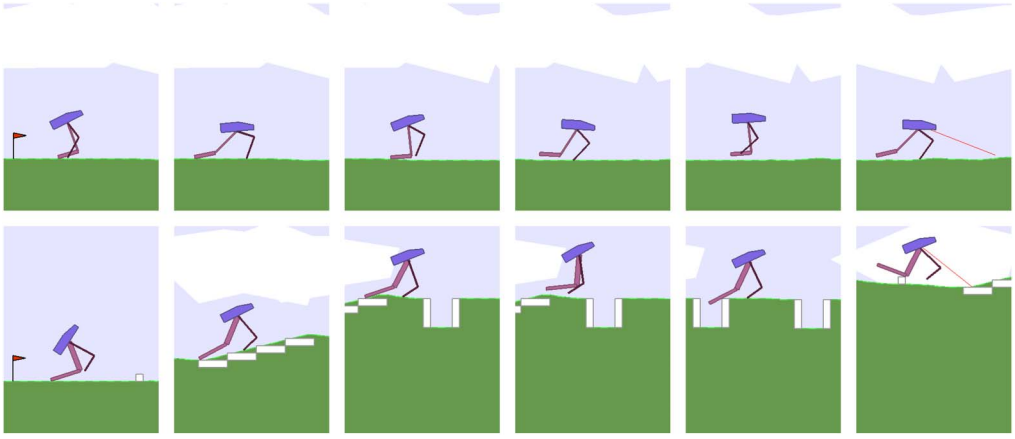

Figure 7. Examples of learned morphology in run #9. Biped + morphology (top) develops a thicker but short rear lower limb, unlike the agent in Figure 3. Biped hardcore + morphology (bottom) develops a larger rear leg, but unlike the agent in Figure 1, its thigh is larger than the lower limb.

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023

its body, offers many benefits. By allowing the agent's body to adapt to its task within some constraints, the agent can not only learn policies that are better for its task, but also learn them more quickly.

The agent may discover design principles during this joint process of body and policy learning. In both `RoboschoolAnt` and `BipedalWalker` experiments, the agent has learned to break symmetry and learn larger rear limbs to facilitate their navigation policies. While also optimizing for material usage for BipedalWalker's limbs, the agent learns that it can still achieve the desired task even on setting the size of its legs to the minimum allowable. Meanwhile, for the much more difficult `BipedalWalkerHardcore-v2` task, the agent learns the appropriate length of its limbs required for the task while still minimizing the material usage.

This approach may lead to useful applications in machine-learning-assisted design, in the spirit of [14, 15]. While not directly related to agent design, machine-learning-assisted approaches have been used to procedurally generate game environments that can also facilitate policy learning of game-playing agents [27, 42, 63, 67, 69]. Game designers can optimize the designs of game character assets while at the same time being able to constrain the characters to keep the essence of their original forms. Optimizing character design may complement existing work on machine-learning-assisted procedural content generation for game design. By framing the approach within the popular OpenAI Gym framework, design firms can create more realistic environments—for instance, incorporate strength of materials, safety factors, and malfunctioning of components under stressed conditions—and plug existing algorithms into this framework to optimize also for design aspects such as energy usage, ease of manufacturing, or durability. The designer may even incorporate aesthetic constraints such as symmetry and aspect ratios that suit her design sense.

In this work we have only explored using a simple population-based policy gradient method [72] for learning. State-of-the-art model-free RL algorithms, such as TRPO [54] and PPO [55], work well when our agent is presented with a well-designed dense reward signal, while population-based methods offer computational advantages for sparse-reward problems [51, 62]. In our setting, as the body design is parameterized by a small set of learnable parameters and is only set once at the beginning of a rollout, the problem of learning the body along with the policy becomes more sparse. In principle, we could allow an agent to augment its body *during* a rollout to obtain a dense reward signal, but we find this impractical for realistic problems. Future work may look at separating the learning from dense rewards and sparse rewards into an inner loop and outer loop, and also examine differences in performance and behaviors in structures learned with various different RL algorithms.

Separation of policy learning and body design into inner loop and outer loop will also enable the incorporation of evolution-based approaches to tackle the vast search space of morphology design, while utilizing efficient RL-based methods for policy learning. The limitation of the current approach is that our RL algorithm can learn to optimize only existing design properties of an agent's body, rather than learn truly novel morphology in the spirit of Karl Sims' "Evolving virtual creatures" [60].

Nevertheless, our approach of optimizing the specifications of an existing design might be practical for many applications. While a powerful evolutionary algorithm that can also evolve novel morphology might come up with robot morphology that easily outperforms the best bipedal walkers in this work, the resulting designs might not be as useful to a game designer who is tasked to work explicitly with bipedal walkers that fit within the game's narrative (although it is debatable whether a game can be more entertaining and interesting if the designer is allowed to explore the space beyond given specifications). Due to the vast search space of all possible morphology, a search algorithm can easily come up with unrealistic or unusable designs that exploit its simulation environment, as discussed in detail in [38], which may be why subsequent morphology evolution approaches constrain the search space of the agent's morphology—for example, to the space of soft-body voxels [17] or to a set of possible pipe frame connection settings [33]. We note that unrealistic designs may also result in our approach, if we do not constrain the learned dimensions to be within $\pm 75\%$ of their original values. For some interesting examples of what REINFORCE discovers without any constraints, we invite the reader to view the *Bloopers* section of https://designrl.github.io/.

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023

Just as REINFORCE [72] can also be applied to the discrete search problem of neural network architecture designs [74], similar RL-based approaches could be used for novel morphology design—not simply for improving an existing design as in this work. We believe the ability to learn useful morphology is an important area for the advancement of AI. Although morphology learning originally initiated from the field of evolutionary computation, we hope this work will engage the RL community to investigate the concept further and encourage idea exchange across communities.

Acknowledgments

We would like to thank the three reviewers from *Artificial Life* journal, as well as Luke Metz, Douglas Eck, Janelle Shane, Julian Togelius, Jeff Clune, and Kenneth Stanley, for their thoughtful feedback and conversation. All experiments were performed on CPU machines provided by Google Cloud Platform.

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023

References

1. Agrawal, S., Shen, S., & van de Panne, M. (2014). Diverse motions and character shapes for simulated skills. *IEEE Transactions on Visualization and Computer Graphics*, *20*(10), 1345–1355.
2. Akimoto, Y., Nagata, Y., Ono, I., & Kobayashi, S. (2012). Theoretical foundation for CMA-ES from information geometry perspective. *Algorithmica*, *64*(4), 698–716.
3. Anderson, M. L. (2003). Embodied cognition: A field guide. *Artificial Intelligence*, *149*(1), 91–130.
4. Auerbach, J., Aydin, D., Maesani, A., Kornatowski, P., Cieslewski, T., Heitz, G., Fernando, P., Loshchilov, I., Daler, L., & Floreano, D. (2014). Robogen: Robot generation through artificial evolution. In *Artificial Life Conference Proceedings 14* (pp. 136–137). Cambridge, MA: MIT Press.
5. Auerbach, J. E., & Bongard, J. C. (2010). Dynamic resolution in the co-evolution of morphology and control. In H. Fellermann, M. Dörr, M. Hanczyc, L. Laursen, S. Maurer, D. Merkle, P.-A. Monnard, K. Støy, & S. Rasmussen (Eds.), *Artificial Life XII: Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems* (pp. 451–458). Cambridge, MA: MIT Press.
6. Auerbach, J. E., & Bongard, J. C. (2010). Evolving CPPNs to grow three-dimensional physical structures. In J. Branke et al. (Eds.), *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation* (pp. 627–634). New York: ACM.
7. Auerbach, J. E., & Bongard, J. C. (2011). Evolving complete robots with CPPN-NEAT: The utility of recurrent connections. In M. Keijzer et al. (Eds.), *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (pp. 1475–1482). New York: ACM.
8. Auerbach, J. E., & Bongard, J. C. (2012). On the relationship between environmental and morphological complexity in evolved robots. In T. Soule et al. (Eds.), *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation* (pp. 521–528). New York: ACM.
9. Auerbach, J. E., & Bongard, J. C. (2014). Environmental influence on the evolution of morphological complexity in machines. *PLoS Computational Biology*, *10*(1), e1003399.
10. Beal, D., Hover, F., Triantafyllou, M., Liao, J., & Lauder, G. (2006). Passive propulsion in vortex wakes. *Journal of Fluid Mechanics*, *549*, 385–402.
11. Bongard, J. (2011). Morphological change in machines accelerates the evolution of robust behavior. *Proceedings of the National Academy of Sciences of the U.S.A.*, *108*(4), 1234–1239.
12. Bresadola, M. (1998). Medicine and science in the life of Luigi Galvani (1737–1798). *Brain Research Bulletin*, *46*(5), 367–380.
13. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *Preprint arXiv:1606.01540*, June.
14. Carter, S., Ha, D., Johnson, I., & Olah, C. (2016). Experiments in handwriting with a neural network. *Distill*.
15. Carter, S., & Nielsen, M. (2017). Using artificial intelligence to augment human intelligence. *Distill*.
16. Catto, E. (2011). Box2D: A 2D physics engine for games. GitHub.

17. Cheney, N., MacCurdy, R., Clune, J., & Lipson, H. (2013). Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation* (pp. 167–174). New York: ACM.

18. Collins, S., Ruina, A., Tedrake, R., & Wisse, M. (2005). Efficient bipedal robots based on passive-dynamic walkers. *Science*, *307*(5712), 1082–1085.

19. Collins, S. H., Wisse, M., & Ruina, A. (2001). A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research*, *20*(7), 607–615.

20. Corucci, F., Cheney, N., Giorgio-Serchi, F., Bongard, J., & Laschi, C. (2018). Evolving soft locomotion in aquatic and terrestrial environments: Effects of material properties and environmental transitions. *Soft Robotics*, *5*(4), 475–495.

21. Coumans, E. (2017). PyBullet Physics Environment. GitHub.

22. Deslandes, A., Moraes, H., Ferreira, C., Veiga, H., Silveira, H., Mouta, R., Pompeu, F. A., Coutinho, E. S. F., & Laks, J. (2009). Exercise and mental health: Many reasons to move. *Neuropsychobiology*, *59*(4), 191–198.

23. Gauci, J., & Stanley, K. O. (2010). Autonomous evolution of topographic regularities in artificial neural networks. *Neural Computation*, *22*(7), 1860–1898.

24. Geijtenbeek, T., Van De Panne, M., & Van Der Stappen, A. F. (2013). Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)*, *32*(6), 206.

25. Gover, M. R. (1996). The embodied mind: Cognitive science and human experience (book). *Mind, Culture, and Activity*, *3*(4), 295–299.

26. Griffis, D. (2018). RL A3C Pytorch Experiments. GitHub.

27. Guzdial, M., Liao, N., & Riedl, M. (2018). Co-creative level design via machine learning. *arXiv preprint arXiv:1809.09420*.

28. Ha, D. (2017). Evolving stable strategies. blog.otoro.net

29. Ha, S., Coros, S., Alspach, A., Kim, J., & Yamane, K. (2017). Joint optimization of robot design and motion parameters using the implicit function theorem. In N. Amato, S. Srinivasa, N. Ayanian, & S. Kuindersma (Eds.), *Robotics: Science and systems*. Cambridge, MA: MIT Press.

30. Ha, S., Coros, S., Alspach, A., Kim, J., & Yamane, K. (2018). Computational co-optimization of design parameters and motion trajectories for robotic systems. *The International Journal of Robotics Research*, 0278364918771172.

31. Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, *9*(2), 159–195.

32. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

33. Jansen, T. (2008). Strandbeests. *Architectural Design*, *78*(4), 22–27.

34. Klimov, O. (2010). BipedalWalker-v2. GitHub.

35. Klimov, O. (2016). BipedalWalkerHardcore-v2. GitHub.

36. Klimov, O., & Schulman, J. (2017). Roboschool. https://openai.com.

37. Leger, C. (1999). *Automated synthesis and optimization of robot configurations: An evolutionary approach*. Ph.D. Thesis, Carnegie Mellon University.

38. Lehman, J., Clune, J., Misevic, D., Adami, C., Beaulieu, J., Bentley, P. J., Bernard, S., Belson, G., Bryson, D. M., Cheney, N., et al. (2018). The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *arXiv preprint arXiv:1803.03453*.

39. Lipson, H., & Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, *406*(6799), 974.

40. Mahon, B. Z., & Caramazza, A. (2008). A critical look at the embodied cognition hypothesis and a new proposal for grounding conceptual content. *Journal of Physiology—Paris*, *102*(1–3), 59–70.

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023

41. McGeer, T. (1990). Passive walking with knees. In A. Koivo (Ed.), *Proceedings, IEEE International Conference on Robotics and Automation* (pp. 1640–1645). New York: IEEE.

42. Millington, I., & Funge, J. (2009). *Artificial intelligence for games*. Boca Raton, FL: CRC Press.

43. Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In M. F. Balcan & K. Q. Weinberger (Eds.), *International Conference on Machine Learning* (pp. 1928–1937). New York: PMLR.

44. Moore, J., Clark, A., & McKinley, P. (2014). Evolutionary robotics on the Web with Webgl and Javascript. *arXiv preprint arXiv:1406.3337*.

45. Ostergaard, E. H., & Lund, H. H. (2003). Evolving control for modular robotic units. In J. Kim (Ed.), *Proceedings, 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation, 2003*, *Vol. 2* (pp. 886–892). New York: IEEE.

46. Paul, C. (2004). Morphology and computation. In S. Schaal (Ed.), *Proceedings of the International Conference on the Simulation of Adaptive Behaviour* (pp. 33–38). Cambridge, MA: MIT Press.

47. Pfeifer, R., & Bongard, J. (2006). *How the body shapes the way we think: A new view of intelligence*. Cambridge, MA: MIT Press.

48. Prokopenko, M., Gerasimov, V., & Tanev, I. (2006). Evolving spatiotemporal coordination in a modular robotic system. In S. Nolfi, G. Baldassarre, R. Calabretta, J. C. T. Hallam, & D. Marocco (Eds.), *International Conference on Simulation of Adaptive Behavior* (pp. 558–569). Berlin: Springer.

49. Raglin, J. S. (1990). Exercise and mental health. *Sports Medicine*, *9*(6), 323–329.

50. Rechenberg, I. (1978). Evolutionsstrategien. In B. Schneider & U. Ranft (Eds.), *Simulationsmethoden in der Medizin und Biologie* (pp. 83–114). Dordrecht: Klewer Academic Publishing.

51. Salimans, T., Ho, J., Chen, X., Sidor, S., & Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *Preprint arXiv:1703.03864*.

52. Schaff, C., Yunis, D., Chakrabarti, A., & Walter, M. R. (2018). Jointly learning to construct and control agents using deep reinforcement learning. *arXiv preprint arXiv:1801.01432*.

53. Schaff, C., Yunis, D., Chakrabarti, A., & Walter, M. R. (2018). Jointly learning to construct and control agents using deep reinforcement learning. In *ICLR 2018 Workshop Version (Non-Archival)*.

54. Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In F. Bach & D. Blei (Eds.), *International Conference on Machine Learning* (pp. 1889–1897). New York: PMLR.

55. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

56. Schwefel, H.-P. (1981). *Numerical optimization of computer models*. New York: Wiley.

57. Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., & Schmidhuber, J. (2010). Parameter-exploring policy gradients. *Neural Networks*, *23*(4), 551–559.

58. Shapiro, L. (2010). *Embodied cognition*. London: Routledge.

59. Sims, K. (1994). Evolving 3D morphology and behavior by competition. *Artificial Life*, *1*(4), 353–372.

60. Sims, K. (1994). Evolving virtual creatures. In S. G. Mair (Ed.), *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (pp. 15–22). New York: ACM.

61. Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, *10*(2), 99–127.

62. Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., & Clune, J. (2017). Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *Preprint arXiv:1712.06567*.

63. Summerville, A., Snodgrass, S., Guzdial, M., Holmgard, C., Hoover, A. K., Isaksen, A., Nealen, A., & Togelius, J. (2018). Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games*, *10*(3), 257–270.

64. Szerlip, P., & Stanley, K. (2013). Indirectly encoded sodarace for artificial life. In P. Lio, O. Miglino, G. Nicosia, S. Nolfi, & M. Pavone, (Eds.), *The 2013 Conference on Artificial Life: A Hybrid of the European*

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023

Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE) (pp. 218–225). Cambridge, MA: MIT Press.

65. Szerlip, P. A., & Stanley, K. O. (2014). Steps toward a modular library for turning any evolutionary domain into an online interactive platform. In H. Sayama, J. Rieffel, S. Risi, R. Doursat, & H. Lipson (Eds.), *Artificial life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems* (pp. 900–907). Cambridge, MA: MIT Press.
66. Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In L. Parker (Ed.), *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 5026–5033). New York: IEEE.
67. Togelius, J., & Schmidhuber, J. (2008). An experiment in automatic game design. In P. Hingston & L. Barone (Eds.), *IEEE Symposium on Computational Intelligence and Games, 2008. CIG'08* (pp. 111–118). New York: IEEE.
68. Tricoli, V., Lamas, L., Carnevale, R., & Ugrinowitsch, C. (2005). Short-term effects on lower-body functional power development: Weightlifting vs. vertical jump training programs. *The Journal of Strength & Conditioning Research*, *19*(2), 433–437.
69. Volz, V., Schrum, J., Liu, J., Lucas, S. M., Smith, A. & Risi, S. (2018). Evolving Mario levels in the latent space of a deep convolutional generative adversarial network. *arXiv preprint arXiv:1805.00728*.
70. Weber, R. (2010). BoxCar2D. http://boxcar2d.com/about.html.
71. Wierstra, D., Schaul, T., Peters, J., & Schmidhuber, J. (2008). Natural evolution strategies. In *IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence)* (pp. 3381–3387). New York: IEEE.
72. Williams, R. J. (1992).Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3–4), 229–256.
73. Wilson, M. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review*, *9*(4), 625–636.
74. Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.
75. Zykov, V., Mytilinaios, E., Desnoyer, M., & Lipson, H. (2007). Evolved and designed self-reproducing modular robotics. *IEEE Transactions on Robotics*, *23*(2), 308–319.

Downloaded from http://direct.mit.edu/artl/article-pdf/25/4/352/1896269/artl_a_00301.pdf by guest on 08 September 2023