

# Reinforcement Learning for Improving Agent Design

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

## 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 a minor alteration to the OpenAI Gym [6] 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. Videos of results at: <https://designrl.github.io/>.

## 1 Introduction

Embodied cognition [2, 30, 42] is the theory that an organism’s cognitive abilities is 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 [17, 54]. Evolution plays a vital role in shaping an organism’s body to adapt to its environment; the brain and its ability to learn is only one of many body components that is co-evolved together [35]. We can observe embodiment in nature by observing that many organisms exhibit complex motor skills, such as the ability to jump [5] or swim [3] even after brain death.

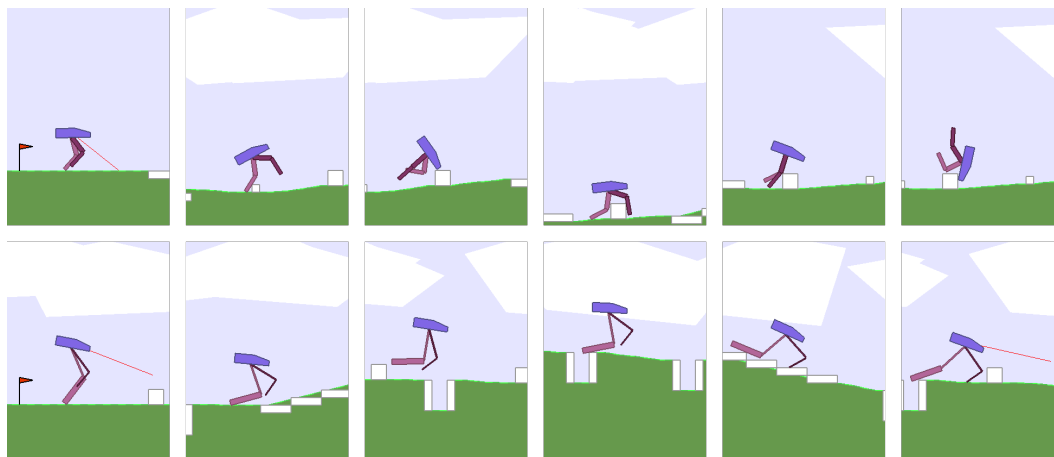


Figure 1: Learning to navigate over randomly generated terrain in BipedalWalkerHardcore-v2 environment [6] (top). Agent learns a better body design while jointly learning to navigate (bottom).

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. For instance, professional athletes spend their lives body training while also improving specific mental skills required to master a particular sport [50]. In everyday life, regular exercise not only strengthens the body but also improves mental conditions [15, 37]. 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 to investigate embodied cognition within the reinforcement learning (RL) framework. Most baseline tasks [27, 48] 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. We demonstrate that an agent can learn a better structure of its body that is not only 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. Our approach may help uncover design principles useful for assisted-design.

## 2 Related Work

There is a broad literature in evolutionary computation, artificial life and robotics devoted to studying, and modelling embodied cognition [35]. In 1994, Karl Sims demonstrated that artificial evolution can produce novel morphologies that resemble organisms observed in nature [43, 44]. Subsequent works further investigated morphology evolution [4, 28], modular robotics [33, 36, 56], and evolving soft robots [10, 13] using indirect encoding [16, 45]. Theo Jansen [24] used evolutionary computation to design physical *Strandbeests* creatures that can walk on their own consuming only wind energy. Literature in the area of passive dynamics study robot designs that rely on natural swings of motion of body components instead of deploying and controlling motors at each joint [11, 12, 31, 34]. Recent work in robotics investigates simultaneously optimizing body design and control of a legged robot [21] using constraint-based modelling, which is related to our RL-based approach. 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 the game playing agent [19, 47, 49, 51].

## 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 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  $w$ .

<pre>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</pre>	<pre>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</pre>
---	---

Figure 2: OpenAI Gym [6] 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 augment its reward based on these parameters (right).

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 [53]. Appendix A.3 provides an overview of this algorithm.

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 4.3.

## 4 Experiments

### 4.1 Learning better legs for better gait: RoboschoolAnt-v1 and BipedalWalker-v2

In this work, we experiment on continuous control environments from Roboschool [27], based on the open source Bullet [14] physics engine, and the Box2D [9] section of the OpenAI Gym [6] set of environments. The RoboschoolAnt-v1<sup>1</sup> environment features a four-legged agent called the *Ant*. The body is supported by 4 legs, and each leg consists of 3 parts which are controlled by 2 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  $\Delta x$  and  $\Delta y$  distances from its joint connection. A size parameter also controls the radius of each leg part.

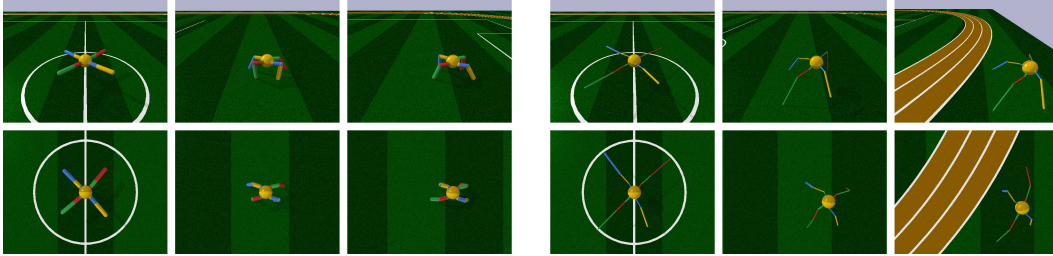


Figure 3: Agent learning a policy to navigate forward in RoboschoolAnt-v1 environment [27] (left). Agent develops longer, thinner legs while supporting the same body during training (right).

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.

	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%

Table 1: Learned agent body for RoboschoolAnt-v1 as % of original design specification.

Figure 3 illustrates the learned agent design compared to 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  $3447 \pm 251$  over 100 trials, compared to  $5789 \pm 479$  for an agent that learned a better body design.

The Bipedal Walker series of environments is based on the Box2D [9] physics engine. Guided by LIDAR sensors, the agent is required to navigate across an environment of randomly generated terrain

<sup>1</sup>A compatible version of this environment is also available in PyBullet [14] which was used for visualization.

within a time limit, without falling over. The agent’s payload – its head, is supported by 2 legs. The top and bottom parts of each leg is controlled by two motor joints. In the easier `BipedalWalker-v2` [25] 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.

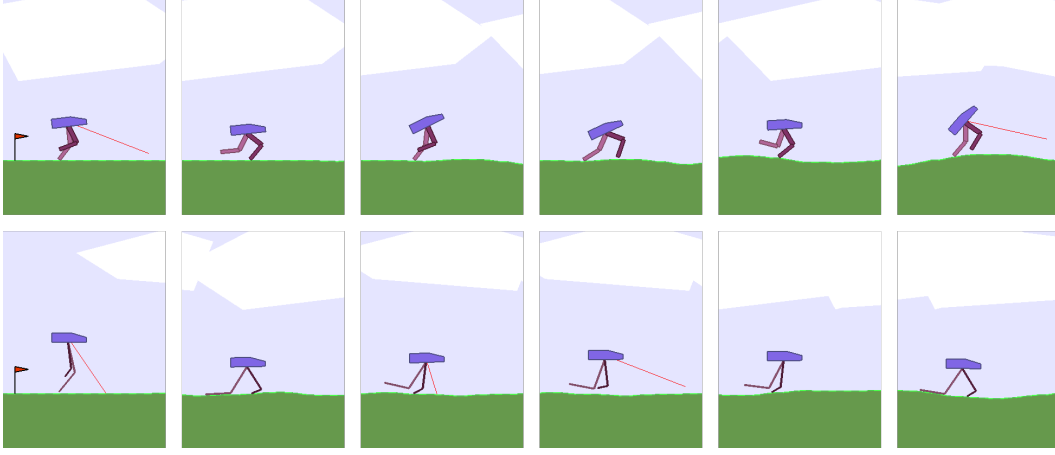


Figure 4: Agent learning a policy to navigate forward in `BipedalWalker-v2` environment [6] (top). Agent learns a body to allow it to bounce forward efficiently (bottom).

Keeping the head payload constant, and also keeping the density of materials and the configuration of the motor joints the same as the original environment, we only allow the lengths and widths for each of the 4 leg parts to be learnable, subject to the same range limit of  $\pm 75\%$  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.

#### 4.2 Joint learning of body design facilitates policy learning: `BipedalWalkerHardcore-v2`

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` [26] 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 1 illustrates the original and learnable versions of the environment.<sup>2</sup>

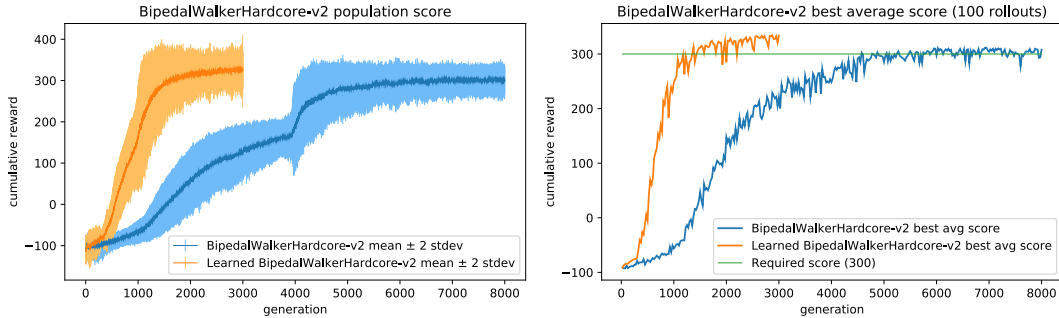


Figure 5: Population-based training curves for both versions of `BipedalWalkerHardcore-v2` (left). Plot of performance of best agent in the population over 100 random trials (right). Original version solved under 4600 generations (40 hours); learnable one solved under 1400 generations (12 hours).

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

<sup>2</sup>As of writing, two methods are reported to solve this task. Population-based training [20] (our baseline), solves this task in 40 hours on a 96-CPU machine, using a small feed forward policy network. A3C [32] adapted for continuous control [18] solves the task in 48 hours on a 72-CPU machine, but requires an LSTM [23] policy.

Its front legs, which are smaller and more manoeuvrable, also act as a sensor for dangerous obstacles ahead that 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  $335 \pm 37$  compared to the baseline score of  $313 \pm 53$ . The full results are summarized in Table 2.

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

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

Table 2: Summary of results for Bipedal Walker environments. Scaled Box2D dimensions reported.

### 4.3 Optimize for both the task and 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 also 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.

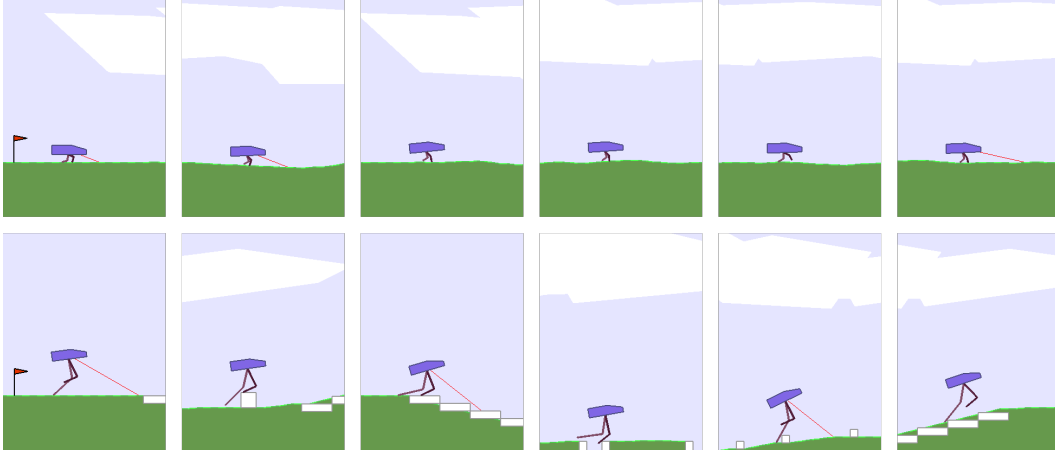


Figure 6: Agent rewarded for smaller legs for the task in BipedalWalker-v2 environment [6] (top). Agent learns the smallest pair of legs that still can solve BipedalWalkerHardcore-v2 (bottom).

This reward augmentation resulted in much a 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  $\sim 25\%$  of the original dimensions, with the exception of the heights of the top leg parts which settled at  $\sim 35\%$  of the initial design, while still achieving an average (unaugmented) score of  $323 \pm 68$ . 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 widths of each leg part close to the lower bound, and instead learn the shortest heights of each leg part required to navigate, achieving a score of  $312 \pm 69$ . Here, the leg area used is 27% of the original.

## 5 Discussion and Future Work

We have shown that allowing a simple population-based policy gradient method to learn not only the policy, but also a small set of parameters describing the environment, such as its body, offer many benefits. By allowing the agent’s body to adapt to its task within some constraints, it can learn policies that are not only better for its task, but also learn them more quickly.

The agent may discover design principles during this process of joint body and policy learning. In both `RoboschoolAnt` and `BipedalWalker` experiments, the agent has learned to break symmetry and learn relatively 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 by setting the size of its legs to the minimum allowable size. 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 [7, 8]. 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 [19, 47, 49, 51]. 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, malfunctioning of components under stressed conditions, and plug existing algorithms into this framework to optimize also for design aspects such as energy usage, easy-of-manufacturing, or durability. The designer may even incorporate aesthetic constraints such as symmetry and aspect ratios that suits her design sense.

In this work we have only explored using a simple population-based policy gradient method [53] for learning. State-of-the-art model-free RL algorithms, such as TRPO [39] and PPO [40] 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 [38, 46]. 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 unpractical 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 behaviours 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 limitations 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’s *Evolving Virtual Creatures* [44].

Nevertheless, our approach of optimizing the specifications of an existing design might be more practical for many applications. An evolutionary algorithm might come up with trivial designs and corresponding simple policies that outperform designs we actually want – for instance, a large ball that rolls forward will easily outperforming the best bipedal walkers, but this might not be useful to a game designer who simply wants to optimize the dimensions of an existing robot character for a video game. Due to the vast search space of morphology, a search algorithm can easily come up with a trivial, but unrealistic or unusable design that exploits its simulation environment [29], which may be why subsequent morphology-evolution approaches constrain the search space of the agent’s morphology, such as constraining to the space of soft-body voxels [10] or constraining to a set of possible pipe frame connection settings [24].

Just as REINFORCE [53] can also be applied to the discrete search problem of neural network architecture designs [55], similar RL-based approaches could be used for novel morphology design – not simply for improving an existing design like 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 Luke Metz and Douglas Eck for their thoughtful feedback and conversation.

## A Appendix

### A.1 Configuration

All agents were implemented using 3 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` has 24 inputs and 4 outputs all bounded between  $-1$  and  $+1$ , with 2 hidden layers of 40 units each.

### A.2 Training

Our population-based training experiments were conducted on 96-CPU core machines on Google Cloud Platform. Following the approach described in [20], 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. Baseline agents were trained for 8000 generations, while learnable environments used 3000 generations.

### A.3 Population-based Policy Gradient Method

In this section we provide an overview of the population-based policy gradient method described in Section 6 of William’s REINFORCE [53] paper for learning a parameter vector  $w$  in a reinforcement learning environment.<sup>3</sup> In this approach,  $w$  is sampled from a probability distribution  $\pi(w, \theta)$  parameterized by  $\theta$ . We define the expected cumulative reward  $R$  as:

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

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

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

In a population size of  $N$ , where we have solutions  $w^1, w^2, \dots, w^N$ , we can estimate this as:

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

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

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

and sample a new set of candidate solutions  $w$  from updating the pdf using learning rate  $\alpha$ . We follow the approach in REINFORCE where  $\pi$  is modelled as a factored multi-variate normal distribution. Williams derived closed-form formulas of the gradient  $\nabla_{\theta} \log \pi(w^i, \theta)$ . In this special case,  $\theta$  will be the set of mean  $\mu$  and standard deviation  $\sigma$  parameters. 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 the  $\nabla_{\theta} \log N(z^i, \theta)$  term in Equation 3, for each individual  $\mu$  and  $\sigma$  element of vector  $\theta$  on each solution  $i$  in the population:

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

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

---

<sup>3</sup>Subsequent population-based methods based on Sec. 6 of REINFORCE include PGPE [41], NES [52], and OpenAI-ES [38]. These population-based policy gradient methods are also closely related to CMA-ES [1, 22].

## References

- [1] Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi. Theoretical foundation for cma-es from information geometry perspective. *Algorithmica*, 64(4):698–716, 2012.
- [2] M. L. Anderson. Embodied cognition: A field guide. *Artificial intelligence*, 149(1):91–130, 2003.
- [3] D. Beal, F. Hover, M. Triantafyllou, J. Liao, and G. Lauder. Passive propulsion in vortex wakes. *Journal of Fluid Mechanics*, 549:385–402, 2006.
- [4] J. Bongard. Morphological change in machines accelerates the evolution of robust behavior. *Proceedings of the National Academy of Sciences*, 108(4):1234–1239, 2011.
- [5] M. Bresadola. Medicine and science in the life of luigi galvani (1737–1798). *Brain Research Bulletin*, 46(5):367–380, 1998.
- [6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *Preprint arXiv:1606.01540*, June 2016.
- [7] S. Carter, D. Ha, I. Johnson, and C. Olah. Experiments in handwriting with a neural network. *Distill*, 2016.
- [8] S. Carter and M. Nielsen. Using artificial intelligence to augment human intelligence. *Distill*, 2017. <https://distill.pub/2017/aia>.
- [9] E. Catto. Box2d: A 2d physics engine for games. <https://github.com/pybox2d/pybox2d>, 2011.
- [10] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson. 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*, pages 167–174. ACM, 2013.
- [11] S. Collins, A. Ruina, R. Tedrake, and M. Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.
- [12] S. H. Collins, M. Wisse, and A. Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research*, 20(7):607–615, 2001.
- [13] F. Corucci, N. Cheney, F. Giorgio-Serchi, J. Bongard, and C. Laschi. Evolving soft locomotion in aquatic and terrestrial environments: effects of material properties and environmental transitions. *Soft robotics*, 5(4):475–495, 2018.
- [14] E. Coumans. PyBullet Physics Environment. <https://github.com/bulletphysics/bullet3>, 2017.
- [15] A. Deslandes, H. Moraes, C. Ferreira, H. Veiga, H. Silveira, R. Mouta, F. A. Pompeu, E. S. F. Coutinho, and J. Laks. Exercise and mental health: many reasons to move. *Neuropsychobiology*, 59(4):191–198, 2009.
- [16] J. Gauci and K. O. Stanley. Autonomous evolution of topographic regularities in artificial neural networks. *Neural Computation*, 22(7):1860–1898, July 2010.
- [17] M. R. Gover. The embodied mind: Cognitive science and human experience (book). *Mind, Culture, and Activity*, 3(4):295–299, 1996.
- [18] D. Griffiths. RL A3C Pytorch Continuous. [https://github.com/dgriff777/a3c\\_continuous](https://github.com/dgriff777/a3c_continuous), 2018.
- [19] M. Guzdial, N. Liao, and M. Riedl. Co-creative level design via machine learning. *arXiv preprint arXiv:1809.09420*, 2018.
- [20] D. Ha. Evolving stable strategies. <http://blog.otoro.net/>, 2017.
- [21] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane. Computational co-optimization of design parameters and motion trajectories for robotic systems. *The International Journal of Robotics Research*, page 0278364918771172, 2018.
- [22] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, June 2001.
- [23] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [24] T. Jansen. Strandbeests. *Architectural Design*, 78(4):22–27, 2008.
- [25] O. Klimov. BipedalWalker-v2. <http://gym.openai.com/>, 2016.
- [26] O. Klimov. BipedalWalkerHardcore-v2. <http://gym.openai.com/>, 2016.
- [27] O. Klimov and J. Schulman. Roboschool. <https://blog.openai.com/roboschool/>, May 2017.
- [28] C. Leger et al. *Automated synthesis and optimization of robot configurations: an evolutionary approach*. Carnegie Mellon University USA, 1999.
- [29] J. Lehman, J. Clune, D. Misevic, C. Adami, J. Beaulieu, P. J. Bentley, S. Bernard, G. Belson, D. M. Bryson, N. Cheney, et al. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *arXiv preprint arXiv:1803.03453*, 2018.



- [30] B. Z. Mahon and A. Caramazza. 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, 2008.
- [31] T. McGeer. Passive walking with knees. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 1640–1645. IEEE, 1990.
- [32] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.
- [33] E. H. Ostergaard and H. H. Lund. Evolving control for modular robotic units. In *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium on*, volume 2, pages 886–892. IEEE, 2003.
- [34] C. Paul. Morphology and computation. In *Proceedings of the International Conference on the Simulation of Adaptive Behaviour Los Angeles, CA, USA*, pages 33–38, 2004.
- [35] R. Pfeifer and J. Bongard. *How the body shapes the way we think: a new view of intelligence*. MIT press, 2006.
- [36] M. Prokopenko, V. Gerasimov, and I. Tanev. Evolving spatiotemporal coordination in a modular robotic system. In *International Conference on Simulation of Adaptive Behavior*, pages 558–569. Springer, 2006.
- [37] J. S. Raglin. Exercise and mental health. *Sports Medicine*, 9(6):323–329, 1990.
- [38] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *Preprint arXiv:1703.03864*, 2017.
- [39] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.
- [40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [41] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.
- [42] L. Shapiro. *Embodied cognition*. Routledge, 2010.
- [43] K. Sims. Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372, 1994.
- [44] K. Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 1994.
- [45] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.
- [46] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *Preprint arXiv:1712.06567*, Dec. 2017.
- [47] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgard, A. K. Hoover, A. Isaksen, A. Nealen, and J. Togelius. Procedural content generation via machine learning (pcgml). *IEEE Transactions on Games*, 2018.
- [48] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.
- [49] J. Togelius and J. Schmidhuber. An experiment in automatic game design. In *Computational Intelligence and Games, 2008. CIG’08. IEEE Symposium On*, pages 111–118. IEEE, 2008.
- [50] V. Tricoli, L. Lamas, R. Carnevale, and C. Ugrinowitsch. 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, 2005.
- [51] V. Volz, J. Schrum, J. Liu, S. M. Lucas, A. Smith, and S. Risi. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. *arXiv preprint arXiv:1805.00728*, 2018.
- [52] D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber. Natural evolution strategies. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3381–3387. IEEE, 2008.
- [53] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.
- [54] M. Wilson. Six views of embodied cognition. *Psychonomic bulletin & review*, 9(4):625–636, 2002.
- [55] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.
- [56] V. Zykov, E. Mytilinaios, M. Desnoyer, and H. Lipson. Evolved and designed self-reproducing modular robotics. *IEEE Transactions on robotics*, 23(2):308–319, 2007.