ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230311

Worrisome Properties of Neural Network Controllers and Their Symbolic Representations

Jacek Cyranka [a;*], Kevin E M Church[b] and Jean-Philippe Lessard[c]

[a]Institute of Informatics, University of Warsaw
[b]Centre de Recherches Mathématiques, Université de Montréal
[c]Department of Mathematics and Statistics, McGill University

Abstract. We raise concerns about controllers' robustness in simple reinforcement learning benchmark problems. We focus on neural network controllers and their low neuron and symbolic abstractions. A typical controller reaching high mean return values still generates an abundance of persistent low-return solutions, which is a highly undesirable property, easily exploitable by an adversary. We find that the simpler controllers admit more persistent bad solutions. We provide an algorithm for a systematic robustness study and prove existence of persistent solutions and, in some cases, periodic orbits, using a computer-assisted proof methodology.

1 Introduction

The study of neural network (NN) robustness properties has a long history in the research on artificial intelligence (AI). Since establishing the existence of so-called adversarial examples in deep NNs in [14], it is well known that NN can output unexpected results by slightly perturbing the inputs and hence can be exploited by an adversary. Since then, the robustness of other NN architectures has been studied [44]. In the context of control design using reinforcement learning (RL), the robustness of NN controllers has been studied from the adversarial viewpoint [29, 42]. Due to limited interpretability and transparency, deep NN controllers are not suitable for deployment for critical applications. Practitioners prefer abstractions of deep NN controllers that are simpler and human-interpretable. Several classes of deep NN abstractions exist, including single layer or linear nets, programs, tree-like structures, and symbolic formulas. It is hoped that such abstractions maintain or improve a few key features: generalizability – the ability of the controller to achieve high performance in similar setups (e.g., slightly modified native simulator used in training); deployability – deployment of the controller in the physical world on a machine, e.g., an exact dynamical model is not specified and the time horizon becomes undefined; verifiability – one can verify a purported controller behavior (e.g., asymptotic stability) in a strict sense; performance – the controller reaches a very close level of average return as a deep NN controller.

In this work, we study the robustness properties of some symbolic controllers derived in [24] as well as deep NN with their a few neuron and symbolic abstractions derived using our methods. By robustness, we mean that a controller maintains its average return values when changing the simulator configuration (scheme/ time-step) at test time while being trained on some specific configuration. Moreover, a robust controller does not admit open sets of simulator solutions with extremely poor return relative to the average. In this regard, we found that NNs are more robust than simple symbolic abstractions, still achieving comparable average return values. To confirm our findings, we implement a workflow of a symbolic controller derivation: regression of a trained deep NN and further fine-tuning. For the simplest benchmark problems, we find that despite the controllers reaching the performance of deep NNs measured in terms of mean return, there exist singular solutions that behave unexpectedly and are persistent for a long time. In some cases, the singular solutions are persistent forever (periodic orbits). The found solutions are stable and an adversary having access to the simulation setup knowing the existence of persistent solutions and POs for specific setups and initial conditions may reconfigure the controlled system and bias it towards the bad persistent solutions; resulting in a significant performance drop, and if the controller is deployed in practice, may even lead to damage of robot/machine. This concern is critical in the context of symbolic controllers, which are simple abstractions more likely to be deployed on hardware than deep NNs. Two systems support the observed issues. First, the standard pendulum benchmark from OpenAI gym [4] and the cartpole swing-up problem.

Each instance of an persistent solution we identify is verified mathematically using computer-assisted proof (CAP) techniques based on interval arithmetic [27, 38] implemented in Julia [3]. Doing so, we verify that the solution truly exists and is not some spurious object resulting from e.g., finite arithmetic precision. Moreover, we prove the adversarial exploitability of a wide class of controllers. The existence of persistent solutions is most visible in the case of symbolic controllers. For deep NN, persistent solutions are less prevalent, and we checked that deep NN controllers' small NN abstractions (involving few neurons) somewhat alleviate the issue of symbolic controllers, strongly suggesting that the robustness is inversely proportional to the number of parameters, starkly contrasting with common beliefs and examples in other domains.

Main Contributions. Let us summarize the main novel contributions of our work to AI community below.

Systematic controller robustness study. In light of the average return metric being sometimes deceptive, we introduce a method for investigating controller robustness by designing an persistent solutions search and the penalty metric.

Identification and proofs of abundant persistent solutions. We sys-

[*] Corresponding Author. Email: jcyranka@gmail.com

tematically find and prove existence of a concerning number of persistent orbits for symbolic controllers in simple benchmark problems. Moreover, we carried out a proof of a periodic orbit for a deep NN controller, which is of independent interest. To our knowledge, this is the first instance of such a proof in the literature.

NN controllers are more robust than symbolic. We find that the symbolic controllers admit significantly more bad persistent solutions than the deep NN and small distilled NN controllers.

1.1 Related Work

(Continuous) RL. A review of RL literature is beyond the scope of this paper (see [34] for an overview). In this work we use state-of-the-art TD3 algorithms dedicated for continuous state/action spaces [12] based on DDPG [25]. Another related algorithm is SAC [16].

Symbolic Controllers. Symbolic regression as a way of obtaining explainable controllers appeared in [22, 20, 24]. Other representations include programs [39, 37] or decision trees [26]. For a broad review of explainable RL see [41].

Falsification of Cyber Physical Systems (CPS) The research on falsification [2, 10, 40, 43] utilizes similar techniques for demonstrating the violation of a temporal logic formula, e.g., for finding solutions that never approach the desired equilibrium. We are interested in solutions that do not reach the equilibrium but also, in particular, the solutions that reach minimal returns.

Verification of NN robustness using SMT Work on SMT like ReLUplex [5, 11, 21] is used to construct interval robustness bounds for NNs only. In our approach we construct interval bounds for solutions of a coupled controller (a NN) with a dynamical system and also provide existence proofs.

Controllers Robustness. Design of NN robust controllers focused on adversarial defence methods [29, 42].

CAPs. Computer-assisted proofs for ordinary differential equations (ODEs) in AI are not common yet. Examples include validation of NN dynamics [23] and proofs of spurious local minima [32].

1.2 Structure of the Paper

Section 2 provides background on numerical schemes and RL framework used in this paper. Section 3 describes the training workflow for the neural network and symbolic controllers. The class of problems we consider is presented in Section 4. We describe the computer-assisted proof methodology in Section 5. Results on persistent periodic orbits appear in Section 6, and we describe the process by which we search for these and related singular solutions in Section 7.

2 Preliminaries

2.1 Continuous Dynamics Simulators for AI

Usually, there is an underlying continuous dynamical system with control input that models the studied problem $s'(t) = f(s(t), a(t))$, where $s(t)$ is the state, $a(t)$ is the control input at time t, and f is a vector field. For instance, the rigid body general equations of motion in continuous time implemented in robotic simulators like MuJoCo [36] are $Mv' + c = \tau + J^T f$, J, f is the constraint Jacobian and force, τ is the applied force, M inertia matrix and c the bias forces. For training RL algorithms, episodes of simulated rollouts $(s_0, a_0, r_1, s_1, \dots)$ are generated; the continuous dynamical system needs to be discretized using one of the available numerical schemes like the Euler or Runge-Kutta schemes [17]. After generating a state rollout, rewards are computed $r_{k+1} = r(s_k, a_k)$. The numerical schemes are characterized by the approximation order, time-step, and explicit/implicit update. In this work, we consider the explicit Euler (E) scheme $s_{k+1} = s_k + hf(s_k, a_k)$; this is a first-order scheme with the quality of approximation being proportional to time-step h (a hyperparameter). Another related scheme is the so-called semi-implicit Euler (SI) scheme, a two-step scheme in which the velocities are updated first. Then the positions are updated using the computed velocities. Refer to the appendix for the exact form of the schemes.

In the research on AI for control, the numerical scheme and time-resolution[1] of observations h are usually fixed while simulating episodes. Assume we are given a controller that was trained on simulated data generated by a particular scheme and h; we are interested in studying the controller robustness and properties after the zero-shot transfer to a simulator utilizing a different scheme or h, e.g., explicit to semi-implicit or using smaller h's.

2.2 Reinforcement Learning Framework

Following the standard setting used in RL, we work with a Markov decision process (MDP) formalism $(\mathcal{S}, \mathcal{A}, F, r, \rho_0, \gamma)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $F\colon \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a deterministic transition function, $r\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, ρ_0 is an initial state distribution, and $\gamma \in (0, 1)$ is a discount factor used in training. $\mathcal{S}$ may be equipped with an equivalence relation, e.g. for an angle variable θ, we have $\theta \equiv \theta + k2\pi$ for all $k \in \mathbb{Z}$. In RL, the agent (policy) interacts with the environment in discrete steps by selecting an action a_t for the state s_t at time t, causing the state transition $s_{t+1} = F(s_t, a_t)$; as a result, the agent collects a scalar reward $r_{t+1}(s_t, a_t)$, the (undiscounted) return is defined as the sum of discounted future reward $R_t = \sum_{i=t}^{T} r(s_i, a_i)$ with $T > 0$ being the fixed episode length of the environment. RL aims to learn a policy that maximizes the expected return over the starting state distribution.

In this work, we consider the family of MDPs in which the transition function is a particular numerical scheme. We study robustness w.r.t. the scheme; to distinguish the *transition function used for training (also called native)* from the *transition function used for testing*, we introduce the notation F_{train} and F_{test} resp. e.g. explicit Euler with time-step h is denoted $F_*(\mathrm{E}, h)$, where $* \in \{test, train\}$.

3 Algorithm for Training of Symbolic Controllers and Small NNs

Carrying out the robustness study of symbolic and small NN controllers requires that the controllers are first constructed (trained). We designed a three-step deep learning algorithm for constructing symbolic and small NN controllers. Inspired by the preceding work in this area the controllers are derived from a deep RL NN controller. The overall algorithm is summarized in Alg. 1.

3.1 RL Training

First we train a deep NN controller using the state-of-the-art model-free RL algorithm TD3 [25, 12] – the SB3 implementation [30]. We choose TD3, as it utilizes the replay buffer and constructs deterministic policies (NN). Plots with the evaluation along the training procedure for studied systems can be found in Appendix C of the extended version of the paper [8].

[1] While in general time-resolution may not be equal to the time step, in this work we set them to be equal.

Algorithm 1 Symbolic/Small NN Controllers Construction

```
input MDP determining studied problem; RL training $h$-params; symbolic & small NN regression $h$-params; fine-tuner $h$-params;
output deep NN policy $\pi_{deep}$; small NN policy $\pi_{small}$; family of symbolic policies $\{\pi_{symb,k}\}$ ($k$ complexity);
1: Apply an off-policy RL algorithm for constructing a deterministic deep NN policy $\pi_{deep}$;
2: Using the replay buffer data apply symbolic regression for computing symbolic abstractions $\{\pi_{symb,k}\}$ (having complexity $k$) of deep NN controller and MSE regression for small NN $\pi_{small}$ policy distillation;
3: Fine-tune the constructed controllers parameters for maximizing the average return using CMA-ES and/or analytic gradient.
```

3.2 Symbolic Regression

A random sample of states is selected from the TD3 training replay buffer. Symbolic abstractions of the deep NN deterministic policies are constructed using the symbolic regression over the replay buffer samples. Following earlier work [22, 20, 24] the search is performed by an evolutionary algorithm. For such purpose, we employ the PySR Python library [6, 7]. The main hyperparameter of this step is the complexity limit (number of unary/binary operators) of the formulas (k in Alg. 1). This procedure outputs a collection of symbolic representations with varying complexity. Another important hyperparameter is the list of operators used to define the basis for the formulas. We use only the basic algebraic operators (add, mul., div, and multip. by scalar). We also tried a search involving nonlinear functions like $tanh$, but the returns were comparable with a larger complexity.

3.3 Distilling Simple Neural Nets

Using a random sample of states from the TD3 training replay buffer we find the parameters of the small NN representation using the mean-squared error (MSE) regression.

3.4 Controller Parameter Fine-tuning

Just regression over the replay buffer is insufficient to construct controllers that achieve expected returns comparable with deep NN controllers, as noted in previous works. The regressed symbolic controllers should be subject to further parameter fine-tuning to maximize the rewards. There exist various strategies for fine-tuning. In this work, we use the non-gradient stochastic optimization covariance matrix adaptation evolution strategy (CMA-ES) algorithm [19, 18]. We also implemented analytic gradient optimization, which takes advantage of the simple environment implementation, and performs parameter optimization directly using gradient descent on the model rollouts from the differentiable environment time-stepping implementation in PyTorch.

4 Studied Problems

We perform our experimental investigation and CAP support in the setting of two control problems belonging to the set of standard benchmarks for continuous optimization. First, the pendulum problem is part of the most commonly used benchmark suite for RL – OpenAI gym [4]. Second, the cart pole swing-up problem is part of the DeepMind control suite [35]. Following the earlier work [13] we used a closed-form implementation of the cart pole swing-up problem. While these problems are of relatively modest dimension, compared to problems in the MuJoCo suite, we find them most suitable to convey our message. The low system dimension makes a self-contained cross-platform implementation easier and eventually provides certificates for our claims using interval arithmetic and CAPs.

4.1 Pendulum

The pendulum dynamics is described by a 1d 2^{nd} order nonlinear ODE. We followed the implementation in OpenAI gym, where the ODEs are discretized with a semi-implicit (SI) Euler method with $h = 0.05$. For training we use $F_{train}(\mathrm{SI}, 0.05)$. Velocity ω is clipped to the range $[-8, 8]$, and control input a to $[-2, 2]$. There are several constants: gravity, pendulum length and mass (g, l, m), which we set to defaults. See Appendix A ([8]) for the details. The goal of the control is therefore to stabilize the up position $\theta = 0 \mod 2\pi$, with zero angular velocity ω. The problem uses quadratic reward for training and evaluation $r = -\lfloor\theta\rfloor^2 - 0.1\omega^2 - 0.001a^2$, where $\lfloor\theta\rfloor = \arccos(\cos(\theta))$ at given time t and action a. The episode length is 200 steps. The max reward is 0, and large negative rewards might indicate long-term simulated dynamics that are not controlled.

4.2 Cartpole Swing-up

The cartpole dynamics is described by a 2d 2^{nd} order nonlinear ODEs with two variables: movement of the cart along a line (x, x'), and a pole attached to the cart (θ, θ'). We followed the implementation given in [15]. The ODEs are discretized by the explicit Euler (E) scheme with $h = 0.01$. As with the pendulum we use clipping on some system states, and several constants are involved, which we set to defaults. See Appendix B ([8]) for details. The goal of the control is to stabilize the pole upwards $\theta = 0 \mod 2\pi$ while keeping the cart x within fixed boundaries. The problem uses a simple formula for reward $r = \cos\theta$, plus the episode termination condition if $|x|$ is above threshold. The episode length is set to 500, hence the reward is within $[-500, 500]$. Large negative reward is usually indicative of undesirable behaviour, with the pole continuously oscillating, the cart constantly moving, and escaping the boundaries fairly quickly.

5 Rigorous Proof Methodology

All of our theorems presented in the sequel are supported by a computer-assisted proof, guaranteeing that they are fully rigorous in a mathematical sense. Based on the existing body of results and our algorithm we developed in Julia, we can carry out the proofs for different abstractions and problems as long as the set of points of non-differentiability is small, e.g., it works for almost all practical applications: ReLU nets, decision trees, and all sorts of problems involving dynamical systems in a closed form. The input to our persistent solutions prover is a function in Julia defining the controlled problem, the only requirement being that the function can be automatically differentiated. To constitute a proof, this part needs to be carried out rigorously with interval arithmetic. Our CAPs are automatic; once our searcher finds a candidate for a persistent solution/PO, a CAP program attempts to verify the existence of the solution/PO by verifying the theorem (Theorem 1) assumptions. If the prover succeeds this concludes the proof.

5.1 Interval Arithmetic

Interval arithmetic is a method of tracking rounding error in numerical computation. Operations on floating point numbers are instead done on *intervals* whose boundaries are floating point numbers. Functions f of real numbers are *extended* to functions $\overline{f}$ defined on intervals, with the property that $\overline{f}(X)$ necessarily contains $\{f(x) : x \in X\}$. The result is that if y is a real number and Y is a thin interval containing y, then $f(y) \in \overline{f}(Y)$. For background, the reader may consult the books [27, 38]. Function iteration on intervals leads to the *wrapping effect*, where the radius of an interval increases along with composition depth. See Figure 1 for a visual.

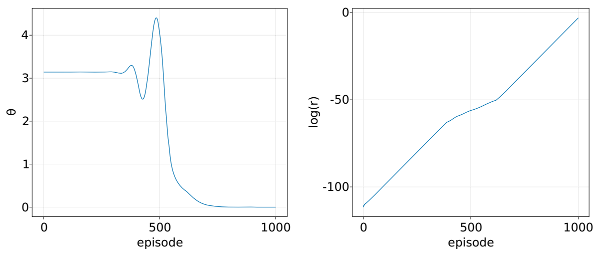


Figure 1: Left: midpoint of interval enclosure of a proven persistent solution (see Appendix Tab. 23 [8]). Right: log-scale of radius of the interval enclosure. Calculations done at 163 bit precision, the minimum possible for this solution at episode length 1000.

5.2 *Computer-assisted Proofs of Periodic Orbits*

For $x = (x_1, \ldots, x_n)$, let $||x|| = \max\{|x_1|, \ldots, |x_n|\}$. The following is the core of our CAPs.

Theorem 1 *Let $G : U \to \mathbb{R}^n$ be continuously differentiable, for U an open subset of $\mathbb{R}^n$. Let $\overline{x} \in \mathbb{R}^n$ and $r^* \geq 0$. Let A be a $n \times n$ matrix[2] of full rank. Suppose there exist real numbers Y, Z_0 and Z_2 such that*

$$||AG(\overline{x})|| \leq Y, \tag{1}$$

$$||I - ADG(\overline{x})|| \leq Z_0 \tag{2}$$

$$\sup_{|\delta| \leq r^*} ||A(DG(\overline{x} + \delta) - DG(\overline{x}))|| \leq Z_2, \tag{3}$$

where $DG(x)$ denotes the Jacobian of G at x, and the norm on matrices is the induced matrix norm. If $Z_0 + Z_2 < 1$ and $Y/(1 - Z_0 - Z_2) \leq r_*$, the map G has a unique zero x satisfying $||x - \overline{x}|| \leq r$ for any $r \in (Y/(1 - Z_0 - Z_2), r_*]$.

A proof can be completed by following Thm 2.1 in [9]. In Sec. 5.3, we identify G whose zeroes correspond to POs. Conditions (1)–(3) imply that the Newton-like operator $T(x) = x - AG(x)$ is a contraction on the closed ball centered at the *approximate zero* $\overline{x}$ with radius $r > 0$. Being a contraction, it has a unique fixed point (x such that $x = T(x)$) by the Banach fixed point theorem. As A is full rank, $G(x) = 0$, hence an orbit exists. The radius r measures how close the approximate orbit $\overline{x}$ is to the exact orbit, x. The contraction is rigorously verified by performing all necessary numerical computations using interval arithmetic. The technical details appear in Appendix D ([8]).

5.3 *Set-up of the Nonlinear Map*

A PO is a finite MDP trajectory. Let the step size be h, and let the period of the orbit be m. We present a nonlinear map that encodes (as zeroes of the map) POs when h is fixed. However, for technical reasons (see Appendix E [8]), it is possible for such a proof to fail. If Alg. 2 fails to prove the existence of an orbit with a fixed step size h, we fall back to a formulation where the step size is not fixed, which is more likely to yield a successful proof. This alternative encoding map G_2 is presented in Appendix D ([8]). Given h, pick $g(h, \cdot) \in \{g_{\mathrm{E}}, g_{\mathrm{SI}}\}$ one of the discrete dynamical systems used for numerically integrating the ODE. Let p be the dimension of the state space, so $g(h, \cdot) : \mathbb{R}^p \to \mathbb{R}^p$. We interpret the first dimension of $\mathbb{R}^p$ to be the angular component, so that a periodic orbit requires a shift by a multiple of 2π in this variable. Given h, the number of steps m (i.e. period of the orbit) and the number of signed rotations j in the angular variable, POs are zeroes of the map (if and only if) $G_1 : \mathbb{R}^{pm} \to \mathbb{R}^{pm}$, defined by

$$G_1(X) = \begin{pmatrix} x_0 - g(h, x_m) + (j2\pi, \mathbf{0}) \\ x_1 - g(h, x_0) \\ x_2 - g(h, x_1) \\ \vdots \\ x_m - g(h, x_{m-1}) \end{pmatrix},$$

where $\mathbf{0}$ is the zero vector in $\mathbb{R}^{p-1}$, $X = (x_1, \ldots, x_m)$ for $x_i \in \mathbb{R}^p$, and $x_1, \ldots, x_m$ are the time-ordered states.

6 Persistent Orbits in Controlled Pendulum

When constructing controllers using machine learning or statistical methods, the most often used criterion for measuring their quality is the mean return from performing many test episodes. The mean return may be a deceptive metric for constructing robust controllers. More strongly, our findings suggest that mean return is not correlated to the presence of periodic orbits or robustness. One would typically expect a policy with high mean return to promote convergence toward states that maximize the return for any initial condition (IC) and also for other numerical schemes. Our experiments revealed reasons to believe this may be true for deep NN controllers. However, in the case of simple symbolic controllers, singular persistent solutions exist that accumulate large negative returns at a fast pace. By persistent solutions we mean periodic orbits that remain ε away from the desired equilibrium. This notion we formalize in Sec. 7.1. We emphasize that all of the periodic orbits that we prove are necessary stable in the usual Lyapunov sense, i.e., the solutions that start out near an equilibrium stay near the equilibrium forever, and hence feasible in numerical simulations. We find such solutions for controllers as provided in the literature and constructed by ourselves employing Alg. 1. We emphasize that our findings are not only numerical, but we support them with (computer-assisted) mathematical proofs of existence.

6.1 *Landajuela et. al [24] Controller*

First, we consider the symbolic low complexity controller for the pendulum $a = -7.08s_2 - (13.39s_2 + 3.12s_3)/s_1 + 0.27$, derived in [24] (with model given in Appendix A [8]), where $s_1 = \cos\theta$, $s_2 = \sin\theta$, $s_3 = \omega = \theta'$, and a is the control input. While this controller looks more desirable than a deep NN with hundreds thousand of parameters, its performance changes dramatically when using slightly different transition function at test-time, i.e., halved h ($F_{test}(\mathrm{SI}, 0.025)$) or the explicit Euler scheme ($F_{test}(\mathrm{E}, 0.05)$). Trajectories in Fig. 2 illustrate that some orbits oscillate instead of stabilizing at the equilibrium $\hat{s} = \hat{\theta} = 0 \bmod 2\pi$. The average return significantly deteriorates for the modified schemes and the same ICs

[2] In practice, a numerical approximation $A \approx DF(\overline{x})^{-1}$.

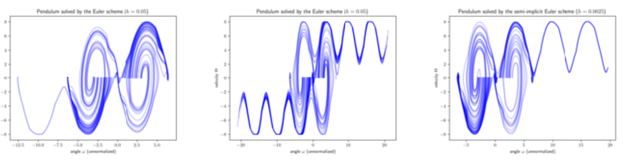
compared to $F_{train}(\text{SI}, 0.05)$; see Tab. 1. Such issues are present in deep NN controllers and small distilled NN to a significantly lower extent. We associate the cause of the return deterioration with existence of 'bad' solutions – persistent periodic orbits (POs) (formal Def. 1). Using CAPs (c.f., Sec. 5) we obtain:

Theorem 2 *For $h \in H = \{0.01, 0.005, 0.0025, 0.001\}$, the nonlinear pendulum system with controller a from [24] described in the opening paragraph of Section 6.1 has a periodic orbit (PO) under the following numerical schemes;*

1) (SI) *with step size $h \in H$,*

2) (E) *at $h = 0.05$ (native), and for all $h \in H$.*

The identified periodic orbits are persistent (see Def. 2) and generate minus infinity return for infinite episode length, with each episode decreasing the reward by at least 0.198.



(a) (SI), $h = 0.05$ (native) (b) (E), $h = 0.05$ (c) (SI), $h = 0.025$

Figure 2: 100 numerical simulations with IC $\omega = 0$ and θ sampled uniformly, time horizon set to $T = 6$, x-axis shows the (unnormalized) ω, and y-axis θ. In (a), all IC are attracted by an equilibrium at $\omega = 0 \bmod 2\pi$, $\theta = 0$. Whereas when applying different F_{test}, (b) and (c) show existence of attracting periodic solutions (they can be continued infinitely as our theorems demonstrate).

6.2 *Our Controllers*

The issues with robustness and performance of controllers of Sec. 6.1 may be an artefact of a particular controller construction rather than a general property. Indeed, that controller had a division by s_1. To investigate this further we apply Alg. 1 for constructing symbolic controllers of various complexities (without divisions). Using Alg. 1 we distill a small NN (single hidden layer with 10 neurons) for comparison. In step 2 we use fine-tuning based on either analytic gradient or CMA-ES, each leading to different controllers. The studied controllers were trained using the default transition $F_{train}(\text{SI}, 0.05)$, and for testing using $F_{test}(\text{E}, 0.05)$, $F_{test}(\text{E}, 0.025)$, $F_{test}(\text{SI}, 0.05)$, $F_{test}(\text{SI}, 0.025)$.

Tab. 1 reveals that the average returns deteriorate when using other numerical schemes for the symbolic controllers obtained using Alg. 1, analogous to the controller from [24]. The average return discrepancies are very large as well. We emphasize that all of the studied metrics for the symbolic controllers are far from the metrics achieved for the deep NN controller. Terminating Alg. 1 at step 2 results in a very bad controller achieving mean return only of -1061, i.e., as observed in the previous works the symbolic regression over a dataset sampled from a trained NN is not enough to construct a good controller. Analogous to Theorem 2, we are able to prove the following theorems on persistent periodic orbits (Def. 1) for the controllers displayed in Table 1.

Theorem 3 *For $h \in H = \{0.025, 0.0125\}$, the nonlinear pendulum system with controller generated by analytic gradient refinement in Tab. 1 has POs under*

1) (SI) *with $h \in H$ and at the native step size $h = 0.05$,*

2) (E) *with $h \in H$.*

The identified periodic orbits are persistent (see Def. 2) and generate minus infinity return for infinite episode length, with each episode decreasing the reward by at least 0.18.

Theorem 4 *For $h = 0.0125$ and $h = 0.05$ (native) with scheme* (E)*, the nonlinear pendulum system with controller generated by CMA-ES refinement in Tab. 1 has POs which generate minus infinity return for infinite episode length, with each episode decreasing the reward by at least 0.20.*

7 Systematic Robustness Study

We consider a controller to be *robust* when it has "good" return statistics at the native simulator and step size, which persist when we change simulator and/or decrease step size. If a degradation of return statistics on varying the integrator or step size is identified, we wish to identify the source.

7.1 *Background on Persistent Solutions and Orbits*

Consider a MDP tuple $(\mathcal{S}, \mathcal{A}, F, r, \rho_0, \gamma)$, a precision parameter $\varepsilon > 0$, a policy $\pi\colon \mathcal{S} \to \mathcal{A}$ (trained using F_{train} and tested using F_{test}), a desired equilibrium $\hat{s}$ (corresponding to the maximized reward r), and episode length N.

Definition 1 *We call a* persistent periodic orbit (PO) *(of period n) an infinite MDP trajectory $(s_0, a_0, r_1, s_1, a_1, \dots)$, such that $s_{kn} = s_0$ for some $n > 1$ and all $k \in \mathbb{N}$, and such that $\|\hat{s} - s_j\| > \varepsilon$ for all $j \geq 0$.*

Definition 2 *A finite MDP trajectory of episode length N $(s_0, a_0, p_1, s_1, a_1, \dots, s_N)$ such that $\|\hat{s} - s_j\| > \varepsilon$ for all $0 \leq j \leq N$ is called a* persistent solution.

Locating the objects in dynamics responsible for degradation of the reward is not an easy task, as they may be singular or local minima of a non-convex landscape. For locating such objects we experimented with different strategies, but found the most suitable the evolutionary search of *penalty maximizing solutions*. The solutions identified using such a procedure are necessarily stable. We introduce a measure of 'badness' of persistent solutions and use it as a search criteria.

Definition 3 *We call* a penalty value*, a function $p\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$, such that for a persistent solution/orbit the accumulated penalty value is bounded from below by a set threshold $M \gg 0$, that is $\sum_{i=0}^{N-1} p(s_i, a_i) \geq M$.*

Remark 4 *The choice of particular penalty in Def. 3 depend on the particular studied example. We choose the following penalties in the studied problems.*

1. $p(s, a) = -r(s, a)$ for pendulum.

2. $p(s, a) = -r(s) + 0.5(\theta')^2 + 0.5(x')^2$ for cartpole swing-up. Subtracting from the native reward value $r(s) = \cos\theta$ the scaled sum of squared velocities (the cart and pole) and turning off the episode termination condition. This allows capturing orbits that manage to stabilize the pole, but are unstable and keep the cart moving. The threshold M in Def. 3 can be set by propagating a number of trajectories with random IC and taking the maximal penalty as M.

Remark 5 *For a PO, the accumulated penalty admits a linear lower bound, i.e. $\sum_{m=0}^{n-1} p(s_m, a_m) \geq Cn$ for some $C > 0$. Thm. 2 implies $C = 0.14$ for the POs in Tab. 6 in the Appendix [8].*

Table 1: Comparison of different controllers for the pendulum. Mean $\pm$ std.dev. rounded to decimal digit, returns over 100 episodes reported for different F_{test} (larger the better). $F_{test} = F_{train}$ marked in bold. In this case mean return is equal to negative accumulated penalty. Absolute return discrepancies measure discrepancy in episodic return between different schemes (E/SI) for the same IC (smaller the better). The meaning of observation vector at given time t, $x_0 = \cos\theta(t)$, $x_1 = \sin\theta(t)$, $x_2 = \omega(t) = \theta(t)'$.

		MEAN RETURN FOR GIVEN F_{test}				
		$\mathbf{h = 0.05}$		$h = 0.025$		DISCREPANCY
ORIGIN	FORMULA	**SI**	E	SI	E	RETURN E / SI
ALG. 1, 3.ANALYTIC (SYMB. $k = 9$)	$((((1.30 \cdot x_2 + 4.18 \cdot x_1)x_0) + 0.36x_1)/ - 0.52)$	-207 ± 183	-604 ± 490	-431 ± 396	-910 ± 853	479 ± 416
ALG. 1, 3.CMA-ES (SYMB. $k = 9$)	$((((-10.59x_2 + -42.47x_1)x_0) + 1.2x_1)/5.06)$	-165 ± 113	-659 ± 461	-331 ± 225	-1020 ± 801	538 ± 401
ALG. 1, SMALL NN	10 NEURONS DISTILLED SMALL NN	-157 ± 99	-304 ± 308	-311 ± 196	-290 ± 169	188 ± 285
[24] (a_1)	$-7.08x_1 - (13.39x_1 + 3.12x_2)/x_0 + 0.27$	-150 ± 87	-703 ± 445	-318 ± 190	-994 ± 777	577 ± 401
TD3 TRAINING	DEEP NN	-149 ± 86	-138 ± 77	-298 ± 171	-278 ± 156	18 ± 38

7.2 Searching for and Proving Persistent Orbits

We designed a pipeline for automated persistent/periodic orbits search together with interval proof certificates. By an interval proof certificate of a PO we mean interval bounds within which a CAP that the orbit exist was carried out applying the Newton scheme (see Sec. 5.2), whereas by a proof certificate of a persistent solution (which may be a PO or not) we mean interval bounds for the solution at each step, with a bound for the reward value, showing that it does not stabilize by verifying a lower bound $\|\hat{s} - s_t\| > \varepsilon$. The search procedure is implemented in Python, while the CAP part is in Julia, refer Sec. 5 for further details.

Algorithm 2 Persistent Solutions/Orbits Search & Prove

```
input $F_{test}$; control policy $\pi$; $h$-parameters of the evolutionary
    search; penalty function $p$; trajectory length; search domain;
output interval certificates of persistent/periodic orbits;
 1: for each MDP do
 2:     for number of searches do
 3:         initialize CMA-ES search within specified bounds;
 4:         search for a candidate maximizing penalty $p$ during the
            fixed episode length;
 5:     end for
 6:     order found candidates w.r.t. their $p$ value;
 7: end for
 8: for each candidate do
 9:     search for nearby periodic orbit with Newton's method cor-
        rection applied to suitable sub-trajectory;
10:     if potential periodic orbit found then
11:         attempt to prove existence of the orbit with Thm. 1;
12:         if proof successful then
13:             return an interval certificate of the orbit;
14:         else
15:             return proof failure;
16:         end if
17:     else
18:         return periodic orbit not found;
19:     end if
20:     produce and return an interval certificate of the uncontrolled
        solution;
21: end for
```

7.3 Findings: Pendulum

Changing simulator or step size resulted in substantial mean return loss (see Tab. 1), and simulation revealed stable POs (see Fig. 2). We proved existence of POs using the methods of Section 5.2–5.3. Proven POs are presented in tables in Appendix F ([8]). See also Fig. 3, where an persistent solution shadows an unstable PO before converging to the stable equilibrium. We present proven persistent solutions in the tables in Appendix F ([8]).

Comparing the mean returns in Tab. 1 we immediately see that deep NN controller performance does not deteriorate as much as for the symbolic controller, whereas the small net is located between the two extremes. This observation is confirmed after we run Alg. 2 for the symbolic controllers and NN. In particular, we did not identify any stable periodic orbits or especially long persistent solutions. However, the Deep NN controller is not entirely robust, admitting singular persistent solutions achieving returns far from the mean; refer to Tab. 4. On the other hand, the small 10 neuron NN also seems to be considerably more robust than the symbolic controllers. For the case $F_{test}(\mathrm{E}, 0.05)$ the average returns are two times larger than for the symbolic controllers, but still two times smaller than for the deep NN. However, in the case $F_{test}(\mathrm{E}, 0.05)$, the average returns are close to those of the deep NN contrary to the symbolic controllers. The small NN compares favorably to symbolic controllers in terms of E/SI return discrepancy metrics, still not reaching the level of deep NN. This supports our earlier conjecture (Sec. 1) that controller robustness is proportional to the parametric complexity.

Table 2: Examples of persistent solutions found by the persistent solutions Search & Prove Alg. 2 for the pendulum maximizing accumulated penalty, episodes of fixed length $N = 1000$. The found persistent solutions were the basis for the persistent orbit/solution proofs presented in Appendix F ([8])

CONTROLLER	MDP	$\sum r(s, a)$
ALG. 1 ($k = 9$)	(SI) $h = 0.05$	-9869.6
ALG. 1 ($k = 9$)	(SI) $h = 0.025$	-1995.7
ALG. 1 SMALL NN	(SI) $h = 0.05$	-926.8
ALG. 1 SMALL NN	(SI) $h = 0.025$	-1578.4
ALG. 1 SMALL NN	(E) $h = 0.05$	-747.3
[24] (a_1)	(SI) $h = 0.05$	-873.8
[24] (a_1)	(SI) $h = 0.025$	-1667.6
[24] (a_1)	(E) $h = 0.05$	-5391.1
DEEP NN	(SI) $h = 0.05$	-426.4
DEEP NN	(SI) $h = 0.025$	-818.6
DEEP NN	(E) $h = 0.05$	-401.4

7.4 Findings: Cartpole Swing-Up

We computed the mean return metrics for a representative symbolic controller, a distilled small NN controller and the deep NN, see Tab. 3. For the symbolic controller, the average return deteriorates more when changing the simulator's numerical scheme to other than the native ($F_{train}(\mathrm{E}, 0.01)$). Notably, the E/SI discrepancy is an order of magnitude larger than in the case of deep NN. As for the pen-

Table 3: Mean ± std.dev. reported, rounded to single decimal digits, of returns over 100 episodes reported for different F_{test} (larger the better). $F_{test} = F_{train}$ marked in bold. Return discrepancies measure discrepancy in episodic return between different schemes (E/SI) for the same IC (smaller the better). The formula for the symbolic controller with $k = 21$ appears in Appendix Tab. 27 [8]

	MEAN RETURN FOR GIVEN F_{test}				
	$\mathbf{h = 0.01}$		$h = 0.005$		DISCREPANCY
ORIGIN	SI	**E**	SI	E	RETURN E / SI
ALG. 1, 3.CMA-ES (SYMB. $k = 21$)	220.2 ± 96.7	334.3 ± 37	474.6 ± 194.3	632.2 ± 119.3	121.9 ± 88.9
ALG. 1, SMALL NN (25 NEURONS)	273.3 ± 128.7	332.9 ± 79.2	585.1 ± 229.1	683.7 ± 103.3	86.6 ± 135.1
TD3 TRAINING	381.2 ± 9.1	382.9 ± 9	760.9 ± 18.4	764.0 ± 18.1	1.7 ± 0.9

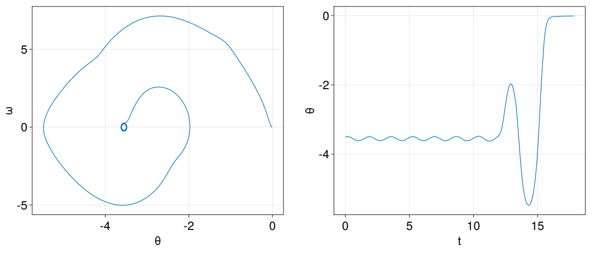


Figure 3: A persistent solution with poor reward ≈ -7527 over episode length 1000 with step size $h = 0.0125$, plotted until near-stabilization at $t = 17.825$. Left: plot in phase space. Right: time series of θ. Other data for this solution is in Appendix Tab. 22 [8]

dulum, the small NN sits between the symbolic and deep NN in terms of the studied metrics. We computed the mean accumulated shaped penalty $p(s,a) = -r(s) + 0.5(\theta')^2 + 0.5(x')^2$ for the selected controllers in Tab. 5. The contrast between the deep NN and the symbolic controller is clear, with the small NN being in between those two extremes. The mean penalty is a measure of the prevalence of persistent solutions. However, we emphasize that the Deep NN controller is not entirely robust and also admits singular persistent solutions with bad returns, refer to Tab. 4. Rigorously proving the returns for the deep NN was not possible in this case; see Rem. 6.

Investigating the persistent solutions found with Alg. 2 in Fig. 4 we see that in case $F_{test}(\mathrm{SI}, 0.01)$ the symbolic controller admits bad persistent solutions with x_t decreasing super-linearly, whereas θ stabilizes at $\theta \sim 0.01$. In contrast, the deep NN exhibits fairly stable control with small magnitude oscillations. This example emphasizes the shaped penalty's usefulness in detecting such bad persistent solutions. We can see several orders of magnitude differences in the accumulated penalty value for the deep NN controller vs. the symbolic controller case. We identify and rigorously prove an abundance of persistent solutions for each of the studied symbolic controllers. For example, we can prove:

Theorem 5 *For the symbolic controller with complexity $k = 21$ and native step size $h = 0.01$, there are 2000-epsiode persistent solutions of the cartpole swing-up model with accumulated penalty $\geq 2.66 \times 10^5$ for the explicit scheme, and $\geq 3.77 \times 10^5$ for the semi-implicit scheme. With the Small NN controller, the conclusions hold with accumulated penalties ≥ 6263 and $\geq 2.68 \times 10^6$.*

We demonstrate persistent solutions for each considered controller in Tab. 4. The found persistent solutions were the basis for the persistent orbit/solution proofs presented in Appendix G ([8]). The symbolic and small NN controllers admit much worse solutions with increasing velocity, as illustrated in Fig. 4b. Deep NN controllers admit such bad solutions when tested using smaller time steps $((E, 0.005), (SI, 0.005))$; see examples in Tab. 4. They also exhibit persistent periodic solutions, albeit with a small ϵ; see Fig. 4a. We have proven the following.

Table 4: Examples of persistent solutions found by the transient solutions Search & Prove Alg. 2 for the cartpole-swingup maximizing the accumulated penalty, episodes of fixed length $N = 2000$ without taking into account the termination condition. The found persistent solutions were the basis for the persistent orbit/solution proofs presented in Appendix G ([8])

CONTROLLER	MDP	$\sum r(s,a)$
ALG. 1 ($k = 21$)	(SI) $h = 0.01$	-41447.2
ALG. 1 ($k = 21$)	(SI) $h = 0.005$	-11204.3
ALG. 1 ($k = 21$)	(E) $h = 0.01$	-29878.0
ALG. 1 ($k = 21$)	(E) $h = 0.005$	-8694.3
ALG. 1 SMALL NN	(SI) $h = 0.01$	-2684696.8
ALG. 1 SMALL NN	(SI) $h = 0.005$	-798442.3
ALG. 1 SMALL NN	(E) $h = 0.01$	-520.9
ALG. 1 SMALL NN	(E) $h = 0.005$	-2343.8
DEEP NN	(SI) $h = 0.01$	306.6
DEEP NN	(SI) $h = 0.005$	-396074.9
DEEP NN	(E) $h = 0.01$	226.5
DEEP NN	(E) $h = 0.005$	-1181.7

Theorem 6 *For h close to*[3] 0.005 *and* $h = 0.01$ *(native), the cartpole swing-up model has POs for (E) and (SI) with the deep NN controller. The mean penalties along orbits are greater than* -0.914 *and are persistent*[4] *with* $\epsilon \geq 0.036$.

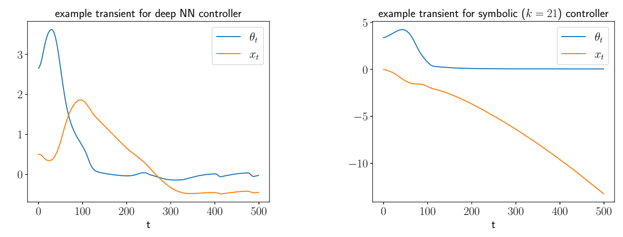
Remark 6 *We were not able to rigorously compute the penalty values of the persistent solutions for the deep NN controller due to wrapping effect of interval arithmetic calculations [38], which is made much worse by the width of the network (400,300) and the long epsiode length (which introduces further composition). However, this is not a problem for the periodic orbits: we enclose them using Theorem 1, which reduces the wrapping effect.*

Table 5: Comparison of different controllers for the cartpole swing-up for $h = 0.01$. Mean and std.dev. (after ±) reported of accumulated penalties $\sum p(s_k) = \sum -r(s_k) + 0.5(\theta_k')^2 + 0.5(x_k')^2$ (larger the worse) over 100 episodes reported for different F_{test}. $F_{test} = F_{train}$ marked in bold. Controllers same as in Tab. 3.

ORIGIN	SI	**E**
ALG. 1, 3.CMA-ES (SYMB. $k = 21$)	3123.0 ± 719.9	2257.2 ± 234.1
ALG. 1, SMALL NN (25 NEURONS)	1413.4 ± 9670.1	404.2 ± 148.4
TD3 TRAINING	335.7 ± 64.7	425.6 ± 72.1

[3] The exact step size is smaller than h, with relative error up to 2%. See Appendix G ([8]) for precise values and detailed data for the POs.

[4] With respect to the translation-invariant seminorm $||(x, \dot{x}, \theta, \dot{\theta})|| = \max\{|\dot{x}|, |\theta|, |\dot{\theta}|\}$



(a) Deep NN controller (b) a symbolic controller

Figure 4: The persistent solutions (evolution of (θ, x) (Def. 2) for cartpole swing-up problem found with Alg. 2 that maximize accumulated penalty $\sum p(s,a) = \sum -r(s) + 0.5(\theta')^2 + 0.5(x')^2$ over episodes of length 2000 without terminations, using SI with $h = 0.01$. (a) $\sum p(s,a) = -306$; (b) $\sum p(s,a) = 37746$.

8 Codebase

Our full codebase is written in Python and Julia shared in a github repository [1]. The reason why the second part of our codebase is written in Julia is the lack of a suitable interval arithmetic library in Python. The Python part of the codebase consists of four independent parts – scripts: deep NN policy training, symbolic/small NN controller regression, regressed controller fine-tuning and periodic orbit/persistent solution searcher. All controllers that we use are implemented in Pytorch [28]. For the deep NN policy training we just use the Stable-baselines 3 library [30], which outputs a trained policy (which achieved the best return during training) and the training replay buffer of data. For the symbolic regression we employ the PySR lib. [6]. For the regressed controller fine-tuning we employ the pycma CMA-ES implementation [18]. Our implementation in Julia uses two external packages: IntervalArithmetic.jl [33] (for interval arithmetic) and ForwardDiff.jl [31] (for forward-mode automatic differentiation). These packages are used together to perform the necessary calculations for the CAPs.

9 Conclusion and Future Work

Our work is a first step towards a comprehensive robustness study of deep NN controllers and their symbolic abstractions, which are desirable for deployment and trustfulness reasons. Studying the controllers' performance in a simple benchmark, we identify and prove existence of an abundance of persistent solutions and periodic orbits. Persistent solutions are undesirable and can be exploited by an adversary. Future work will apply the developed methods to study higher dimensional problems often used as benchmarks for continuous control.

10 Acknowledgements

The project is financed by the Polish National Agency for Academic Exchange. The first author has been supported by the Polish National Agency for Academic Exchange Polish Returns grant no. PPN/PPO/2018/1/00029 and the University of Warsaw IDUB New Ideas grant. This research was supported in part by PL-Grid Infrastructure.

References

[1] Code repository. https://github.com/MIMUW-RL/worrisome-nn. Accessed: 2023-07-27.
[2] Houssam Abbas, Georgios Fainekos, Sriram Sankaranarayanan, Franjo Ivančić, and Aarti Gupta, 'Probabilistic temporal logic falsification of cyber-physical systems', *ACM Trans. Embed. Comput. Syst.*, **12**(2s), (may 2013).
[3] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah, 'Julia: A fresh approach to numerical computing', *SIAM review*, **59**(1), 65–98, (2017).
[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.
[5] Rudy Bunel, Ilker Turkaslan, Philip H.S. Torr, Pushmeet Kohli, and M. Pawan Kumar, 'A unified view of piecewise linear neural network verification', in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, p. 4795–4804, Red Hook, NY, USA, (2018). Curran Associates Inc.
[6] Miles Cranmer. Pysr: Fast & parallelized symbolic regression in python/julia, September 2020.
[7] Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho, 'Discovering symbolic models from deep learning with inductive biases', *NeurIPS 2020*, (2020).
[8] Jacek Cyranka, Kevin E M Church, and Jean-Philippe Lessard, 'Worrisome properties of neural network controllers and their symbolic representation —- extended version', *arXiv preprint arXiv:2307.15456*, (2023). https://arxiv.org/abs/2307.15456.
[9] Sarah Day, Jean-Philippe Lessard, and Konstantin Mischaikow, 'Validated Continuation for Equilibria of PDEs', *SIAM Journal on Numerical Analysis*, **45**(4), 1398–1424, (jan 2007).
[10] Tommaso Dreossi, Alexandre Donzé, and Sanjit A. Seshia, 'Compositional falsification of cyber-physical systems with machine learning components', *J. Autom. Reason.*, **63**(4), 1031–1053, (dec 2019).
[11] Rüdiger Ehlers, 'Formal verification of piece-wise linear feed-forward neural networks', in *Automated Technology for Verification and Analysis*, eds., Deepak D'Souza and K. Narayan Kumar, pp. 269–286, Cham, (2017). Springer International Publishing.
[12] Scott Fujimoto, Herke van Hoof, and David Meger, 'Addressing Function Approximation Error in Actor-Critic Methods', *arXiv e-prints*, arXiv:1802.09477, (February 2018).
[13] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen, 'Improving PILCO with Bayesian neural network dynamics models', in *Data-Efficient Machine Learning workshop, International Conference on Machine Learning*, (2016).
[14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, 'Explaining and harnessing adversarial examples', *arXiv preprint arXiv:1412.6572*, (2014).
[15] David Ha, 'Evolving stable strategies', *blog.otoro.net*, (2017).
[16] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al., 'Soft actor-critic algorithms and applications', *arXiv preprint arXiv:1812.05905*, (2018).
[17] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I (2nd Revised. Ed.): Nonstiff Problems*, Springer-Verlag, Berlin, Heidelberg, 1993.
[18] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019.
[19] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos, 'Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)', *Evolutionary Computation*, **11**(1), 1–18, (2003).
[20] Daniel Hein, Steffen Udluft, and Thomas A. Runkler, 'Interpretable policies for reinforcement learning by genetic programming', *Engineering Applications of Artificial Intelligence*, **76**, 158–169, (2018).
[21] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer, 'Reluplex: An efficient smt solver for verifying deep neural networks', in *Computer Aided Verification*, eds., Rupak Majumdar and Viktor Kunčak, pp. 97–117, Cham, (2017). Springer International Publishing.
[22] Jiří Kubalík, Eduard Alibekov, and Robert Babuška, 'Optimal control via reinforcement learning with symbolic policy approximation', *IFAC-PapersOnLine*, **50**(1), 4162–4167, (2017). 20th IFAC World Congress.

[23] Christian Kuehn and Elena Queirolo. Computer validation of neural network dynamics: A first case study, 2022.

[24] Mikel Landajuela, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol, 'Discovering symbolic policies with deep reinforcement learning', in *Proceedings of the 38th International Conference on Machine Learning*, eds., Marina Meila and Tong Zhang, volume 139 of *Proceedings of Machine Learning Research*, pp. 5979–5989. PMLR, (18–24 Jul 2021).

[25] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra, 'Continuous control with deep reinforcement learning.', in *ICLR*, eds., Yoshua Bengio and Yann LeCun, (2016).

[26] Guiliang Liu, Oliver Schulte, Wang Zhu, and Qingcan Li, 'Toward interpretable deep reinforcement learning with linear model u-trees', in *ECML/PKDD*, (2018).

[27] Ramon E. Moore, *Interval analysis*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1966.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala, 'Pytorch: An imperative style, high-performance deep learning library', in *Advances in Neural Information Processing Systems 32*, 8024–8035, Curran Associates, Inc., (2019).

[29] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta, 'Robust adversarial reinforcement learning', in *Proceedings of the 34th International Conference on Machine Learning*, eds., Doina Precup and Yee Whye Teh, volume 70 of *Proceedings of Machine Learning Research*, pp. 2817–2826. PMLR, (06–11 Aug 2017).

[30] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann, 'Stable-baselines3: Reliable reinforcement learning implementations', *Journal of Machine Learning Research*, **22**(268), 1–8, (2021).

[31] Jarrett Revels, Miles Lubin, and Theodore Papamarkou. Forward-mode automatic differentiation in julia, 2016.

[32] Itay Safran and Ohad Shamir, 'Spurious local minima are common in two-layer ReLU neural networks', in *Proceedings of the 35th International Conference on Machine Learning*, eds., Jennifer Dy and Andreas Krause, volume 80 of *Proceedings of Machine Learning Research*, pp. 4433–4441. PMLR, (10–15 Jul 2018).

[33] David P. Sanders, Luis Benet, Luca Ferranti, Krish Agarwal, Benoît Richard, Josua Grawitter, Eeshan Gupta, Marcelo Forets, Michael F. Herbst, yashrajgupta, Eric Hanson, Braam van Dyk, Christopher Rackauckas, Rushabh Vasani, Sebastian Micluţa-Câmpeanu, Sheehan Olver, Twan Koolen, Caroline Wormell, Daniel Karrasch, Favio André Vázquez, Guillaume Dalle, Jeffrey Sarnoff, Julia TagBot, Kevin O'Bryant, Kristoffer Carlsson, Morten Piibeleht, Mosè Giordano, Ryan, Robin Deits, and Tim Holy. Juliaintervals/intervalarithmetic.jl: v0.20.8, October 2022.

[34] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press, second edn., 2018.

[35] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller, 'DeepMind Control Suite', *arXiv e-prints*, arXiv:1801.00690, (January 2018).

[36] Emanuel Todorov, Tom Erez, and Yuval Tassa, 'Mujoco: A physics engine for model-based control', in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, (2012).

[37] Dweep Trivedi, Jesse Zhang, Shao-Hua Sun, and Joseph J Lim, 'Learning to synthesize programs as interpretable and generalizable policies', in *Advances in Neural Information Processing Systems*, eds., M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, volume 34, pp. 25146–25163. Curran Associates, Inc., (2021).

[38] Warwick Tucker, *Validated Numerics*, Princeton University Press, jul 2011.

[39] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri, 'Programmatically interpretable reinforcement learning', in *Proceedings of the 35th International Conference on Machine Learning*, eds., Jennifer Dy and Andreas Krause, volume 80 of *Proceedings of Machine Learning Research*, pp. 5045–5054. PMLR, (10–15 Jul 2018).

[40] Masaki Waga, 'Falsification of cyber-physical systems with robustness-guided black-box checking', in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, HSCC '20, New York, NY, USA, (2020). Association for Computing Machinery.

[41] Lindsay Wells and Tomasz Bednarz, 'Explainable ai and reinforcement learning—a systematic review of current approaches and trends', *Frontiers in Artificial Intelligence*, **4**, (2021).

[42] Tsui-Wei Weng, Krishnamurthy (Dj) Dvijotham*, Jonathan Uesato*, Kai Xiao*, Sven Gowal*, Robert Stanforth*, and Pushmeet Kohli, 'Toward evaluating robustness of deep reinforcement learning with continuous control', in *International Conference on Learning Representations*, (2020).

[43] Yoriyuki Yamagata, Shuang Liu, Takumi Akazaki, Yihai Duan, and Jianye Hao, 'Falsification of cyber-physical systems using deep reinforcement learning', *IEEE Transactions on Software Engineering*, **47**(12), 2823–2840, (2021).

[44] Hanshu YAN, Jiawei DU, Vincent TAN, and Jiashi FENG, 'On robustness of neural ordinary differential equations', in *International Conference on Learning Representations*, (2020).