Deep Reinforcement Learning for Conservation Decisions

A Preprint

Marcus Lapeyrolerie
Department of Environmental Science, Policy, and Management
University of California, Berkeley
Berkeley, California

Melissa Chapman
Department of Environmental Science, Policy, and Management
University of California, Berkeley
Berkeley, California

Kari Norman
Department of Environmental Science, Policy, and Management
University of California, Berkeley
Berkeley, California

Carl Boettiger
Department of Environmental Science, Policy, and Management
University of California, Berkeley
Berkeley, California
cboettig@berkeley.edu

June 8, 2021

Abstract

Enter the text of your abstract here.

Keywords blah · blee · bloo · these are optional and can be removed

1 Introduction

Advances in both available data and computing power have begun to open the door to a greater role for machine learning (ML) in addressing some of our planet's most pressing environmental problems, such as the growing frequency and intensity of wildfire (Moritz et al. 2014), over-exploited fisheries (Worm et al. 2006), declining biodiversity (Dirzo et al. 2014), and zoonotic pandemics (Dobson et al. 2020). But will ML approaches really help us tackle a changing planet? Applications of ML in ecology have begun to illustrate the promise of two methods of ML: *supervised learning* (**joseph2020?**) and *unsupervised learning* (**unsupervised?**), but have so far largely overlooked the third and possibly most promising approach in the ML triad: *reinforcement learning* (RL). Three features distinguish RL from other ML methods in ways that are particularly well suited to addressing global ecological change issues:

1) RL is explicitly focused on the task of selecting actions in an uncertain and changing environment to maximize some objective,
2) RL does not require massive amounts of representative sampled historical data,
3) RL approaches easily integrate with existing ecological models and simulations, which may be our best guide to understanding and predicting future possibilities.

Despite relevance to uncertainty and decision making that could make RL uniquely well suited for ecological and conservation problems, it has so far seen little application in this area. To date, the problems considered by RL research have largely been drawn from examples in robotic movement and games like Go and Starcraft (**lillicrap?**; Silver et al. 2018; **alphastar?**) Complex environmental problems share many similarities to these tasks and games: the need to plan many moves ahead given a large number of possible outcomes, to account for uncertainty and to respond with contingency to the unexpected. RL agents typically develop their strategies by interacting with simulators, a practice that should not be unsettling to ecologists since learning from simulators is common across ecology. Rich, processes-based simulations such as the SORTIE model in forest management (Pacala et al. 1996), Ecopath with Ecosim in fisheries management (Steenbeek et al. 2016), or climate change policy models (Nordhaus 1992) are already used to explore scenarios and inform ecosystem management. Decision-theoretic approaches based on optimal control techniques can only find the best strategy in the simplest of ecological models; the so called "curse of dimensionality" makes problems with a large number of states or actions intractable by conventional methods (Wilson et al. 2006; Marescot et al. 2013). Neural-network-based RL techniques, referred to as *deep RL*, have proven particularly effective in problems involving complex, high-dimensional spaces that have previously proven intractable to classical methods. We believe that the application of RL methods to simulations of ecological processes will open the door to a new paradigm for tackling tough environmental change problems. We also believe that such a transformation could bring risks and pitfalls comparable and varied as its solutions.

In this paper, we draw on examples from fisheries management and ecological tipping points to illustrate how deep RL techniques can successfully discover optimal solutions to previously solved management scenarios and discover highly effective solutions to unsolved problems. Our simplest example approaches the known optimal solution for a classic fisheries management problem with no prior knowledge of the underlying model. Our second example considers ecosystem management under slow environmental degradation: a scenario which is not amenable to classic optimization methods such as dynamic programming. In this case there is no existing optimal solution to compare against, but an appropriately tuned RL agent can out-perform a sensible rule-of-thumb response. These examples demonstrate that RL-based approaches are capable but by no means a magic bullet: reasonable solutions require careful design of training environments, choice of RL algorithms, tuning and evaluation, as well as substantial computational power. Our examples are intentionally simple, aiming to provide a clear template for understanding that could be easily extended to cover more realistic conditions. We include an extensive appendix with carefully annotated code which should allow readers to both reproduce and extend this analysis. We further include implementations of three fully featured simulation modules following current standards for RL benchmark problems as Python modules distributed on PyPi and GitHub. Each module includes several more challenging variations to the examples described here. We intend to continue to expand this library of ecological management benchmarks with the hope both of challenging researchers already working in RL to tackle some of these pressing issues and for ecologists and environmental scientists to continue to refine these benchmarks into more realistic ecological simulations.

2 RL overview

Any application of RL can be divided into two components: an *environment* and an *agent*. An *environment* is typically a computer simulation, though it is possible to use the real world as the RL environment (**robotics?**). The *agent*, which is often a computer program, continuously interacts with the environment. At each time step, the agent observes the current *state* of the environment then performs an *action*; as a result of this action, the environment will transition to a new state and will transmit a numerical *reward* signal to the agent. The goal of the agent is to learn how to maximize its expected cumulative reward. The agent learns how to achieve this objective during a period called *training*. In training, the agent *explores* the available states and actions. Once the agent comes across a highly rewarding sequence of states and actions, the agent will reinforce this behavior so that it is more likely for the agent to *exploit* the same high reward trajectory in the future. Throughout this process, the agent's behavior is codified into what is called a *policy*, which describes what action an agent should take for a given state.

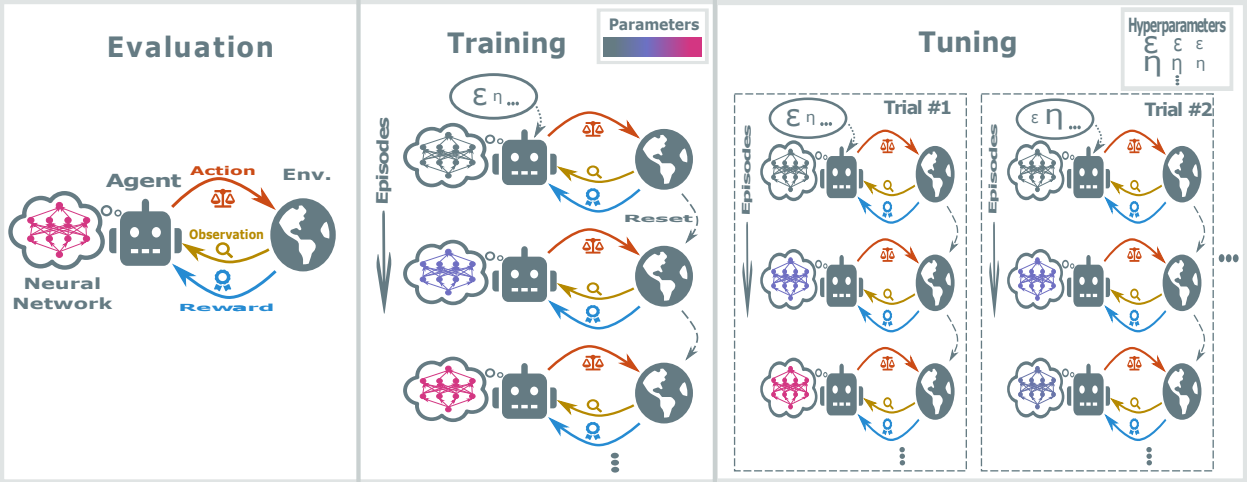


Figure 1: Deep Reinforcement Learning: A deep RL *agent* uses a *neural network* to select an *action* in response to an *observation* of the *environment*, and receives a *reward* from the environment as a result. During *training*, the agent tries to maximize its cumulative reward by interacting with the environment and learning from experience. In the RL loop, the agent performs an action, and the environment returns a reward as well as an observation of the environment's state. The agent-environment loop continues until the environment reaches a terminal state, after which the environment will reset, causing a new *episode* to begin. Across training episodes, the agent will continually update the *parameters* in its neural network, so that the agent will select better actions. Before training starts, the researcher must input a set of *hyperparameters* to the agent; hyperparameters direct the learning process and thus affect the outcome of training. A researcher finds the best set of hyperparameters during *tuning*. Hyperparameter tuning consists of iterative *trials*, in which the agent is trained with different sets of hyperparameters. At the end of a trial, the agent is evaluated to see which set of hyperparameters results in the highest cumulative reward. An agent is *evaluated* by recording the cumulative reward over one episode, or the mean reward over multiple episodes. Within evaluation, the agent does not update its neural network; instead, the agent only uses the neural network to select actions.

2.1 Designing and building an environment

An environment is a mathematical function, computer program, or real world experience that takes an agent's proposed *action* as input and returns an observation of the environment's current *state* and an associated *reward* as output. In contrast to classical approaches (Marescot et al. 2013), there are few restrictions on what comprises a state or action. States and actions may be continuous or discrete, completely or partially observed, single or multidimensional. The main focus of building an RL environment, however, is in the environment's transition dynamics and reward function. The designer of the environment can make the environment follow any transition and reward function provided that both are functions of the current state and action. This ability allows RL environments to model a broad range of decision making problems. For example, we can set the transitions to be deterministic or stochastic. We can also specify the reward function to be *sparse*, whereby a positive reward can only be received after a long sequence of actions, e.g. the end point in a maze. In other environments, an agent may have to learn to forgo immediate rewards (or even accept an initial negative reward) in order to maximize the net discounted reward as we illustrate in examples here.

Brockman et al. (2016) identified (1) the lack of standardization of environments used in publication and (2) the need for better benchmark environments as two core problems limiting more rapid advancement of RL research, and proposed the creation of the OpenAI `gym` framework to address both problems. The `gym` framework defines a standard interface and methods by which a developer can describe an arbitrary environment in a computer program. This allows the design and application of software agents which can interact and learn in that environment without knowing anything about the environment's internal details. The `gym` Python module created by OpenAI also provides over 100 pre-built environments that serve as benchmark problems against which new RL algorithms can be tested. The `gym` framework is now recognized by all major software frameworks for writing RL algorithms (see Appendix, 'Deep RL Frameworks'). While this library of environments spans a range of difficulties from classic control problems that can be solved

in closed form to complex three-dimensional robotics simulators, they include no examples of ecological or biological problems. We use the `gym` framework to turn several existing ecological models into valid environmental simulators that can similarly be used in any RL framework. In Appendix C, we give detailed instruction on how an OpenAI `gym` is constructed.

2.2 Training and tuning an agent

Training a deep RL agent on an environment involves allowing the agent to interact with the environment for thousands or even millions of time steps. The training period involves a trial and error process in which the agent inputs actions to the environment and uses the reward signal to learn an optimal policy. According to the algorithm that the agent follows, the agent will explore the state-action space and use its experience to improve its decisions. The amount of time needed for an agent to learn high reward yielding behavior cannot be predetermined and depends on a host of factors including the amount of dedicated compute, complexity of the environment, complexity of the agent and more. Yet, overall, it has been well established that deep, model-free RL agents tend to be very sample inefficient (Gu et al. 2017), so it is recommended to provide a generous training budget for these agents. An advantage of using deep, model-free algorithms is that since they function autonomously and do not require any knowledge about the environment, we can readily test how these algorithms perform without knowing anything about the environment's transition dynamics or the inner workings of the algorithm.

Hyperparameters refer to the parameters in a deep, RL algorithm that control the learning process, e.g. the step size to use for gradient descent steps. Since the optimal hyperparameters vary across environments and can not be predetermined (Henderson et al. 2019), it is necessary to find a good-performing set of hyperparameters in a process called hyperparameter tuning. Hyperparameter tuning consists of the following general steps in repetition: proposing a new set of hyperparameters, training an agent with these hyperparameters, then examining the resulting performance. This procedure can be automated through standard multi-dimensional optimization approaches. Since one round of training can be time intensive, hyperparameter tuning, which consists of usually numerous rounds of training, can be particularly time intensive. Hyperparameter tuning, however, is a necessary step in evaluating the performance of an algorithm and should be a standard component of any deep RL workflow.

3 Formal RL Concepts

In this brief section, we will introduce the concept of the RL environment as a partially observable Markov decision process (POMDP); we will then use POMDP definitions to define the RL objective.

The reinforcement learning environment is typically formalized as a discrete-time POMDP. A POMDP is a 7-tuple that consists of the following:

- $\mathcal{S}$: a set of states called the state space
- $\mathcal{A}$: a set of actions called the action space
- Ω : a set of observations called the observation space
- $E(o_t|s_t)$: an emission distribution, which accounts for an agent's observation being different from environment's state
- $T(s_{t+1}|s_t, a_t)$: a state transition operator which describes the dynamics of the system
- $r(s_t, a_t)$: a reward function
- $d_0(s_0)$: an initial state distribution
- $\gamma \in (0, 1]$: a discount factor

The agent interacts with the environment in an iterative loop, whereby the agent only has access to the observation space, action space and the discounted reward signal, $\gamma^t\, r(s_t, a_t)$. At each time step, the agent makes an observation of the environment at time t, o_t, and selects an action, a_t, according to its policy, π, which maps observations to a probability distribution over actions, $\pi : \Omega \to p(\mathcal{A} \mid \Omega)$. The action will cause the environment to transition to a new state according to the state transition operator. The agent will then receive a new observation and a discounted reward signal, $\gamma^t\, r(s_t, a_t)$, as feedback. This process will repeat until a terminal state is reached, after which the environment can reset to an initial state. As the agent interacts with the environment, the agent will create a trajectory in state-action space given as

$\tau = (s_0, a_0, \ldots, s_H, a_H)$ where H denotes the length of the state-action sequence. From these definitions, we can provide an agent's trajectory distribution for a given policy as,

$$p_\pi(\tau) = d_0(s_0) \prod_{t=0}^{H} \pi(a_t|o_t)\, E(o_t|s_t)\, T(s_{t+1}|s_t, a_t)$$

The goal of reinforcement learning is for the agent to find an optimal policy distribution, $\pi^*(a_t|o_t)$, that maximizes the expected return, $J(\pi)$:

$$\pi^* = \underset{\pi}{\text{argmax}}\ \mathbb{E}_{\tau \sim p_\pi(\tau)}\Big[\sum_{t=0}^{H} \gamma^t r(s_t, a_t)\Big] = \underset{\pi}{\text{argmax}}\ J(\pi)$$

Note that this formalism extends to an infinite horizon case when $H = \infty$. While there are RL methods to deal with infinite horizon cases, we will only present methods for finite horizon or episodic POMDPs. In the appendix, we will go into further detail on the general approaches that model-free deep RL algorithms take towards optimizing the RL objective.

4 Deep RL Algorithms

There is an ever increasing multitude of reinforcement learning algorithms, and while there are far too many algorithms to discuss in detail in this paper, we will herein classify and touch on some current methods.

To solve the RL problem, algorithms either take a *model-free* or *model-based* approach. The distinction is that *model-free* algorithms do not attempt to learn or use a model of the environment; yet, *model-based* algorithms employ a model of the environment to achieve the RL objective. A trade-off between these approaches is that when it is possible to quickly learn a model of the environment or the model is already known, model-based algorithms tend to require much less interaction with the environment to learn good-performing policies (Janner et al. 2019). Yet, frequently, learning a model of the environment is very difficult, and in these cases, model-free algorithms tend to outperform (Janner et al. 2019). Since our examples use model-free agents, we will focus on model-free algorithms in this section.

Among model-free methods, there is divergence around what functions the agent is attempting to learn in order to achieve the RL objective. Generally, model-free agents are either learning a policy function, a value function or both. For context, *value functions* are proxies for how high of a cumulative reward an agent can expect to receive from a given state or state-action pair. If an agent knows the value function, then the agent can find the optimal action at any state by selecting the action that will maximize the value function in expectation. The class of algorithms that exclusively try to learn a value function are called *value-based* methods. In contrast, *policy gradient* algorithms exclusively learn the policy. And, lastly, *actor-critic* algorithms attempt to learn both a policy and value function, whereby the value function is used to inform the agent on the goodness of a selected action. We go into greater depth on these different classes of RL algorithms in Appendix A.

Neural networks become useful in RL when the environment has a large state-action space, which happens frequently with realistic decision-making problems. Classic RL algorithms, like policy iteration, value iteration and TD-learning, fail when the state-action space is large, e.g. when the possible states and actions can not be represented in a tractable table (Sutton and Barto 2018). Yet, over the last 10 years, researchers have shown that neural network-based RL algorithms can perform well on previously unsolvable environments Silver et al. (2016). Neural networks have been widely useful in reinforcement learning because neural nets have the property of being general function approximators (Hornik, Stinchcombe, and White 1989). Whenever there is a need for an agent to approximate some function, say a policy function or value function, neural networks can be used in this capacity. While there are other function approximators that can be used in RL, e.g. Gaussian processes (Grande, Walsh, and How 2014), neural networks have excelled in this role because of their ability to learn complex, non-linear, high dimensional functions and their ability to adapt given new information (Arulkumaran et al. 2017). We will not discuss in detail how neural networks work (see LeCun, Bengio, and Hinton 2015), but the general process for how neural networks are used in RL is that at each learning step, the agent, will adjust the parameters in its neural network with which it approximates the policy function and/or value function, so that according to past experiences, the agent will be more likely to engage in highly rewarding behavior in the future. This same process occurs in value learning and

actor-critic-based deep RL algorithms, except these agents are adjusting either a value network or both a value and policy network respectively

Over the course of training, RL agents will engage in trial and error learning, whereby it will be necessary for the agent to explore new sequences of actions as well as exploit past high rewarding sequences. Occasionally, the agent will need to propose exploratory actions that may result in higher rewards than previously seen. The agent may also need to focus on past sequences of actions which may require marginal improvement to reach optimality. It is very difficult for the agent to know when it should engage in explorative or exploitative behavior. This *exploration-exploitation dilemma* is an open research problem (Berger-Tal et al. 2014). Current deep RL algorithms take a range of approaches towards this (see Appendix A). The manner in which RL algorithms explore depend on whether the algorithms are *on-policy* or *off-policy*. The distinction between on and off-policy evaluation comes from what policy the agent is using to select actions during training. For off-policy algorithms, the agent uses a policy during training that can be different from the policy that the agent will use during evaluation; while in on-policy algorithms, the agent updates the same policy in training as the agent will use during evaluation. Other points of divergence include whether the agent learns a deterministic or stochastic policy, and whether the algorithm is formulated to work on a discrete, continuous or both a discrete and continuous state and action space.

5 Examples

We provide two examples that illustrate the application and potential of deep RL to ecological and conservation problems, highlighting both the potential and the inherent challenges.

5.1 Sustainable harvest

The first example focuses on the important but well-studied problem of setting harvest quotas in fisheries management. Determining fishing quotas is both a critical ecological issue (Worm et al. 2006, 2009; Costello et al. 2016), and also a textbook example that has long informed the management of renewable resources within fisheries and beyond (Colin W. Clark 1990).

Given a population growth model that predicts the total biomass of a fish stock in the following year as a function of the current biomass, it is straight forward to determine what biomass corresponds to the maximum growth rate of the stock, or B_{MSY}, the biomass at Maximum Sustainable Yield (MSY) (Schaefer 1954). When the population growth rate is stochastic, the problem is slightly harder to solve, as the harvest quota must constantly adjust to the ups and downs of stochastic growth, but it is still possible to show the optimal strategy merely seeks to maintain the stock at B_{MSY}, adjusted for any discounting of future yields (Reed 1979).

This provides a natural first benchmark for deep RL approaches, since we can compare the RL solution to the mathematical optimum directly. The problem is also of interest because the RL approach can be extended to consider ecological dynamics that either surpass the complexity possible with SDP methods, or violate assumptions of such classical approaches). Moreover, fisheries management is both an important global challenge in it's own right (Worm et al. 2006; Costello et al. 2016) as well as a common test case to understand ecological dynamics and conservation management more broadly (**something?**).

For illustrative purposes, we consider the simplest version of the dynamic optimal harvest problem as outlined by (Colin W. Clark 1973) (for the deterministic case) and (Reed 1979) (under stochastic recruitment). The manager seeks to optimize the net present value (discounted cumulative catch) of a fishery, observing the stock size each year and setting an appropriate harvest quota in response. In the classical approach, the best model of the fish population dynamics must first be estimated from data, potentially with posterior distributions over parameter estimates reflecting any uncertainty. From this model, the optimal harvest policy – that is, the function which returns the optimal quota for each possible observed stock size – can be determined either by analytic (Reed 1979) or numerical (Marescot et al. 2013) methods, depending on the complexity of the model. In constrast, a model-free deep RL algorithm makes no assumption about the precise functional form or parameter values underlying the dynamics – it is in principle agnostic to the details of the simulation.

We illustrate the deep RL approach using the algorithm known as Twin Delayed Deep Deterministic Policy Gradient or more simply, TD3 (Fujimoto, Hoof, and Meger 2018). TD3 is an extension of Deep Deterministic Policy Gradient (DDPG, **DDPG?**), which itself extended the original Deep Q-Network (DQN, Mnih et al. 2015) algorithm to continuous action space. A step-by-step walk-through for training both agents on this

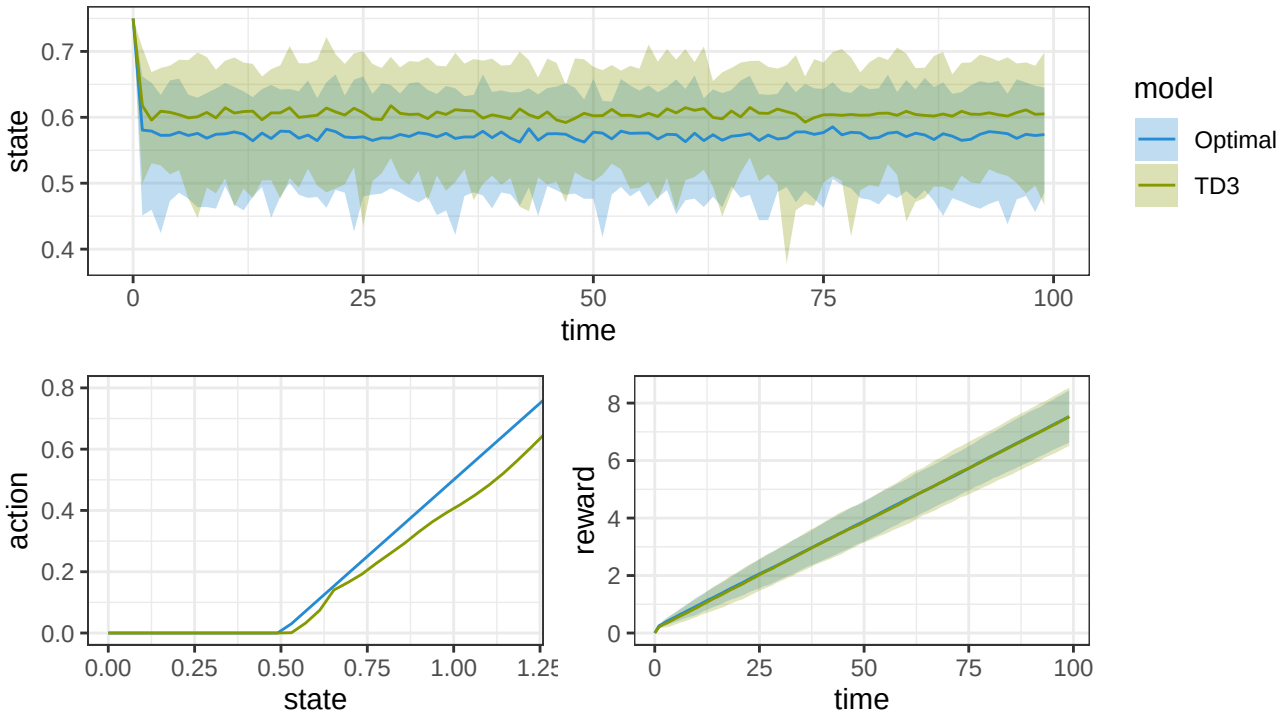


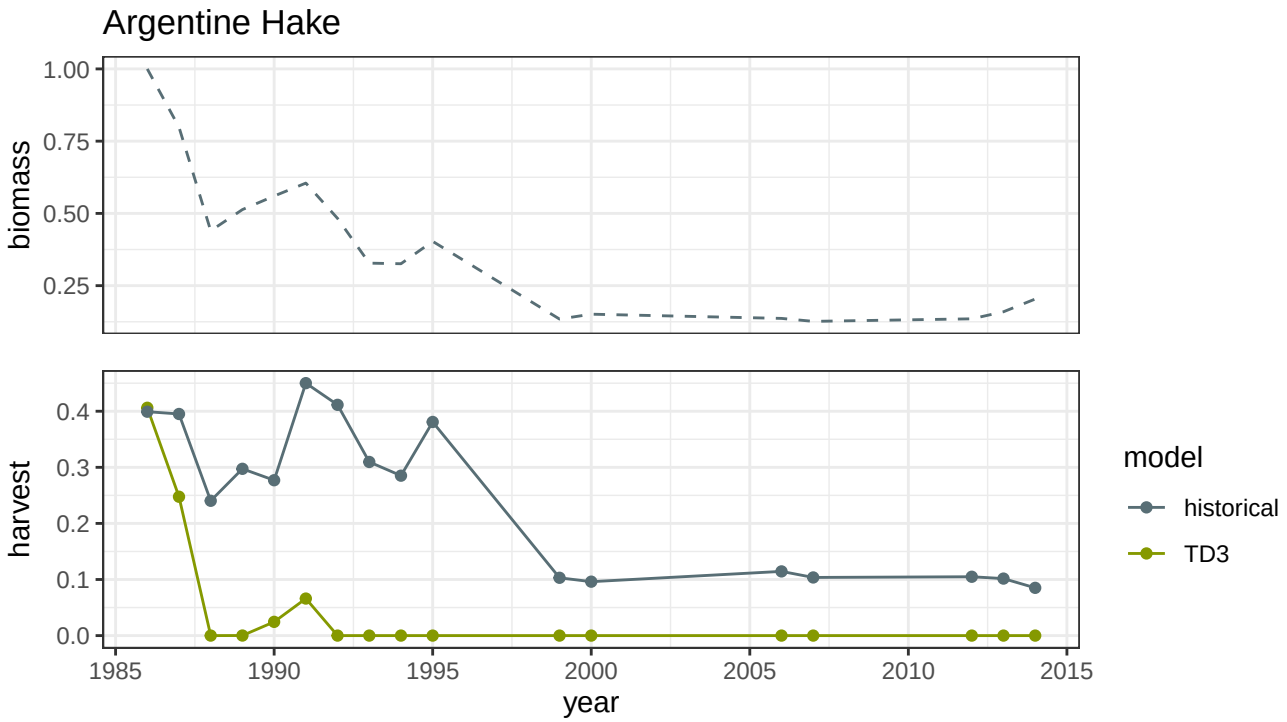
Figure 2: Fisheries management using neural network agents trained with RL algorithm TD3 compared to optimal management. Top panel: mean fish population size over time across 100 replicates. Shaded region shows the 95% confidence interval over simulations. Lower left: The optimal solution is policy of constant escapement. Below the target escapement of 0.5, no harvest occurs, while any stock above that level is immediately harvested back down (red line). RL agents both adopt a policies that cease any harvest below this level, while allowing a somewhat higher escapement than optimal just above the optimal escapement level. TD3 achieves nearly-optimal mean utility.

environment is provided in the Appendix. We compare the resulting management, policy, and reward under either RL agent to that achieved by the optimal management solution [Fig 2]. Despite having no knowledge of the underlying model, the RL agent learns enough to achieve nearly optimal performance.

The cumulative reward (utility) realized across 100 stochastic replicates is indistinguishable from that of the optimal policy [Fig 2]. Nevertheless, comparing the mean state over replicate simulations reveals some differences in the RL strategy, wherein the stock is maintained at a slightly higher-than-optimal biomass. Because our state space and action space are sufficiently low-dimensional in this example, we are also able to visualize the policy function directly, and compare to the optimal policy [Fig 2]. This confirms that quotas tend to be slightly lower than optimal, most notably at larger stock sizes. These features highlight a common challenge in the design and training of RL algorithms. RL cares only about improving the realized cumulative reward, and may sometimes achieve this in unexpected ways. Because these simulations rarely reach stock sizes at or above carrying capacity, these larger stock sizes show a greater deviation from the optimal policy than we observe at more frequently visited lower stock sizes. Training these agents in a variety of alternative contexts can improve there ability to generalize to other scenarios.

How would an RL agent be applied to empirical data? In principle, this is straight forward. Once our agent has been trained on appropriate simulation environments, it is able to propose a quota given an observation of the stock after appropriate transformation of variables. To illustrate this, we examine the quota that would be recommended by our newly trained RL agent, above, against historical harvest levels of Argentine hake based on stock assessments from 1986 - 2014 (**ram?**, see Appendix D). Hake stocks showed a marked decline throughout this period, while harvests deceased only in proportion [Fig 3]. In contrast, our RL agent would have recommended significantly lower quotas over most of the same interval, including the closure of the fishery as stocks were sufficiently depleted. While we have no way of knowing for sure if the RL quotas would

have led to recovery, let alone an optimal harvest rate, the contrast between those recommended quotas and the historical catch is notable.



This approach is not as different from conventional strategies at it may seem. In a conventional approach, ecological models are first estimated from empirical data, (stock assessments in the fisheries case). Quotas can then be set based directly on these model estimates, or by comparing alternative candidate "harvest control rules" (the policy function) against model-based simulations of stock dynamics. This latter approach, known in fisheries as Management Strategy Evaluation [MSE; (**Punt2016?**)] is already very closely analogous to the RL process. Instead of researchers evaluating a handful of control rules, the RL agent takes on the task of proposing and evaluating the millions of possible control rules represented by its neural network.

5.2 Ecological Tipping Points

Our second example focuses on a case for which we do not already have an existing, provably optimal policy to compare against. We consider the generic problem of an ecosystem facing slowly deteriorating environmental conditions which move the dynamics closer towards a tipping point [Fig 2]. This model of a critical transition has been posited widely in ecological systems, from the simple consumer-resource model of (May 1977) on which our dynamics are based, to microbial dynamics (Dai et al. 2012) lake ecosystem communities (Carpenter et al. 2011) to planetary ecosystems (Barnosky et al. 2012). On top of these ecological dynamics we introduce an explicit ecosystem service model quantifying the value of more desirable 'high' state relative to the 'low' state. For simplicity, we assume a proportional benefit b associated with the ecosystem state $X(t)$. Thus when the ecosystem is near the 'high' equilibrium $\hat{X}_H$, the corresponding ecosystem benefit $b\hat{X}_H$ is higher than at the low equilibrium, bx_L, consistent with the intuitive description of ecosystem tipping points (Barnosky et al. 2012).

We also enumerate the possible actions which a manager may take in response to environmental degradation. In the absence of any management response, we assume the environment deteriorates at a fixed rate α, which can be thought of as the incremental increase in global mean temperature or similar anthropogenic forcing term. Management can respond to slow or even reverse this trend by choosing an opposing action A_t. We assume that large actions are proportionally more costly than small actions, consistent with the expectation of diminishing returns: taking the cost associated with an action A_t as equal to cA_t^2. Many alterations of these basic assumptions are also possible: our `gym_conservation` implements a range of different scenarios

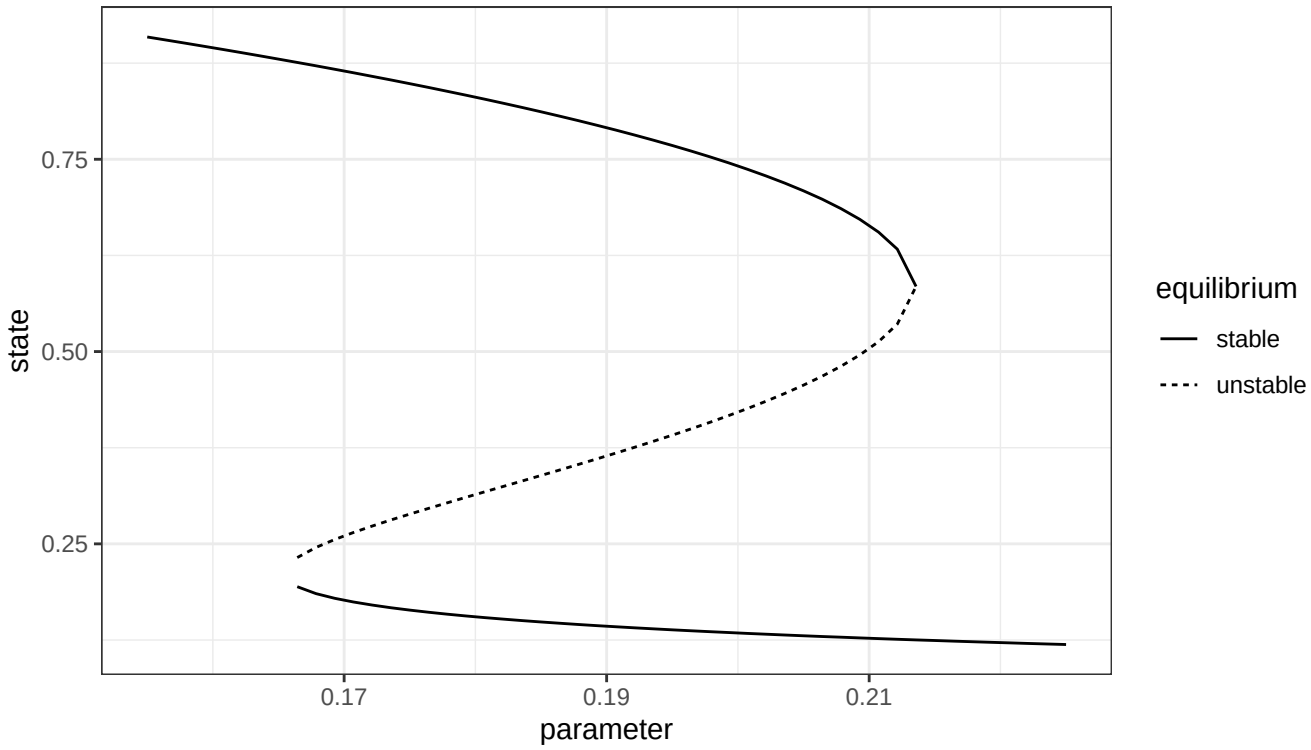


Figure 3: Bifurcation diagram for tipping point scenario. The ecosystem begins in the desirable 'high' state under an evironmental parameter (e.g. global mean temperature, arbitrary units) of 0.19. In the absence of conservation action, the environment worsens (e.g. rising mean temperature) as the parameter increases. This results in only a slow degredation of the stable state, until the parameter crosses the tipping point threshold at about 0.215, where the upper stable branch is anihilated in a fold bifurcation and the system rapidly transitions to lower stable branch, around state of 0.1. Recovery to the upper branch requires a much greater conservation investment, reducing the parameter all the way to 0.165 where the reverse bifurcation will carry it back to the upper stable branch.

with user-configurable settings. In each case, the manager observes the current state of the system each year and must then select the policy response that year.

Because this problem involves a parameter whose value changes over time (the slowly deteriorating environment), the resulting ecosystem dynamics are not autonomous. This precludes our ability to solve for the optimal management policy using classical theory such as for Markov Decision Processes (MDP, (Marescot et al. 2013)), typically used to solve sequential decision-making problems. However, it is often argued that simple rules can achieve nearly optimal management of ecological conservation objectives in many cases (**Possingham?**; **Possingham?**; **morePossingham?**). A common conservation strategy employs a fixed response level rather than a dynamic policy which is toggled up or down each year: for example, declaring certain regions as protected areas in perpetuity. An intuitive strategy faced with an ecosystem tipping point would be 'perfect conservation,' in which the management response is perfectly calibrated to counter-balance any further decline. While the precise rate of such decline may not be known in practice (and will not be known to RL algorithms before-hand either), it is easy to implement such a policy in simulation for comparative purposes. We compare this rule-of-thumb to the optimal policy found by training an agent using the TD3 algorithm.

The TD3-trained agent proves far more successful in preventing chance transitions across the tipping point, consistently achieving a higher cumulative ecosystem service value across replicates than the steady-state strategy.

Examining the replicate management trajectories and corresponding rewards [Fig 3] reveal that the RL agent incurs significantly higher costs in the initial phases of the simulation, dipping the cumulative reward into

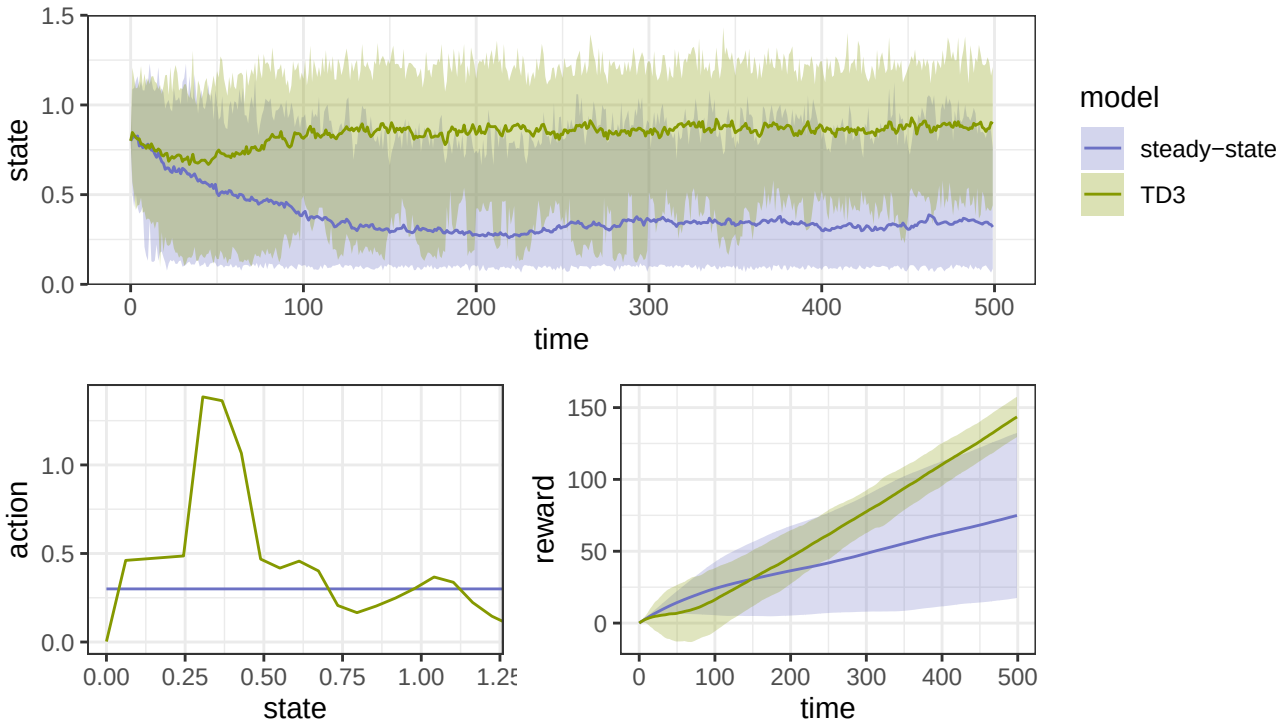


Figure 4: Ecosystem dynamics under management using the steady-state rule-of-thumb strategy compared to management using a neural network trained using the TD3 RL algorithm. Top panel: mean and 95% confidence interval of ecosystem state over 100 replicate simulations. As more replicates cross the tipping point threshold under steady-state strategy, the mean slowly decreases, while the TD3 agent preserves most replicates safely above the tipping point. Lower left: the policy function learned using TD3 relative to the policy function under the steady state. Lower right: mean rewards under TD3 management evenutally exceed those expected under the steady state strategy as a large initial investment in conservation eventually pays off.

negative territory on average and well below the reward realized by the simple steady-state strategy. This initial investment then begins to pay off – by about the 200th time step the RL agent has surpassed the performance of the steady state strategy. The policy plot provides more intuition for the RL agent's strategy: at very high state values, the RL agent opts for no conservation action – so far from the tipping point, no response is required. Near the tipping point, the RL agent steeply ramps up the conservation effort, and retains this effort even as the system falls below the critical threshold, where a sufficiently aggressive response can tip the system back into recovery. For a system at or very close to the zero-state, the RL agent gives up, opting for no action. Recall that the quadratic scaling of cost makes the rapid response of the TD3 agent much more costly to achieve the same net environmental improvement divided into smaller increments over a longer timeline. However, our RL agent has discovered that the extra investment for a rapid response is well justified as the risk of crossing a tipping point increases.

A close examination of the trajectories of individual simulations which cross the tipping point under either management strategy [see appendix B] further highlights the difference between these two approaches. Under the steady-state strategy, the system remains poised too close to the tipping point: stochastic noise eventually drives most replicates across the threshold, where the steady-state strategy is too weak to bring them back once they collapse. As replicate after replicate stochastically crashes, the mean state and mean reward bend increasingly downwards. In contrast, the RL agent edges the system slightly farther away from the tipping point, decreasing but not eliminating the chance of a chance transition. In the few replicates that experience a critical transition anyway, the RL agent usually responds with sufficient commitment to ensure their recovery [Appendix B]. Only 3 out of 100 replicates degrade far enough for the RL agent to give up the high cost of trying to rescue them. The RL agent's use of a more dynamic strategy out-preforms the steady-state strategy. Numerous kinks visible in the RL policy function also suggest that this solution is not yet optimal. Further

tuning of hyper-parameters, increased training, alterations or alternatives to the training algorithm would likely be able to further improve upon this performance. Nevertheless, such quirks are likely to be common features of RL solutions – long as they have minimal impact on realized rewards.

5.3 Additional Environments

Ecology holds many open problems for deep RL.
It is straight-forward to extend the simple environments presented here to reflect greater biological complexity, or to consider a host of more realistic decision scenarios. We provide an initial library of example environments at `https://boettiger-lab.github.io/conservation-gym`. Some environments in this library include a wildfire `gym` that poses the problem on wildfire suppression with a cellular automata model, an epidemic `gym` that examines timing of interventions to curb disease spread, as well as more complex variations of the fishing and conservation environments presented above.

6 Discussion

Ecological challenges facing the planet today are complex and outcomes are both uncertain and consequential. Even our best models and best research will never provide a crystal ball to the future, only better elucidate possible scenarios. Consequently, that research must also confront the challenge of making robust, resilient decisions in a changing world. The science of ecological management and quantitative decision-making has a long history (e.g. Schaefer 1954; Walters and Hilborn 1978) and remains an active area of research (Wilson et al. 2006; Fischer et al. 2009; Polasky et al. 2011). However, the limitations of classical methods such as optimal control frequently constrain applications to relatively simplified models (Wilson et al. 2006), ignoring elements such as spatial or temporal heterogeneity and autocorrelation, stochasticity, imperfect observations, age or state structure or other sources of complexity that are both pervasive and influential on ecological dynamics (Hastings and Gross 2012). Complexity comes not only from the ecological processes but also the available actions. Conservation efforts can be implemented in a wide variety of ways, and the most effective policies may vary in both space and time. While such complex dynamics and large state and action spaces are intractable to classical methods, they are straight forward to implement in simulation environments such as those used to train an RL algorithm. Neural networks used in deep RL have proven remarkably effective in handling such complexity, particularly when leveraging immense computing resources increasingly available through advances in hardware and software (Matthews 2018).

The rapidly expanding class of Model-free RL algorithms is particularly appealing given the ubiquitous presence of model uncertainty in ecological dynamics. Rarely do we know the underlying functional forms for ecological processes – is population recruitment more Beverton-Holt or Ricker? Methods which must first assume something about the structure or functional form of a process before estimating the corresponding parameter can only ever be as good as those structural assumptions. Frequently available ecological data is insufficient to distinguish between possible alternative models (**devalpine?**), or the correct model may be non-identifiable with any amount of data. Model-free RL approaches offer a powerful way to factor this thorny issue. The model-free RL agent needs to make no such a priori hypothesis about a particular functional form. Model-free approaches have proven successful at learning effective policies even when the underlying model may be difficult or impossible to learn (Pong et al. 2020), as long as simulations of possible mechanisms are available.

The examples presented here only scrape the surface of possible RL applications to conservation problems. The examples we have focused on are intentionally quite simple, though it is worth remembering that these very same simple models have a long history of relevance and application in both research and policy contexts. Despite their simplicity, the optimal strategy is not always obvious before hand, however intuitive it may appear in retrospect. In the case of the ecosystem tipping point scenario, the optimal strategy is unknown, and the approximate solution found by our RL implementation could almost certainly be improved upon. Even at it's best, an RL-trained agent can only be as good as the scenarios on which it has been trained. In these simple examples in which the simulation implements a single model, training is analogous to classical methods which take the model as given (Marescot et al. 2013). But classical approaches can be difficult to generalize when the underlying model is unknown: alternative models may require a completely different approach. In contrast, the process of training an RL algorithm on a more complex problem is no different than the simple one: we only need access to a simulation which can generate plausible future states in response to possible actions. Many of our most realistic ecological models exist only as numerical simulations of lower-level dynamics, such as the individual-based models used in the management of forests and wildfire

(Pacala et al. 1996; Moritz et al. 2014), disease (Dobson et al. 2020), marine ecosystems (Steenbeek et al. 2016), or global climate change (Nordhaus 1992).

Successfully applying RL to complex ecological problems is no easy task. The flexibility of the approach makes it relatively straightforward to include realistic biology and environmental science in framing an RL problem. Yet, even on relatively uncomplicated environments, training an RL agent can be more challenging than expected due to an entanglement of reasons like hyperparameter instability and poor exploration that can be very difficult to resolve (Henderson et al. 2019). As the examples we have presented illustrate, it is important to begin with simple problems, including those for which an optimal strategy is already known. Such examples provide important benchmarks against which to calibrate the performance, tuning and training requirements of RL, and can easily be extended into more complex problems once RL agents can master the basics. In the case that an agent has been trained to perform well on a realistic environment, there will still be a range of open questions in using this agent to inform decision-making:

- Since deep neural networks lack transparency (Castelvecchi 2016), can we be confident that the deep RL agent will generalize its past experience to new situations?
- Given that there have been many examples of reward misspecification leading to undesirable behavior (Hadfield-Menell et al. 2020), what if we have selected an objective that unexpectedly causes damaging behavior?

Deep RL is still a very young field, where despite several landmark successes, potential far outstrips practice. Recent advances in the field have proven the potential of the approach to solve complex problems (Silver et al. 2016, 2017, 2018; Mnih et al. 2015), but typically leveraging large teams with decades of experience in ML and millions of dollars worth of computing power (Silver et al. 2017). Successes have so far been concentrated in applications to games and robotics, not scientific and policy domains, though this is beginning to change (**protein_folding?**). Leading developers of new deep RL algorithms benchmark the performance of their algorithms against such problems in part because simulations of games and robotic movement are readily available through open source frameworks designed to interface with RL algorithms [brockman2016], and performance can easily be compared against alternatives on public leaderboards [brockman2016]. Iterative improvements to well-posed public challenges have proven immensely effective in the computer science community in tackling difficult problems, which allow many teams with diverse expertise not only to compete but to learn from each other (Villarroel, Taylor, and Tucci 2013; Deng et al. 2009). Advances in both computational power and algorithmic efficiency continue to reduce costs at a dramatic pace, significantly faster than Moore's law (Hernandez and Brown 2020). By working to develop similarly well-posed challenges as clear benchmarks, ecology and environmental science researchers may both be better able to replicate that collaborative, iterative success in cracking hard problems. Such benchmarks may further improve collaboration across disciplines as well, attracting the attention of AI expertise to these key issues facing our planet. If RL opens the door to greater role for artificial intelligence and engagement of technology firms in the machinery that determines environmental policy, this raises new concerns as well as new opportunities.

The increasingly influential role of AI in sectors as diverse as policing, finance, and employment has raised concern about the ethics, biases, and power entrenched in both decision-making algorithms and their outputs (Kalluri 2020). As algorithms play a greater role in environmental and conservation policy, will similar issues arise? While this question has garnered recent interest in the academic literature (Wearn, Freeman, and Jacoby 2019; Adams 2019; Scoville et al. 2021), the use of decision making algorithms in environmental contexts is still relatively isolated and in it's early stages. Examples of AI applications, and corresponding concerns, have largely been confined to the context of enforcing environmental laws (i.e. algorithms which predict wildlife poaching events (Adams 2019) or identify illegal fishing behavior (McDonald et al. 2021; Swartz et al. 2021)). However, more diverse applications of AI algorithms - such as applications which contribute to meeting sustainable development goals (Vinuesa et al. 2020) or designing protected area networks - raise additional concerns of entrenching power asymmetries between those reliant on natural resources and those designing algorithms and their objectives. These issues of power become particularly acute when private industry holds the keys to propriety algorithms or environmental data, and generally operate without the same community engagement obligations of public agencies. In these cases, good intentions might not be sufficient to ensure equitable outcomes. Interdisciplinary collaborations with experts in algorithmic governance and ethics will be important ensure that applications of AI to ecological challenges promotes equity in both the procedural process of developing objective functions and the distribution of benefits that result from decisions.

References

Adams, William M. 2019. "Geographies of Conservation II: Technology, Surveillance and Conservation by Algorithm." *Progress in Human Geography* 43 (2): 337–50.

Arulkumaran, Kai, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. "A Brief Survey of Deep Reinforcement Learning." *IEEE Signal Processing Magazine* 34 (6): 26–38. https://doi.org/10.1109/MSP.2017.2743240.

Barnosky, Anthony D., Elizabeth A. Hadly, Jordi Bascompte, Eric L. Berlow, James H. Brown, Mikael Fortelius, Wayne M. Getz, et al. 2012. "Approaching a State Shift in Earth's Biosphere." *Nature* 486 (7401): 52–58. https://doi.org/10.1038/nature11018.

Berger-Tal, Oded, Jonathan Nathan, Ehud Meron, and David Saltz. 2014. "The Exploration-Exploitation Dilemma: A Multidisciplinary Framework." *PLOS ONE* 9 (4): e95693. https://doi.org/10.1371/journal.pone.0095693.

Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. "OpenAI Gym." *arXiv:1606.01540 [Cs]*, June. http://arxiv.org/abs/1606.01540.

Carpenter, Stephen R, J. J. Cole, Michael L Pace, Ryan D. Batt, William A Brock, Timothy J. Cline, J. Coloso, et al. 2011. "Early Warnings of Regime Shifts: A Whole-Ecosystem Experiment." *Science (New York, N.Y.)* 1079 (April). https://doi.org/10.1126/science.1203672.

Castelvecchi, Davide. 2016. "Can We Open the Black Box of AI?" *Nature News* 538 (7623): 20. https://doi.org/10.1038/538020a.

Clark, Colin W. 1973. "Profit Maximization and the Extinction of Animal Species." *Journal of Political Economy* 81 (4): 950–61. https://doi.org/10.1086/260090.

Clark, Colin W. 1990. *Mathematical Bioeconomics: The Optimal Management of Renewable Resources, 2nd Edition*. Wiley-Interscience.

Costello, Christopher, Daniel Ovando, Tyler Clavelle, C. Kent Strauss, Ray Hilborn, Michael C. Melnychuk, Trevor A Branch, et al. 2016. "Global fishery prospects under contrasting management regimes." *Proceedings of the National Academy of Sciences* 113 (18): 5125–29. https://doi.org/10.1073/pnas.1520420113.

Dai, Lei, Daan Vorselen, Kirill S Korolev, and J. Gore. 2012. "Generic Indicators for Loss of Resilience Before a Tipping Point Leading to Population Collapse." *Science (New York, N.Y.)* 336 (6085): 1175–77. https://doi.org/10.1126/science.1219805.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. "ImageNet: A Large-Scale Hierarchical Image Database." In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–55. Miami, FL: IEEE. https://doi.org/10.1109/CVPR.2009.5206848.

Dirzo, Rodolfo, Hillary S Young, Mauro Galetti, Gerardo Ceballos, Nick JB Isaac, and Ben Collen. 2014. "Defaunation in the Anthropocene." *Science* 345 (6195): 401–6.

Dobson, Andrew P., Stuart L. Pimm, Lee Hannah, Les Kaufman, Jorge A. Ahumada, Amy W. Ando, Aaron Bernstein, et al. 2020. "Ecology and Economics for Pandemic Prevention." *Science* 369 (6502): 379–81. https://doi.org/10.1126/science.abc3189.

Fischer, Joern, Garry D Peterson, Toby A. Gardner, Line J Gordon, Ioan Fazey, Thomas Elmqvist, Adam Felton, Carl Folke, and Stephen Dovers. 2009. "Integrating Resilience Thinking and Optimisation for Conservation." *Trends in Ecology & Evolution* 24 (10): 549–54. https://doi.org/10.1016/j.tree.2009.03.020.

Fujimoto, Scott, Herke van Hoof, and David Meger. 2018. "Addressing Function Approximation Error in Actor-Critic Methods." *arXiv:1802.09477 [Cs, Stat]*, October. http://arxiv.org/abs/1802.09477.

Grande, Robert, Thomas Walsh, and Jonathan How. 2014. "Sample Efficient Reinforcement Learning with Gaussian Processes." In *Proceedings of the 31st International Conference on Machine Learning*, edited by Eric P. Xing and Tony Jebara, 32:1332–40. Proceedings of Machine Learning Research 2. Bejing, China: PMLR. http://proceedings.mlr.press/v32/grande14.html.

Gu, Shixiang, Timothy Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. 2017. "Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic." *arXiv:1611.02247 [Cs]*, February. http://arxiv.org/abs/1611.02247.

Hadfield-Menell, Dylan, Smitha Milli, Pieter Abbeel, Stuart Russell, and Anca Dragan. 2020. "Inverse Reward Design." *arXiv:1711.02827 [Cs]*, October. http://arxiv.org/abs/1711.02827.

Hastings, Alan, and Louis J. Gross, eds. 2012. *Encyclopedia of Theoretical Ecology.* Oakland, CA: University of California Press.

Henderson, Peter, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2019. "Deep Reinforcement Learning That Matters." *arXiv:1709.06560 [Cs, Stat]*, January. http://arxiv.org/abs/1709.06560.

Hernandez, Danny, and Tom B. Brown. 2020. "Measuring the Algorithmic Efficiency of Neural Networks." *arXiv:2005.04305 [Cs, Stat]*, May. http://arxiv.org/abs/2005.04305.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. 1989. "Multilayer Feedforward Networks Are Universal Approximators." *Neural Networks* 2 (5): 359–66. https://doi.org/10.1016/0893-6080(89)90020-8.

Janner, Michael, Justin Fu, Marvin Zhang, and Sergey Levine. 2019. "When to Trust Your Model: Model-Based Policy Optimization." *arXiv:1906.08253 [Cs, Stat]*, November. http://arxiv.org/abs/1906.08253.

Kalluri, Pratyusha. 2020. "Don't Ask If Artificial Intelligence Is Good or Fair, Ask How It Shifts Power." *Nature* 583 (7815): 169–69.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep Learning." *Nature* 521 (7553): 436–44. https://doi.org/10.1038/nature14539.

Marescot, Lucile, Guillaume Chapron, Iadine Chadès, Paul L. Fackler, Christophe Duchamp, Eric Marboutin, and Olivier Gimenez. 2013. "Complex Decisions Made Simple: A Primer on Stochastic Dynamic Programming." *Methods in Ecology and Evolution* 4 (9): 872–84. https://doi.org/10.1111/2041-210X.12082.

Matthews, David. 2018. "Supercharge Your Data Wrangling with a Graphics Card." *Nature* 562 (7725): 151–52. https://doi.org/10.1038/d41586-018-06870-8.

May, Robert M. 1977. "Thresholds and Breakpoints in Ecosystems with a Multiplicity of Stable States." *Nature* 269 (5628): 471–77. https://doi.org/10.1038/269471a0.

McDonald, Gavin G, Christopher Costello, Jennifer Bone, Reniel B Cabral, Valerie Farabee, Timothy Hochberg, David Kroodsma, Tracey Mangin, Kyle C Meng, and Oliver Zahn. 2021. "Satellites Can Reveal Global Extent of Forced Labor in the World's Fishing Fleet." *Proceedings of the National Academy of Sciences* 118 (3).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, et al. 2015. "Human-Level Control Through Deep Reinforcement Learning." *Nature* 518 (7540): 529–33. https://doi.org/10.1038/nature14236.

Moritz, Max A., Enric Batllori, Ross A. Bradstock, A. Malcolm Gill, John Handmer, Paul F. Hessburg, Justin Leonard, et al. 2014. "Learning to Coexist with Wildfire." *Nature* 515 (7525): 58–66. https://doi.org/10.1038/nature13946.

Nordhaus, W. D. 1992. "An Optimal Transition Path for Controlling Greenhouse Gases." *Science* 258 (5086): 1315–19. https://doi.org/10.1126/science.258.5086.1315.

Pacala, Stephen W., Charles D. Canham, John Saponara, John A. Silander, Richard K. Kobe, and Eric Ribbens. 1996. "Forest Models Defined by Field Measurements: Estimation, Error Analysis and Dynamics." *Ecological Monographs* 66 (1): 1–43. https://doi.org/10.2307/2963479.

Polasky, Stephen, Stephen R. Carpenter, Carl Folke, and Bonnie Keeler. 2011. "Decision-making under great uncertainty: environmental management in an era of global change." *Trends in Ecology & Evolution* 26 (8): 398–404. https://doi.org/10.1016/j.tree.2011.04.007.

Pong, Vitchyr, Shixiang Gu, Murtaza Dalal, and Sergey Levine. 2020. "Temporal Difference Models: Model-Free Deep RL for Model-Based Control." *arXiv:1802.09081 [Cs]*, February. http://arxiv.org/abs/1802.09081.

Reed, William J. 1979. "Optimal escapement levels in stochastic and deterministic harvesting models." *Journal of Environmental Economics and Management* 6 (4): 350–63. https://doi.org/10.1016/0095-0696(79)90014-7.

Schaefer, Milner B. 1954. "Some aspects of the dynamics of populations important to the management of the commercial marine fisheries." *Bulletin of the Inter-American Tropical Tuna Commission* 1 (2): 27–56. https://doi.org/10.1007/BF02464432.

Scoville, Caleb, Melissa Chapman, Razvan Amironesei, and Carl Boettiger. 2021. "Algorithmic Conservation in a Changing Climate." *Current Opinion in Environmental Sustainability* 51: 30–35.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, et al. 2016. "Mastering the Game of Go with Deep Neural Networks and Tree Search." *Nature* 529 (7587): 484–89. https://doi.org/10.1038/nature16961.

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, et al. 2018. "A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go Through Self-Play." *Science* 362 (6419): 1140–44. https://doi.org/10.1126/science.aar6404.

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, et al. 2017. "Mastering the Game of Go Without Human Knowledge." *Nature* 550 (7676): 354–59. https://doi.org/10.1038/nature24270.

Steenbeek, Jeroen, Joe Buszowski, Villy Christensen, Ekin Akoglu, Kerim Aydin, Nick Ellis, Dalai Felinto, et al. 2016. "Ecopath with Ecosim as a Model-Building Toolbox: Source Code Capabilities, Extensions, and Variations." *Ecological Modelling* 319 (January): 178–89. https://doi.org/10.1016/j.ecolmodel.2015.06.031.

Sutton, Richard S, and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT press.

Swartz, Wilf, Andrés M Cisneros-Montemayor, Gerald G Singh, Patrick Boutet, and Yoshitaka Ota. 2021. "AIS-Based Profiling of Fishing Vessels Falls Short as a 'Proof of Concept' for Identifying Forced Labor at Sea." *Proceedings of the National Academy of Sciences* 118 (19).

Villarroel, J. Andrei, John E. Taylor, and Christopher L. Tucci. 2013. "Innovation and Learning Performance Implications of Free Revealing and Knowledge Brokering in Competing Communities: Insights from the Netflix Prize Challenge." *Computational and Mathematical Organization Theory* 19 (1): 42–77. https://doi.org/10.1007/s10588-012-9137-7.

Vinuesa, Ricardo, Hossein Azizpour, Iolanda Leite, Madeline Balaam, Virginia Dignum, Sami Domisch, Anna Felländer, Simone Daniela Langhans, Max Tegmark, and Francesco Fuso Nerini. 2020. "The Role of Artificial Intelligence in Achieving the Sustainable Development Goals." *Nature Communications* 11 (1): 1–10.

Walters, Carl J, and Ray Hilborn. 1978. "Ecological Optimization and Adaptive Management." *Annual Review of Ecology and Systematics* 9 (1): 157–88. https://doi.org/10.1146/annurev.es.09.110178.001105.

Wearn, Oliver R, Robin Freeman, and David MP Jacoby. 2019. "Responsible AI for Conservation." *Nature Machine Intelligence* 1 (2): 72–73.

Wilson, Kerrie A., Marissa F. McBride, Michael Bode, and Hugh P. Possingham. 2006. "Prioritizing Global Conservation Efforts." *Nature* 440 (7082): 337–40. https://doi.org/10.1038/nature04366.

Worm, Boris, Edward B Barbier, Nicola Beaumont, J Emmett Duffy, Carl Folke, Benjamin S Halpern, Jeremy B C Jackson, et al. 2006. "Impacts of biodiversity loss on ocean ecosystem services." *Science (New York, N.Y.)* 314 (5800): 787–90. https://doi.org/10.1126/science.1132294.

Worm, Boris, Ray Hilborn, Julia K Baum, Trevor A Branch, Jeremy S Collie, Christopher Costello, Michael J Fogarty, et al. 2009. "Rebuilding global fisheries." *Science (New York, N.Y.)* 325 (5940): 578–85. https://doi.org/10.1126/science.1173146.