JOINTLY LEARNING TO CONSTRUCT AND CONTROL AGENTS USING DEEP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

The physical design of a robot and the policy that controls its motion are inherently coupled. However, existing approaches largely ignore this coupling, instead choosing to alternate between separate design and control phases, which requires expert intuition throughout and risks convergence to suboptimal designs. In this work, we propose a method that jointly optimizes over the physical design of a robot and the corresponding control policy in a model-free fashion, without any need for expert supervision. Given an arbitrary robot morphology, our method maintains a distribution over the design parameters and uses reinforcement learning to train a neural network controller. Throughout training, we refine the robot distribution to maximize the expected reward. This results in an assignment to the robot parameters and neural network policy that are jointly optimal. We evaluate our approach in the context of legged locomotion, and demonstrate that it discovers novel robot designs and walking gaits for several different morphologies, achieving performance comparable to or better than that of hand-crafted designs.

1 INTRODUCTION

An agent’s ability to navigate through and interact with its environment depends not just on its skill at planning and controlling its motion, but also on its physical design. Different physical designs are inherently better suited to different tasks and environments. By making appropriate choices during fabrication, mechanical elements can be designed to improve robustness to non-idealities such as errors in perception, delays in actuation, etc., and indeed, make control problem an easier one to solve. At the same time, robots that take different forms may find completely different control strategies to be optimal to complete the same task. Therefore, the physical and computational design of an agent are inherently coupled, and must ideally be jointly optimized if the robot is to successfully complete a task in a particular environment.

Consider the development of a legged robot for locomotion. Variations in physical design will require changes to the joint torques in order to preserve a particular locomotion behavior (e.g., heavier feet require greater torque at the ankle), and will likely result in completely different walking gaits, even when the morphology is preserved. In fact, some changes to design may render locomotion impossible for the target operating environment (e.g., a robot with long feet may be unable to locomote up an incline). It is therefore beneficial to not simply consider the robot’s design or gait to be fixed, but to optimize both jointly for the target environment and task. Similar co-design can be beneficial in other settings—for example for the control policy and physical characteristics of digits in robotic grippers for grasping.

While a robot’s physical design and the corresponding control policy are inherently coupled, most existing methods ignore this coupling, instead choosing to alternate between separate design and control phases. Existing approaches that jointly reason over design and control (Digumarti et al., 2014; Ha et al., 2017; Spielberg et al., 2014) assume knowledge of an accurate model of the robot dynamics and require expert supervision (e.g., to provide a suitable initial design and guide the optimization process). However, these restrictive assumptions limits their applicability to a handful of specific settings, and often yield solutions heavily influenced by expert intuition.

In this work, we seek a general approach—one that can optimize a robot’s physical characteristics jointly with controllers of a desired complexity (Fig. 1), that can be applied to general tasks in some given environment, and that can explore the joint search space of physical design and computational
control in a purely data-driven way, without a model of the robot dynamics and independent of the biases of expert intuition. We develop this approach in the context of determining the physical parameters of an articulated agent—the lengths and thicknesses of each limb in a given morphology—through joint training with a neural network for control, with the objective of achieving locomotion. Our method maintains a distribution over these physical parameters, and simultaneously trains the parameters of this distribution with those of a neural network controller, using deep reinforcement learning. In this way, we pursue a design distribution and control policy that are jointly optimal for the given task and environment. Experimental results show that starting from random initializations, our approach is able to find novel designs and walking gaits that match or exceed the performance of manually designed agents. To do the best of our knowledge, our method is the first to successfully carry out such a joint optimization of design and control in a completely model-free manner.

2 RELATED WORK

Similar to our objective, attention has been paid recently to the problem of jointly optimizing the parameters that specify robot structure and a parameterization of its motion (e.g., gaits). Particularly relevant is the work of Ha et al. (2017), who relate design and motion parameters via a set of implicit functions that express robot dynamics, desired trajectories, and actuation limits. These functions encode a manifold that is then linearized to model the relationship between design and motion via the implicit function theorem. The method then solves for the desired parameters in a local fashion via constraint-based optimization. Similarly, Spielberg et al. (2014) describe an approach that jointly reasons over physical design and motion parameters for robots with articulated degrees of freedom (e.g., legged robots). They formulate the problem in the framework of trajectory optimization by incorporating parameters involved in the design of articulated robots, including dimensions, masses, mass distribution, and moments of inertia, together with the contact force and torque variables typically associated with trajectory optimization. They use weighted actuation cost as the objective, subject to a regularization penalty on parameters. Their method is guaranteed to find a feasible design so long as the problem is initialized within a small neighborhood of a feasible solution. Unlike our approach, their method requires that the user provide an estimate of the robot configuration at each time step, as well as an accurate analytic model of constraint dynamics (e.g., foot contact), which is computationally expensive to enforce. Meanwhile, Digumarti et al. (2014) similarly describe an evolutionary method that jointly reasons over design and motion parameters for legged robots, while also tuning the parameters of a robust controller that tracks these motions. Their method is limited to biologically inspired quadrupedal foothold patterns, and does not account for contact dynamics. In contrast, our approach applies to arbitrary morphologies and does not require that we model contact dynamics.

While the focus on simultaneous optimization over robot design and motion parameters is relatively new, there is a long history of research focused on the related problem of co-design of physical structure and control (Park & Asada, 1994; Pil & Asada, 1996), (Reyer & Papalambros, 2002),
et al. (2014) synthesize electromechanical robot designs in a compositional fashion based upon a
(2013) learn optimum kinematic linkages that are capable of reproducing these motions. Mehta
requirements of a particular task. Given a user demonstration of the desired behaviors, Coros et al.
Meanwhile, much work has focused on the problem of determining robot designs that meet the
methods result in the same gradient estimate in the event that the parameter distribution is Gaus-
formulate as a Gaussian mixture model over design parameters. Indeed, the two aforementioned
algorithms. These approaches are similar to the way in which we learn the robot design, which we
over. This results in gradient estimates that are far less noisy than is typical of policy gradient al-
suited to difficult reinforcement learning problems, Salimans et al. (2017) recently showed that they
perform similarly to state-of-the-art RL methods on difficult continuous control problems, including
number of methods interleave trajectory optimization and supervised learning with neural network
regression (Levine & Abbeel, 2014; Levine & Koltun, 2014; Mordatch & Todorov, 2014; Mordatch
et al., 2015). Unlike our method, the use of trajectory optimization makes these approaches reliant
upon knowledge of the model.

A great deal of attention of-late has focused on the problem of learning complex control policies
directly from low-level sensory input, without any knowledge of the system dynamics. Methods that
have proven particularly effective combine neural networks that learn representations of the high-
dimensional raw sensor input with deep reinforcement learning (Riedmiller, 2005; Mnih et al., 2013;
2015; Schulman et al., 2015a). While much of the work in this area focuses on low-dimensional,
discrete action spaces, several methods have been recently proposed that learn continuous control
policies through deep reinforcement learning. These techniques have been applied to control simple,
simulated robots (Wawrzyński, 2009; Wawrzyński & Tanwani, 2013; Watter et al., 2015; Lillicrap
et al., 2015), perform robot manipulation Levine et al. (2016a;b); Gu et al. (2017), and control legged
robots (Schulman et al., 2015b; Peng et al., 2016).

Black-box optimization (Schmidhuber et al., 2007; Risi & Togelius, 2017) is an alternative to using
reinforcement learning to training the control policy. These approaches have the advantage that they
do not involve backpropagating gradients, are insensitive to reward sparsity, and can handle long
time horizons. While black-box optimization strategies have traditionally been thought of as ill-
suited to difficult reinforcement learning problems, Salimans et al. (2017) recently showed that they
perform similarly to state-of-the-art RL methods on difficult continuous control problems, including
those that involve locomotion. Closely related is the policy-gradient method Sehnke et al. (2010),
who define the policy as a distribution over the parameters of a controller, which they then sample
over. This results in gradient estimates that are far less noisy than is typical of policy gradient al-
gorithms. These approaches are similar to the way in which we learn the robot design, which we
formulate as a Gaussian mixture model over design parameters. Indeed, the two aforementioned
methods result in the same gradient estimate in the event that the parameter distribution is Gauss-
ian Salimans et al. (2017).

Meanwhile, much work has focused on the problem of determining robot designs that meet the
requirements of a particular task. Given a user demonstration of the desired behaviors, Coros et al.
(2013) learn optimum kinematic linkages that are capable of reproducing these motions. Mehta
et al. (2014) synthesize electromechanical robot designs in a compositional fashion based upon a
complete user-specified structural specification of the robot. Mehta et al. (2016) build upon this work, allowing the user to specify functional objectives via structured English, which is parsed to a formal specification using linear temporal logic. Censi (2017) describes a theory for co-design that includes the ability to select discrete robot parts according to functional constraints, but do not reason over geometry or motion.

Related to our approach is recent work that jointly optimizes sensor design and inference algorithms for perception systems. Chakrabarti (2016) considers the problem of jointly learning a camera sensor’s multiplexing pattern along with reconstruction methods for imaging tasks. They model inference as a neural network together and use stochastic gradient descent to backpropagate the loss to a neural layer representation of the multiplexing pattern. Related, Schaff et al. (2017) jointly learn design and inference for beacon-based localization. They encode beacon allocation (spatially and across transmission channels) as a differential neural layer that interfaces with a neural network for inference. Joint optimization then follows from standard techniques for training neural networks.

3 APPROACH

In this section, we begin by describing the standard reinforcement learning framework for training agent policies, and then describe our approach of extending this framework to also learn the physical design of the agent.

3.1 REINFORCEMENT LEARNING BACKGROUND

In the standard reinforcement learning setting, an agent interacts with its environment, usually a Markov Decision Process, over a number of discrete timesteps. At each time step \( t \), the agent receives a state \( s_t \in \mathcal{S} \) and takes action \( a_t \in \mathcal{A} \) according to a policy \( \pi : \mathcal{S} \rightarrow \mathcal{A} \). Then, the agent receives a scalar reward \( r_t \) and the next state \( s_{t+1} \) from the environment. This process continues until a terminal state is reached. The goal of reinforcement learning is to then find a policy \( \pi^* \) that maximizes the expected return \( \mathbb{E}[R_t] \), where
\[
R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}
\]
and \( \gamma \in (0, 1) \) is a discount factor.

Policy gradient methods are a class of algorithms often used to optimize reinforcement learning problems, due to their ability to optimize cumulative reward and the ease with which the can be used with neural networks and other nonlinear function approximators. Consequently, they are commonly used for reinforcement learning problems that involve complex, continuous action spaces. Policy gradient methods directly parameterize a stochastic policy \( \pi_\theta(a_t|s_t) \) and perform stochastic gradient ascent on the expected return. “Vanilla” policy gradient methods compute an estimate of the gradient \( \nabla_\theta \mathbb{E}[R_t] \) using a sample-based mean computed over \( \nabla_\theta \log \pi_\theta(a_t|s_t) R_t \) (Williams, 1992), which yields an unbiased gradient estimate (Sutton et al., 2000). While the variance in the resulting estimate decreases with the number of samples, sampling is computationally expensive. Another way to reduce variance, while maintaining an unbiased estimate, is to estimate the gradient by comparing the reward to a “baseline” reward \( b(s_t) \).

While effective, these methods can be unstable, especially in the context of deep reinforcement learning. Small changes to the policy parameters may cause large changes in the distribution of visited states. Several methods have been proposed to mitigate these effects. Among them, Schulman et al. (2015a) introduce Trust Region Policy Optimization (TRPO), which imposes a constraint on the KL-divergence between policies before and after an update. Recently, Schulman et al. (2017) proposed proximal policy optimization (PPO), a first-order class of methods similar to TRPO. PPO alternates between sampling data through interaction with the environment and optimizing the objective
\[
\hat{E}_t \left[ \min(\pi_\theta(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)) \right],
\]
where
\[
r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\hat{\pi}_{t-1}(a_t|s_t)}
\]
and \( \hat{E}_t \) represents an empirical average over a finite sample set. This objective seeks to maximize the expected return while encouraging a small step in policy space. The clipping within the objective removes any incentive for moving \( r_t(\theta) \) outside the interval \( [1-\epsilon, 1+\epsilon] \). The net effect of this objective is a simple, robust policy gradient algorithm that attains or matches state-of-the-art results on a wide array of tasks (Schulman et al., 2017).
iterations are repeated until convergence. We then use this data to shift the robot policy sample methods that have proven effective for complex learning problems (Salimans et al., 2017). We parameter space, similar to Sehnke et al. (2010). This is equivalent to black-box optimization. Without knowledge of the model, we optimize the robot parameters we freeze the policy parameters, and proceed to optimize the parameters of the robot distribution. Fixed number Optimization. However, instead of collecting data from a single robot, we sample a robot robot distribution, Our method (Algorithm 3.2) employs an alternating minimization of the parameters of the policy and where be a stochastic policy parameterized by Formally, let (2) Joint Optimization of Design and Control In our approach, we extend the standard reinforcement learning formulation by considering a space of possible robot designs . Specifically, we assume that for every design in our approach is to find the jointly optimal design and policy that maximize expected reward. In the context of joint design and control for robotic agents, each design represents the problem of controlling a specific robot in order to accomplish a given task. Due to the constraint that all designs shares a common action space, we assume that there is a single morphology that is shared by all possible robot designs. This optimization is effectively a joint search over the spaces of all possible designs and policies. Without any more information about the transition dynamics, the best approach would be to train separate policies for every possible design, and pick the best performing design-policy pair. However, it is often reasonable to assume that robots that are “nearby” in parameter space exhibit similar dynamics. In other words, there is some notion of local smoothness to the transition dynamics as a function of . Therefore, if the two robots are similar enough, a good policy for one will perform reasonably well for the other. This suggests that features learned while training the policy associated with one robot will be useful for other robots as well, so long as the diversity in robot parameters is consistent with the generalizability of the control policy. With this in mind, we propose to train a single, shared policy to maximize the expected return over a distribution of robots. The use of a single controller has two primary advantages. First, it permits efficient training across all designs. Second, it allows us to quickly evaluate the potential of unseen robots by using the policy as a baseline for performance.

Formally, let be a distribution over robots parameterized by . Additionally, let be a stochastic policy parameterized by . Note that in addition to the current state , the policy also depends on the robot it is acting on. Our goal is to solve the following optimization problem:

where is the expectation over robots and trajectories those robots induce.

Our method (Algorithm 3.2) employs an alternating minimization of the parameters of the policy and robot distribution, and respectively. We optimize the policy parameters using Proximal Policy Optimization. However, instead of collecting data from a single robot, we sample a robot after a fixed number of timesteps according to the robot distribution . After iterations of PPO, we freeze the policy parameters, and proceed to optimize the parameters of the robot distribution. Without knowledge of the model, we optimize the robot parameters via policy gradient over parameter space, similar to Sehnke et al. (2010). This is equivalent to black-box optimization methods that have proven effective for complex learning problems (Salimans et al., 2017). We sample different robots and compute their returns for a single episode acting under policy . We then use this data to shift the robot policy in a direction maximizes expected return. These iterations are repeated until convergence.

Algorithm 1 Joint Optimization

```plaintext
Initialize \( \pi(\theta \mid s, r), p(\phi) \)
while True do
    for \( i \in [1..n_p] \) do
        Sample \( r_1, ..., r_n \) from \( p(\phi) \)
        Collect rollouts with each robot
        Update \( \theta \) to maximize PPO’s surrogate objective.
    end for
    for \( i \in [1..n_r] \) do
        Sample \( r_1, ..., r_m \) from \( p(\phi) \)
        Compute returns \( R_{1}, ..., R_{m} \).
        Update \( \phi \) to maximize PPO’s surrogate objective.
    end for
end while
```
Under review as a conference paper at ICLR 2018

Figure 2: From left to right: the default Hopper, Walker2d, and Ant robots. The Hopper and Walker2d are constrained to walk in a line, while the Ant can walk anywhere in the plane.

4 RESULTS

We validate our approach with three commonly used robots provided in OpenAI’s Roboschool (Schulman et al., 2017). Environments in Roboschool are built on top of Bullet Physics, a popular open-source physics engine. In a series of locomotion experiments, we show that our approach is not only able to discover novel robot designs and gaits, but also outperforms two out of three of the given robots trained on the same task, and yields performance comparable to the third.

4.1 EXPERIMENT DETAILS

We evaluate our approach on three legged locomotion tasks within Roboschool: RoboschoolHopper, RoboschoolWalker2d, and RoboschoolAnt (note that we subsequently drop the Roboschool prefix for brevity). Figure 2 depicts the default Roboschool design for each robot. These environments describe a locomotion task that has the robots moving along the positive x-axis (to the right in the figures show) towards a distant goal. The reward function defined in these environments is a weighted sum of: rewards for forward progress and staying upright, and penalties for applying torques and for joints that reach their rotational limits. The episode ends when the robot falls over or reaches a maximum number of timesteps.

We learn robot designs using the morphologies specified by the standard Roboschool designs. For each morphology, we parameterize the robot in terms of the length and radius (e.g., mass) of each link (or just the radius in the case of the sphere for the ant body). We impose symmetry, and share parameters across each leg for the Walker2d and Ant designs. This parameterization permits a wide variety of robots of different shapes and sizes, some of which are better suited to the locomotion objective than others, and some designs that do not permit a control policy that results in locomotion.

We model the control policy $\pi(a_t | s_t, E; \theta_p)$ as a convolutional neural network consisting of three fully-connected layers with 64 units and tanh activation functions. A final layer maps the output to the robot action space. With the exception of the last robot-specific layer, the architecture is the same for all experiments. Meanwhile, we represent the distribution over the parameterized robot design $p(\cdot; \theta_r)$ using a Gaussian mixture model with four mixture components. Empirically, we find that having a multimodal distribution has two primary advantages. First, it promotes exploration in the space of robot designs by allowing the framework to simultaneously maintain several distinct hypotheses. Second, a mixture model encourages the policy network to generalize across a wider array of robots. We initialize the means of each component randomly within a wide range, and initialize the variances in order to span the range of each parameter.

We train our method with eight threads of Proximal Policy Optimization (Schulman et al., 2017) for a total of 300M environment timesteps across all threads. We use the publicly available implementation of PPO provided in OpenAI Baselines (Hesse et al., 2017). We tune all hyperparameters on the Hopper, and subsequently fix them across all experiments.
4.2 EXPERIMENTS

We evaluate the performance of our approach to joint optimization over robot design and control on the Hopper, Walker2d, and Ant robots, and compare against a policy trained according to the standard Roboschool design. We evaluate the consistency and robustness of our approach relative to the initial robot parameter distribution using several different random seeds. For each robot morphology, we find that our method learns robot designs and the corresponding control policy that are either comparable (Walker2d) or exceed the performance (Hopper and Ant) achievable using the default designs (Fig. 3). Further, we see that our method achieves these levels of performance by discovering unique robot designs (Fig. 1) together with novel walking gaits (Fig. 4) that can not be achieved with the default designs. We stress the fact that our method learned these designs and control policies from a random initialization, without access to a dynamics model or any expert supervision.

Our method learns a joint design and controller for the Hopper that outperforms the baseline by as much as 50%. Our learned robot exploits the small size of the foot to achieve faster, more precise walking gaits. Meanwhile, the longer torso of the learned robot improves stability, allowing it to maintain balance while locomoting at a faster pace. We found this behavior to be consistent across several different Hopper experiments, with the method converging to designs with a small foot and long, thin torso.

Our method learns a physical design of the Ant that is significantly different from the default design (Fig. 2). Consequently, the learned Ant drastically outperforms the baseline, improving reward by up to 116%. Our method learns a design that has a small, lightweight body and extremely long legs. The long legs enable the ant to apply large torque at contact, allowing it to move at a fast pace.

Our framework learned different design-control pairs for the Walker that perform similarly to the default design. Across several different experiments, we see two distinct, successful designs and walking gaits. Interestingly, neither agent walks primarily on its feet. The first design has small, thick legs and long feet. The controller is then able to exploit the thick middle leg link, which protrudes slightly past the foot, to push off the ground. The long foot then provides additional balance. The second design is similar in geometry to the baseline Walker2d, but moves in a very different way. By lowering the knee joint and lengthening the foot, the Walker2d is able to efficiently balance on its knees and toes. This low stance allows the Walker2d to fully extend its leg backwards, creating a long, powerful stride, similar to that of a sprinter using starting blocks.

5 CONCLUSION

We proposed what is, to the best of our knowledge, the first model-free algorithm that jointly optimizes over the physical design of a robot and the corresponding control policy, without any need for expert supervision. Given an arbitrary morphology, our robot maintains a distribution over the
Figure 4: Here, we compare locomotion of default and learned robots, visualizing both their physical design and corresponding learned gaits. We pick the best results for Hopper and Ant, and two of the best results for Walker2d (due to diversity in gaits). Note that for each robot type, we show a blend of a fixed number of frames in the same time interval, allowing direct comparison between the speed with which different designs are able to locomote.

robot design parameters and learns these parameters together with a neural network controller using policy gradient-based reinforcement learning. This results in an assignment to the policy over robot parameters and the control policy that are jointly optimal. We evaluated our approach on a series of different legged robot morphologies, demonstrating that it results in novel robot designs and walking gaits, achieving performance that either matches or exceeds that of manually defined designs.

This work provides several opportunities for future work. Among them is to explore the ability to learn legged robot designs and controllers that are able to locomote despite uneven terrain, the presence of obstacles, changes in slope, and variations in friction. Further, we are currently working on relaxing the assumption that the morphology is pre-defined, permitting our algorithm to search both over different morphologies as well as their particular parameterization. It is worth stressing our joint approach to design and control is not specific to legged robots. Thus, we are also investigating applications to different types of agents (e.g., robot end-effectors). This will require that we revisit the use of a Gaussian mixture model as the representation of the robot distribution for agents with higher degrees-of-freedom and, in turn, more complex design distributions.
REFERENCES


