Certificate-Guided Robust Training for Deep Reinforcement Learning via State Abstraction

Qixuan Cao
Shanghai Key Laboratory of Trustworthy Computing
East China Normal University
Shanghai, China
51265902038@stu.ecnu.edu.cn

Dapeng Zhi
School of Computer Engineering
Jiangsu University of Technology
Changzhou, China
zhi.dapeng@163.com

Min Zhang*
Shanghai Key Laboratory of Trustworthy Computing
East China Normal University
Shanghai, China
zhangmin@sei.ecnu.edu.cn

***Abstract*—Deep reinforcement learning (DRL) systems have achieved unprecedented results in recent years. However, even the best DRL control policies are often vulnerable to state perturbations, which may cause these systems to act unexpectedly, thereby prohibiting their widespread deployment. Multiple methods have therefore been put forth to robustify DRL systems during their training. Albeit promising, these approaches suffer from two main setbacks: (i) they tend to be extremely expensive and (ii) they are typically tailored to specific DRL algorithms. To bridge these gaps, we propose a novel approach to improve the robustness of DRL systems. Our approach employs recent advances in abstraction-based training and exploits the induced finite abstract state space to efficiently compute robustness certificates, which are incorporated into a certificate-guided robust objective. The resulting abstract DRL system can then be easily trained using a novel loss function defined by robustness certificates while still maintaining high performance. Our approach is unified in that it fully applies to any DRL training algorithm. We demonstrate our approach on a wide range of benchmarks. It significantly outperforms the existing methods, improving the accumulated rewards by 92.0% on average under various perturbations and attacks.**

***Index Terms*—robust deep reinforcement learning, state abstraction, abstract neural networks, robustness certification**

I. Introduction

Recent years have witnessed the superior performance of DRL systems in various domains, from self-driving [1], [2] to robot control [3]–[5], and manufacturing [6]–[8]. However, in many safety-critical areas (e.g., self-driving), DRL systems still cannot be fully deployed due to concerns about their reliability and robustness. Most of the widely-used DRL policies are often vulnerable to various perturbations [9].

To address these concerns, a plethora of methods have been put forth recently, attempting to robustify DRL systems using robust training [10]–[13]. However, these training methods are not general but tailored for specific DRL algorithms. For instance, CARRL [14] computes lower bounds on state-action values and selects actions under worst-case conditions. It is only compatible with Deep Q-Network policies. Another line of work considers adversarial training and retraining systems with random perturbations and adversarial attacks [15]–[18]. These approaches train systems and adversarial attackers together or train systems using calculated redundant adversarial samples, increasing the computational and sampling complexity during training.

Recently, training data abstraction has demonstrated its generality in easy-to-verify DRL system training [19], efficient sampling in neural episodic control [20] and robust image classifiers [21]. From Martin's perspective, symbolic abstraction of training data can be a promising direction for safe and secure machine learning [22]. The central idea underlying these approaches is to abstract the training data and then to train neural network models on abstract data. However, the use of training-data abstraction for robust deep reinforcement learning remains largely unexplored.

To bridge the above gaps, we propose a unified abstraction-based robust training method for DRL systems. We are inspired by [23], which introduces an abstraction-layer encoding of interval-based state abstraction to facilitate reachability analysis of neural-network-controlled systems. While [23] leverages this encoding primarily for formal verification, we observe that the same abstraction mechanism is also a natural fit for *robust training* in DRL: by mapping perturbed observations into shared abstract cells, it promotes decision invariance under bounded perturbations, and it is compatible with existing DRL algorithms, including DQN [24], DDPG [25], PPO [26], A3C [27], SAC [28], etc.

Crucially, in the robust DRL setting we further exploit a property of state abstraction that has not been used for training: the abstraction induces a *finite abstract state space*. This finiteness allows us to efficiently compute per-state robustness certificates by reasoning over abstract states that preserve the same decision, and to incorporate these certificates into a **certificate-guided robust objective**, which is an additional robust loss function for training DRL systems that are verified to be robust in each training state. As a result, our method avoids expensive adversarial retraining and remains broadly compatible with standard DRL algorithms, while explicitly shaping the learned policy to be robust.

We implement our approach in the prototype SaRob (State-abstraction-based Robustification) and extensively evaluate it on various benchmarks, including Mujoco [29], Atari [30], ProcGen [31], as well as a series of classic control prob-

*Corresponding author.

lems [32]–[34]. We train a range of models with our approach, combined with normal DRL algorithms [24]–[28] and compare them with those trained by existing robust training approaches, e.g., RS-DQN [17], SA-MDP [10], RADIAL [11], and WocaR-RL [12]. Experimental results show that SaRob outperforms *all* competing approaches.

Overall, we make the following three main contributions:

1) We propose a unified abstraction-based robust training framework that can be plugged into standard DRL pipelines, enabling robustness-aware optimization while remaining compatible with a broad family of DRL algorithms.
2) We exploit the finite abstract state space to compute per-state robustness certificates efficiently, and incorporate them into a certificate-guided robust objective for training.
3) We implement a prototype called SaRob and evaluate it extensively on various DRL applications including Mujoco, Atari, ProcGen, and classic control problems. SaRob outperforms the existing approaches with an average of 92.0% and up to 312.3% improvement in system robustness.

II. Background and Problem Formulation

A. Abstraction-Based Training

In state abstraction, a set of individual system states is aggregated and represented as an abstract state in domains such as intervals, octagons, and polyhedra [35]. AI models can be trained on abstract states. Example models include Interval Neural Network (INN) [36] and Granular Neural Network [37]. Jin et al. [19] proposed training and verifying DRL systems on interval-based abstract states so that the trained systems can be model-checked to satisfy safety properties. Tian et al. [23] further leverage abstraction-based training to facilitate reachability analysis of DNN-controlled systems, by encoding interval-based state abstraction into the network via an abstraction layer. Earlier works [38], [39] have shown that state abstraction effectively trains near-optimal RL systems with finite state spaces. Oala et al. [40] demonstrated that INNs can produce fast, theoretically justified uncertainty scores for DNNs that are easy to interpret. Recently, Li [20] introduced neural episodic control with interval-based state abstraction, which is a simple but effective state-abstraction-based episodic control method with a more comprehensive episodic memory and a novel state evaluation. All these works show that the DRL systems trained using state abstraction are faster to validate and have higher sample efficiency than other existing approaches. Unlike them, in this paper, we focus on a unified robust training method for DRL systems through state abstraction and additional robust loss functions derived from abstract states, which is something that previous abstraction-based works have not paid attention to.

B. Robust Deep Reinforcement Learning

Several attempts have been made recently to improve the robustness of different DRL systems. Lütjens [14] proposed to compute guaranteed lower bounds on state-action values to determine the optimal action under a worst-case deviation in input space for DQN. RADIAL [11] designed adversarial loss functions by leveraging existing formal verification bounds w.r.t. neural network robustness to train robust DRL systems whose action spaces are discrete. SA-MDP [10] studied the fundamental properties of state-adversarial Markov decision processes and developed a theoretically principled policy regularization. WocaR-RL [12] proposed a robust training framework for DRL that directly estimated and optimized the worst-case reward of a policy trained by DQN or PPO under bounded attacks without requiring extra samples. All these works aim to improve the robustness of DRL systems. These approaches are restricted to specific DRL algorithms and cannot be generalized to all existing algorithms.

C. Problem Formulation

A DRL system can be formally modeled as a tuple $M = (S, A, R, p, \pi, \gamma, S_0)$, where S is a potentially infinite state space, A is the set of actions, $R : S \times A \times S \to \mathbb{R}$ is the reward function, $p : S \times A \to \mathcal{P}(S)$ is the transition probability function where $\mathcal{P}(S)$ is the set of all probability measures over S, $\pi : S \to \mathcal{P}(A)$ is the learned policy implemented by a DNN, $\gamma \in [0, 1]$ is the discount factor of reward, and $S_0 \subseteq S$ is the set of initial states [13]. In this paper, we focus on deterministic state transitions. That is, given a state $s \in S$ and a corresponding action $a \in A$, the successor state of s is uniquely determined by some system dynamics $f : S \times A \to S$ where f refers to some Dirac measure $p(\cdot \mid s, a)$.

This deterministic formulation allows us to study robustness by analyzing the sensitivity of the learned policy to bounded state perturbations. Intuitively, given a trained system and any state s, the system is called robust if it can still make the optimal decision when s is perturbed by up to a bounded magnitude ε. In the following, we consider l_p-bounded perturbations on states with magnitude ε, i.e., $\mathcal{B}^{\varepsilon} = \{\delta \in \mathbb{R}^n \mid \|\delta\|_p \leq \varepsilon\}$, and this formulation can be easily extended to other types of perturbations. Based on this perturbation model, we next formalize robustness in terms of the maximum allowable perturbation that preserves the policy action at a given state.

Definition 1 (Robustness Certificate [41]): Given a DRL system with policy π and a state s, the robustness certificate of s is the maximum perturbation magnitude $\overline{\varepsilon}(s)$, such that for any perturbation $\delta \in \mathcal{B}^{\overline{\varepsilon}(s)}$, the perturbed state $\tilde{s} := s + \delta$ has the same action as s according to π, i.e., $\pi(\tilde{s}) = \pi(s)$.

According to Definition 1, a larger $\overline{\varepsilon}(s)$ implies better robustness on the given state s [41]. However, it is almost impossible to directly compute the robustness certificates, not to mention employing the calculated results to improve the robustness of DRL systems. The challenging step of calculating $\overline{\varepsilon}(s)$ is to compute all the system states that have the same action with respect to a trained policy and the given state s. This step is computationally expensive for continuous-state systems due to the complexity of hybrid system behaviors [33] and the inclusion of opaque AI models like DNNs [42]. Thus, all existing approaches can either calculate only lower bounds of robustness certificates [41] or are applicable to some specific

types of systems and DNNs, e.g., the proposed approaches be applicable only to Q-learning algorithms [41] and ReLU networks [43].

In this work, we train a DRL system using state abstraction and solve the following problem:

> *Given a task M trained by any DRL algorithm, our goal is to calculate and maximize robustness certificate $\overline{\varepsilon}$ over all training states and consequently get a robust DRL system for M.*

III. Abstraction-Based Robust Training

This section presents our robust training method. We first introduce the key idea of calculating robustness certificates and then design robustness-defined loss functions by leveraging the computed certificates.

A. Interval-Based State Abstraction

Definition 2 (State Abstraction [38]): A state abstraction is a function: $\phi : S \to \hat{S}$, that maps each concrete system state $s \in S$ into an abstract state $\hat{s} \in \hat{S}$.

Given a set S of concrete states, one key step of training a DRL system via state abstraction is to construct a discrete and finite set $\hat{S}$ of abstract states with a corresponding function $\phi : S \to \hat{S}$ that maps each concrete state s in S into an abstract state $\hat{s}$ in $\hat{S}$ [20], [38]. ϕ is thus called a state abstraction function, and its dual function $\varphi : \hat{S} \to 2^S$ is called a concretization function such that for each $\hat{s} \in \hat{S}$, $\varphi(\hat{s}) = \{s | s \in S \wedge \phi(s) = \hat{s}\}$.

We consider the simplest yet effective state abstraction using intervals which has achieved remarkable performance [20] and yields easy-to-verify trained policies [19].

Given an n-dimensional system with a bounded continuous state space $S = \prod_{i=1}^{n}[L_i, U_i]$ where $L_i, U_i \in \mathbb{R}$, a system state $s \in S$ is a vector of n real numbers. We divide the interval $[L_i, U_i]$ of each dimension into a finite set of unit intervals. Concretely, we can discretize and abstract each interval $[L_i, U_i]$ into a finite set $\mathcal{I}_i = \{[L_i, L_i + g_i], \ldots, [U_i - g_i, U_i]\}$ of all the divided unit intervals. As a consequence, we obtain an abstract-state space $\hat{S} = \bigcup_{j=1}^{N} \hat{s}^j = \mathcal{I}_1 \times \ldots \times \mathcal{I}_n$. An abstract state $\hat{s}$ is essentially a vector of n unit intervals $(I_1, I_2, \ldots, I_n)$ with $I_i \in \mathcal{I}_i$:

$$\hat{s}^i = [L_i + \lfloor \frac{(s^i - L_i)}{g_i} \rfloor \times g_i, L_i + \lfloor \frac{(s^i - L_i)}{g_i} \rfloor \times g_i + g_i] \quad (1)$$

where $s = [s^1, \cdots, s^n]$ is the concrete state, $\hat{s} = [\hat{s}^1, \cdots, \hat{s}^n]$ is the corresponding abstract state, g_i is the length of i-th unit interval and $\lfloor \cdot \rfloor$ means the round down operator. We assume every abstract state has the same length of the unit interval and call it *abstraction granularity*. The non-uniform case will be considered as future work.

To implement interval-based state abstraction, we adopt an abstraction-layer encoding of the abstraction function as a plug-in wrapper to the policy network, following [23]. Specifically, we insert an abstraction layer between the input layer and the first hidden layer so that *Abstraction-Layer*$(s) = \phi(s)$ is fed to the remaining network during training and execution. We denote the resulting DRL system and policy by M_ϕ and π_ϕ, respectively.

The corresponding abstraction layer w.r.t. the above interval-based abstraction function ϕ is shown in Fig. 2. There are $2n$ neurons in the abstraction layer. Their activation functions are $\phi_i^u(x) = L_i + \lfloor \frac{(x - L_i)}{g_i} \rfloor \times g_i + g_i$, $\phi_i^l(x) = L_i + \lfloor \frac{(x - L_i)}{g_i} \rfloor \times g_i$, $i = 1, \cdots, n$, respectively. All biases are set to 0. The weights between any neuron s_i in the input layer and the neurons whose activation functions are $\phi_i^u(\cdot)$ and $\phi_i^l(\cdot)$ in the abstraction layer are 1, and 0 for the others. Finally, the constructed abstraction layer implements the interval-based state abstraction function, i.e., $\hat{s} =$ *Abstraction-Layer*(s).

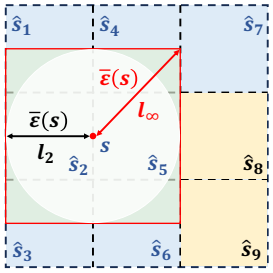


Fig. 1: Examples of calculating robustness certificates.

B. Robustness Certification on Abstract States

According to Definition 1, we can easily and accurately compute $\overline{\varepsilon}(s)$ for any arbitrarily given state s of a DRL system M_ϕ. We assume the action determined by the current policy in s is a, i.e., $\pi_\phi(s) = a$. Let $\hat{S}_a \subseteq \hat{S}$ be the subset of abstract states whose concrete states have the same action a according to π_ϕ:

$$\hat{S}_a = \{\hat{s} | \hat{s} \in \hat{S} \wedge \hat{s} = \phi(s) \wedge \pi_\phi(s) = a\}. \quad (2)$$

Benefiting from the finiteness of abstract state space, $\hat{S}_a$ can be computed in polynomial time by enumerating each abstract state $\hat{s}$ in $\hat{S}$ and checking if $\hat{s} = \phi(s) \wedge \pi_\phi(s) = a$ holds by picking a concrete state from each abstract state. According to $\hat{S}_a$ and the concretization function φ, we can define the subset of concrete state space whose elements have the same action a as s has:

$$S_a = \{s' | s' \in \varphi(\hat{s}) \wedge \hat{s} \in \hat{S}_a\}. \quad (3)$$

After obtaining S_a, we can compute $\overline{\varepsilon}(s)$ according to Definition 1 as follows:

$$\begin{aligned} \overline{\varepsilon}(s) = \max_{\varepsilon \in \mathbb{R}} \ & \varepsilon \\ \text{s.t. } \forall \delta \in \mathcal{B}^\varepsilon, \ & s + \delta \in S_a \end{aligned} \quad (4)$$

where $\mathcal{B}^\varepsilon = \{\delta \in \mathbb{R}^n \mid \|\delta\|_p \leq \varepsilon\}$ and S_a is the same as that in Equation (3). We show an illustrative example of computing robustness certificates for a 2D DRL system trained on interval-based abstract states in Fig. 1.

Example 1 (Calculating Robustness Certificates):

We show that our method can calculate the robustness certificates under any l_p-perturbation magnitude. Fig. 1 depicts an illustrative example of calculating $\overline{\varepsilon}(s)$ for a 2D DRL system trained on interval-based abstract states. Let S be the concrete state space of the system and $\hat{S} = \{\hat{s}_1, \ldots, \hat{s}_9\}$ be an abstract state space of S w.r.t. the abstraction function ϕ. For a given concrete state s, we assume $\phi(s) = \hat{s}_2$ and s has the action a_1 by the trained policy of the system. According to Equation (2), we check each abstract state in $\hat{S}$ and collect those states whose

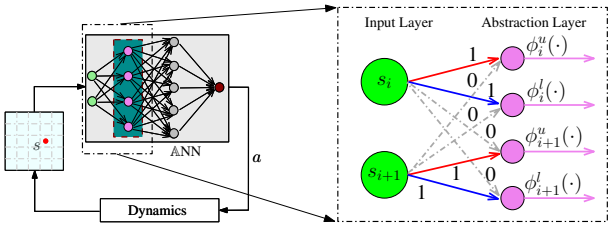


Fig. 2: Robust training via interval-based state abstraction.

action is a_1, i.e., $\hat{S}_{a_1}=\{\hat{s}_1,\ldots,\hat{s}_7\}$. Then, the corresponding S_{a_1} is the blue area in the state space. Finally, $\overline{\varepsilon}(s)$ with l_2-norm and l_∞-norm are shown in Fig. 1.

C. Abstraction-Based Robust Training

Following the abstraction-layer encoding in [23], we insert a fixed abstraction layer between the input layer and the first hidden layer to implement interval-based state abstraction. This layer is parameter-free and thus its connections are frozen; the remaining network parameters are trained with standard DRL optimization, implemented in mainstream frameworks such as TensorFlow and PyTorch [44], [45].

With the aid of state abstraction, we perform robust training for DRL systems by designing robust loss functions using the calculated $\overline{\varepsilon}(s)$. The loss function of our training method consists of two terms:

$$\mathcal{LOSS} = k\mathcal{L}_{nominal} + (1-k)\mathcal{L}_{robust} \tag{5}$$

where $\mathcal{L}_{nominal}$ denotes the nominal loss function for standard DRL algorithms, and $\mathcal{L}_{robust}$ denotes the robust loss which we design to account for perturbations. k is a hyperparameter controlling the trade-off between standard performance and robust performance with values between 0 and 1. Note that nominal DRL training algorithms have $k = 1$ throughout the full training process.

The design of $\mathcal{L}_{robust}$ is motivated by maximizing the $\overline{\varepsilon}(s)$ of each state s in the sampled batch, which indicates the policy should perform well under perturbations:

$$\mathcal{L}_{robust} = \frac{|batch|}{\sum_{s_i \in batch} \overline{\varepsilon}(s_i)} \tag{6}$$

Robust training can be achieved by adding robust loss functions, which maximize the robustness certificates over all the training states. That is, once the training is completed, the value of the loss function is minimized. According to Equation (6), the robustness certificates are therefore maximized. A maximal certificate implies a maximum perturbation magnitude under which any perturbation does not change the decision of the trained model, as a robust model is defined according to Definition 1. We depict the specific robust loss functions for DQN in Example 2.

Example 2 (Specific Robust Loss for DQN): The action space for DQN is finite, and the action is determined by the max Q-value. Given the sampled batch $\langle s_i, a_i, r_i, s_i', is\text{-}end_i\rangle, i = 1,\cdots,m$, we define y_i as follows:

$$y_i = \begin{cases} r_i & is\text{-}end_i = 1 \\ r_i + \gamma \max\limits_{a'} Q(s_i', a') & otherwise \end{cases} \tag{7}$$

where $Q(\cdot)$ calculates the Q-value of the (s', a') pair and γ is the discount factor. Then, the nominal loss for standard DQN is $\mathcal{L}_{nominal} = \frac{1}{m}\sum_{i=1}^{m}[y_i - Q(s_i, a_i)]^2$. Additionally, we can calculate robustness certificates for each s_i in the sampled batch using Equation (4). Finally, we have:

$$\mathcal{LOSS} = \frac{k}{m}\sum_{i=1}^{m}[y_i - Q(s_i, a_i)]^2 + \frac{(1-k)m}{\sum_{i=1}^{m}\overline{\varepsilon}(s_i)} \tag{8}$$

Then, the abstraction-based network can be trained using this loss function.

IV. Implementation and Evaluation

We have implemented a prototype called SaRob and extensively evaluated its effectiveness on a variety of tasks and benchmarks. In particular, we experimentally evaluate (i) the robustness improvement compared with existing robust training methods, (ii) the compatibility with different DRL algorithms, (iii) convergence and efficiency, and (iv) the effect of abstraction granularity.

A. Robustness Comparison on DRL-based Control Systems

We compare our robust training method with the existing methods, including SAMDP [10], RADIAL [11] and WocaR-RL [12], on four DRL-based control systems [32]–[34]. We train systems using the corresponding nominal DRL algorithms and four robust training algorithms including SaRob, and evaluate their performance under different perturbations and attacks, including Gaussian noise, PGD attacks and RS attacks [10].

In Table I, we highlight the best performance in each group of experiments. We can observe that when no perturbations or attacks are imposed, all training methods can produce DRL systems with performance close to that of the nominal ones, as reflected by the accumulated rewards. However, in the case of perturbing the systems by Gaussian noises, there are 10 out of 12 cases where the systems trained by our approach still maintain higher performance than those trained by competing approaches. Similarly, our approach performs best in 11 out of 12 cases of applying both PGD and RS attacks to the systems. Additionally, when ACC is perturbed by an RS attack (step=10 and $\varepsilon = 0.15$), the performance degradation of the system trained by our approach is around 0.9, whereas it is 97.1, 34.1, and 24.1 for the systems trained by nominal DDPG, SAMDP, and WocaR-RL, respectively.

B. Compatibility with Various DRL Algorithms

Our robust training approach is compatible with various existing DRL algorithms. We apply our training algorithm to the image-based DRL tasks using different DRL algorithms and compare the robustness of our trained models with those trained by existing robust training methods under different perturbations and attacks. For fair comparisons, the network architectures and training hyper-parameters are the same as those of the compared methods, except that we need to insert abstraction layers into the networks.

As shown in Table II, there are 10 out of 12 cases where the systems trained using SaRob-DQN retain the best performance, when perturbing the systems with different PGD attacks. In

TABLE I: Robustness comparison of state-based control systems under different perturbations.

	Methods	w/o perturbation	Gaussian Noise			PGD attack (step=50)		
			σ = 0.02	σ=0.04	σ=0.06	ε=0.01	ε=0.03	ε=0.05
Mountain Car	Nominal	−108.2 ± 4.1	−112.6 ± 12.0	−136.0 ± 19.4	−166.2 ± 15.2	−109.2 ± 2.3	−118.9 ± 12.9	−147.6 ± 10.3
	SAMDP	**−106.1 ± 5.7**	−111.2 ± 6.7	−121.4 ± 10.9	−156.6 ± 21.1	−112.8 ± 8.4	−116.3 ± 9.3	−136.6 ± 12.4
	RADIAL	−108.0 ± 1.3	−109.1 ± 1.7	−123.3 ± 10.5	−159.7 ± 15.9	−108.5 ± 1.3	−115.9 ± 3.8	−131.4 ± 10.4
	WocaR-RL	−107.2 ± 1.9	−108.2 ± 2.2	−114.7 ± 7.3	−143.4 ± 16.7	**−108.2 ± 1.8**	−114.0 ± 8.9	−131.7 ± 12.3
	SaRob	−106.2 ± 2.3	**−106.4 ± 1.7**	**−112.4 ± 2.1**	**−137.4 ± 6.8**	−111.6 ± 6.2	**−113.8 ± 4.8**	**−125.3 ± 9.1**
	Methods	**w/o perturbation**	**Gaussian Noise**			**RS attack (step=50)**		
			σ=0.04	σ=0.08	σ=0.12	ε=0.4	ε=0.8	ε=1.2
B1	Nominal	−119.3 ± 1.9	−119.5 ± 2.1	−122.9 ± 4.5	−137.3 ± 11.4	−214.7 ± 8.7	−233.8 ± 47.3	−319.2 ± 172.8
	SAMDP	−118.3 ± 1.1	−118.4 ± 1.2	−118.8 ± 1.3	−123.8 ± 6.2	−189.3 ± 54.0	−194.3 ± 48.3	−296.7 ± 145.4
	WocaR-RL	−117.9 ± 1.0	−118.6 ± 0.9	−120.8 ± 0.8	−122.5 ± 6.3	−183.8 ± 41.6	−209.2 ± 46.2	−319.8 ± 153.8
	SaRob	**−117.7 ± 0.9**	**−117.7 ± 0.8**	**−118.0 ± 0.8**	**−122.4 ± 2.4**	**−157.4 ± 31.1**	**−190.0 ± 45.8**	**−217.6 ± 40.1**
	Methods	**w/o perturbation**	**Gaussian Noise**			**RS attack (step=70)**		
			σ = 0.02	σ = 0.03	σ = 0.04	ε = 0.01	ε = 0.015	ε = 0.02
Pendulum	Nominal	−101.2 ± 3.2	−290.3 ± 2.7	−425.2 ± 3.2	−481.8 ± 2.0	−106.3 ± 5.5	−204.1 ± 6.1	−247.0 ± 6.2
	SAMDP	−99.8 ± 3.1	−101.5 ± 3.8	**−109.8 ± 3.5**	−159.7 ± 2.9	−107.8 ± 8.2	−110.0 ± 12.3	−158.3 ± 12.8
	WocaR-RL	−97.3 ± 4.1	**−98.9 ± 4.7**	−116.2 ± 5.7	−147.4 ± 12.9	−104.9 ± 8.7	−110.6 ± 9.1	−142.8 ± 27.3
	SaRob	**−96.3 ± 5.3**	−100.9 ± 7.7	−110.0 ± 10.9	**−143.9 ± 13.8**	**−99.9 ± 7.9**	**−105.4 ± 8.7**	**−111.2 ± 19.8**
	Methods	**w/o perturbation**	**Gaussian Noise**			**RS attack (step=10)**		
			σ = 0.1	σ = 0.2	σ = 0.3	ε = 0.05	ε = 0.1	ε = 0.15
ACC	Nominal	**−123.3 ± 2.3**	−342.9 ± 48.0	−361.5 ± 41.4	−395.4 ± 100.7	−125.3 ± 2.3	−131.4 ± 9.1	−220.4 ± 79.1
	SAMDP	−124.9 ± 2.2	−254.6 ± 9.2	−282.2 ± 12.1	−300.8 ± 97.9	−125.1 ± 2.3	−133.2 ± 11.8	−159.0 ± 38.1
	WocaR-RL	−124.0 ± 3.5	−244.8 ± 6.0	−265.5 ± 7.6	−282.9 ± 112.1	−124.9 ± 1.2	−128.9 ± 12.3	−148.1 ± 39.1
	SaRob	−123.7 ± 2.8	**−125.9 ± 4.0**	**−134.7 ± 17.0**	**−160.2 ± 60.4**	**−123.7 ± 2.8**	**−123.7 ± 2.8**	**−124.6 ± 3.1**

TABLE II: Robustness comparison of image-based Atari games under different PGD attacks.

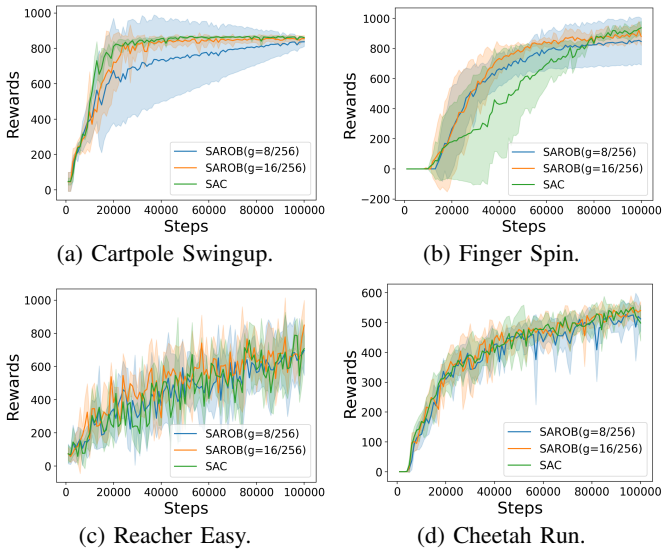
Environment	Attack	Method					
		DQN	**RS-DQN**	**SA-DQN**	**RADIAL-DQN**	**WocaR-DQN**	**SaRob-DQN**
BankHeist	w/o Attack	1325.5 ± 5.7	238.7	1237.6 ± 1.7	1349.5 ± 1.7	1220.0 ± 12.0	**1350.0 ± 10.0**
	PGD Attack (ε=1/255)	29.5 ± 2.4	190.7	1237.0 ± 2.0	1349.5 ± 1.7	1220.0 ± 12.0	**1350.0 ± 10.0**
	PGD Attack (ε=3/255)	0.0 ± 0.0	*N/A*	1213.0 ± 2.5	1348.0 ± 1.7	1214.0 ± 7.0	**1350.0 ± 10.0**
	PGD Attack (ε=5/255)	0.0 ± 0.0	*N/A*	1130.0 ± 29.1	**1182.5 ± 43.3**	*N/A*	1173.3 ± 118.7
Freeway	w/o Attack	33.9 ± 0.1	32.93	30.0 ± 0.0	33.2 ± 0.2	31.2 ± 0.4	**34.0 ± 0.0**
	PGD Attack (ε=1/255)	0.0 ± 0.0	32.53	30.0 ± 0.0	33.4 ± 0.2	31.2 ± 0.5	**34.0 ± 0.0**
	PGD Attack (ε=3/255)	0.0 ± 0.0	*N/A*	30.1 ± 0.1	33.4 ± 0.1	31.4 ± 0.3	**34.0 ± 0.0**
	PGD Attack (ε=5/255)	0.0 ± 0.0	*N/A*	27.65 ± 0.2	29.1 ± 0.2	*N/A*	**29.7 ± 0.7**
Pong	w/o Attack	**21.0 ± 0.0**	19.73	**21.0 ± 0.0**	**21.0 ± 0.0**	**21.0 ± 0.0**	**21.0 ± 0.0**
	PGD Attack (ε=1/255)	−21.0 ± 0.0	19.73	**21.0 ± 0.0**	**21.0 ± 0.0**	**21.0 ± 0.0**	**21.0 ± 0.0**
	PGD Attack (ε=3/255)	−21.0 ± 0.0	*N/A*	**21.0 ± 0.0**	**21.0 ± 0.0**	**21.0 ± 0.0**	**21.0 ± 0.0**
	PGD Attack (ε=5/255)	−20.9 ± 0.1	*N/A*	−19.8 ± 0.1	**21.0 ± 0.0**	*N/A*	**21.0 ± 0.0**
RoadRunner	w/o Attack	43390 ± 973	12106.67	**45870 ± 1380**	44495 ± 1165	44156 ± 2279	42350 ± 1843
	PGD Attack (ε=1/255)	0.0 ± 0.0	5753.33	44300 ± 1753	**44445 ± 1148**	44079 ± 2154	42066 ± 1182
	PGD Attack (ε=3/255)	0.0 ± 0.0	*N/A*	20170 ± 1822	39560 ± 1621	38720 ± 1765	**40266 ± 1766**
	PGD Attack (ε=5/255)	0.0 ± 0.0	*N/A*	3350 ± 335	23820 ± 942	*N/A*	**35350 ± 1134**

particular, SaRob-DQN achieves up to 9.6 times performance improvement compared to SA-DQN when the perturbation magnitude becomes large ($\varepsilon = 5/255$).

Table III presents the comparison of SaRob-DDPG and SA-DDPG [10] under different attacks including Critic, Random, MAD, RS, and RS+MAD. In 15 out of 20 cases, the systems trained by SaRob-DDPG retain the best performance. Additionally, for the RS attacks and RS+MAD attacks, the systems trained using SaRob-DDPG can maintain the best robustness for all tasks.

Table IV shows comparisons between our robust training method and RADIAL for A3C and PPO, respectively. In the case of perturbing the systems with different PGD attacks, there are 7 out of 9 cases where the systems trained by our approach retain the best performance. In particular, SaRob-A3C achieves up to 77.6 times performance improvement compared to nominal A3C when the perturbation magnitude becomes large (RoadRunner with $\varepsilon = 5/255$).

One interesting observation from the experiments is that these robust training methods including SaRob can perform better than nominal algorithms even in non-perturbation settings. The results suggest that a robust objective can be helpful to improve systems' performance, which is consistent with the claim of SAMDP [10].



(a) Cartpole Swingup. (b) Finger Spin.

(c) Reacher Easy. (d) Cheetah Run.

Fig. 3: Learning curves of SAC and SaRob-SAC.

TABLE III: Robustness comparison of Mujoco environments under different attacks.

Environment	Method	Accumulated Reward					
		w/o Attack	Critic	Random	MAD	RS	RS+MAD
Ant	DDPG	1931.5 ± 687.5	210.1 ± 273.8	1696.4 ± 752.7	195.1 ± 179.4	301.8 ± 236.1	188.6 ± 242.7
	SA-DDPG (SGLD)	2204.1 ± 803.6	1982.5 ± 600.1	2143.4 ± 604.2	**2201.7 ± 429.6**	2014.3 ± 472.3	2056.4 ± 649.6
	SA-DDPG (Convex)	**2251.4 ± 455.2**	1716.8 ± 635.7	**2208.7 ± 403.5**	2091.9 ± 562.8	1758.7 ± 637.9	2031.6 ± 666.9
	SaRob-DDPG	2167.1 ± 831.9	**2110.8 ± 630.7**	2164.1 ± 801.0	2111.6 ± 660.4	**2103.2 ± 268.4**	**2099.0 ± 623.3**
Hopper	DDPG	**3447.1 ± 544.1**	2267.5 ± 1311.7	**3342.7 ± 668.7**	2841.8 ± 1010.2	590.4 ± 129.8	481.5 ± 1162.8
	SA-DDPG (SGLD)	3059.6 ± 286.5	**2875.6 ± 424.1**	3056.3 ± 160.6	3008.4 ± 289.9	1603.4 ± 638.3	1593.9 ± 634.4
	SA-DDPG (Convex)	3074.9 ± 558.4	2663.2 ± 772.1	3051.3 ± 573.9	2895.0 ± 718.6	1486.1 ± 713.1	1369.6 ± 566.4
	SaRob-DDPG	3070.3 ± 283.5	2815.5 ± 544.1	3063.9 ± 243.5	**3010.9 ± 391.1**	**1615.8 ± 637.2**	**1668.7 ± 696.6**
Reacher	DDPG	**−4.6 ± 1.4**	−24.2 ± 4.9	**−9.3 ± 2.8**	−28.6 ± 4.5	−20.5 ± 5.3	−28.3 ± 4.2
	SA-DDPG (SGLD)	−5.6 ± 1.7	−12.4 ± 4.0	−11.8 ± 4.4	−12.5 ± 4.2	−11.9 ± 4.2	−12.5 ± 4.2
	SA-DDPG (Convex)	−5.3 ± 1.5	−11.6 ± 3.5	−9.9 ± 3.2	−12.4 ± 3.3	−11.4 ± 3.2	**−12.4 ± 3.3**
	SaRob-DDPG	−5.8 ± 2.0	**−11.2 ± 3.4**	−9.8 ± 3.2	**−11.4 ± 3.5**	**−10.7 ± 3.1**	**−12.4 ± 3.5**
Inv. Pend.	DDPG	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	283.8 ± 319.1	84.3 ± 102.7
	SA-DDPG (SGLD)	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	577.2 ± 91.0	498.6 ± 74.3
	SA-DDPG (Convex)	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**
	SaRob-DDPG	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**

TABLE IV: Robustness comparison of Atari games and ProcGen benchmarks under different PGD attacks.

Environment	Attack	Method		
		A3C	RADIAL-A3C	SaRob-A3C
BankHeist	w/o Attack	1109.0 ± 21.4	1036.5 ± 23.4	**1193.3 ± 20.5**
	ε=1/255	**1102.5 ± 49.4**	975.0 ± 22.2	1013.3 ± 102.7
	ε=3/255	534.5 ± 58.2	949.0 ± 19.5	**965.0 ± 84.8**
	ε=5/255	115.0 ± 27.8	712.0 ± 46.4	**820.0 ± 60.8**
Pong	w/o Attack	**21.0 ± 0.0**	**21.0 ± 0.0**	**21.0 ± 0.0**
	ε=1/255	**21.0 ± 0.0**	**21.0 ± 0.0**	**21.0 ± 0.0**
	ε=3/255	**21.0 ± 0.0**	**21.0 ± 0.0**	**21.0 ± 0.0**
	ε=5/255	−17.85 ± 0.33	**21.0 ± 0.0**	**21.0 ± 0.0**
RoadRunner	w/o Attack	34420 ± 604	**34825 ± 981**	34540 ± 596
	ε=1/255	31040 ± 2173	31960 ± 933	**32650 ± 847**
	ε=3/255	3025 ± 317	29920 ± 1496	**30600 ± 1064**
	ε=5/255	350 ± 93	**31545 ± 1480**	27150 ± 1784
		PPO	**RADIAL-PPO**	**SaRob-PPO**
Fruitbot	w/o Attack	**26.09 ± 0.33**	26.08 ± 0.29	26.06 ± 0.30
	ε=1/255	22.53 ± 0.38	**26.06 ± 0.30**	25.90 ± 0.29
	ε=3/255	13.51 ± 0.39	25.87 ± 0.30	**25.89 ± 0.31**
	ε=5/255	8.51 ± 0.35	**25.81 ± 0.30**	11.51 ± 0.37
Coinrun	w/o Attack	6.65 ± 0.15	6.66 ± 0.15	**6.79 ± 0.15**
	ε=1/255	5.22 ± 0.16	6.71 ± 0.15	**6.77 ± 0.15**
	ε=3/255	3.58 ± 0.15	6.71 ± 0.15	**6.77 ± 0.15**
	ε=5/255	3.36 ± 0.15	6.67 ± 0.15	**6.69 ± 0.15**
Jumper	w/o Attack	**4.22 ± 0.16**	3.85 ± 0.15	3.99 ± 0.15
	ε=1/255	3.90 ± 0.15	3.93 ± 0.15	**3.97 ± 0.15**
	ε=3/255	3.10 ± 0.15	3.75 ± 0.15	**3.78 ± 0.15**
	ε=5/255	3.15 ± 0.15	3.59 ± 0.15	**3.66 ± 0.15**

TABLE V: Robustness comparisons of MuJoCo tasks trained by SaRob with different abstraction granularities.

Environment	Attack	SaRob-SAC		
		g =8/256	g =16/256	g =32/256
Cartpole Swingup	w/o Attack	851.3 ± 8.6	**855.7 ± 16.1**	839.9 ± 11.2
	σ=5/256	850.2 ± 9.5	**854.9 ± 16.1**	838.8 ± 23.6
	σ=10/256	848.3 ± 8.4	**855.0 ± 16.6**	835.6 ± 24.3
	σ=15/256	851.5 ± 9.0	**853.8 ± 16.5**	831.2 ± 30.6
	σ=20/256	850.0 ± 9.7	**852.8 ± 15.8**	825.8 ± 28.2
Finger Spin	w/o Attack	842.8 ± 137.2	**880.7 ± 52.2**	808.1 ± 150.9
	σ=5/256	841.9 ± 139.5	**875.4 ± 49.4**	809.2 ± 145.6
	σ=10/256	844.9 ± 140.9	**880.3 ± 46.1**	810.8 ± 146.4
	σ=15/256	835.8 ± 138.9	**883.2 ± 43.7**	806.4 ± 140.2
	σ=20/256	835.4 ± 139.3	**878.7 ± 42.8**	803.6 ± 141.1
Reacher Easy	w/o Attack	**728.4 ± 91.3**	724.2 ± 115.1	482.3 ± 79.5
	σ=5/256	**732.0 ± 91.5**	726.1 ± 118.3	474.3 ± 52.0
	σ=10/256	679.6 ± 121.8	**718.6 ± 58.3**	424.8 ± 104.9
	σ=15/256	626.4 ± 46.4	**723.0 ± 100.2**	442.3 ± 24.1
	σ=20/256	622.5 ± 185.6	**647.8 ± 158.6**	405.3 ± 76.4

C. Convergence and Efficiency Analysis

Figure 3 shows the learning curves of SAC and SaRob-SAC under different granularities in four MuJoCo environments, and each task is trained for one million frames. The solid line and shaded regions represent the mean and standard deviation, respectively. As the training progresses, the trends of SAC and our method are consistent, which means our abstraction-based robust training does not affect the convergence of algorithms. A suitable granularity can enable our robust training method to achieve the same performance as the nominal one under the same number of training steps.

We use SaRob-SAC as an example to evaluate its efficiency on the high-dimensional image-based tasks in MuJoCo environment. We train these tasks with one million frames on a single 24564MiB GPU using SAC. The training times for Cartpole Swingup, Finger Spin and Reacher Easy are 4.81, 21.87 and 13.32 hours, respectively. We train the same tasks using SaRob, and the network architectures, training hyperparameters, and other settings are the same as those in SAC except that we need to insert an abstraction layer into the networks. The training times of SaRob on these tasks are 5.64, 24.93 and 13.78 hours, respectively. SaRob incurs less than 11% training-time overhead on average due to the computation of robustness certificates in the loss.

D. Effect of Abstraction Granularity in Robust Training

We evaluate the effect of abstraction granularity by training the MuJoCo tasks with different granularities and examining the system performance in terms of accumulated rewards under different levels of Gaussian perturbations. Table V shows that when the granularity increases from 8/256 to 16/256, the robustness of the systems increases with the increase of abstraction granularities, as shown by the bold rewards in the table. However, as the abstraction granularity further increases (32/256), systems' performance is affected and decreases even if no perturbations are imposed. That is because some important information may be lost due to coarse abstraction. The results show that the abstraction granularity of training data is an important factor in balancing DRL systems' performance and robustness; however, this factor is not considered by existing robust DRL approaches.

V. Conclusion

We presented a unified framework for robust training of DRL systems through state abstraction and certificate-guided

optimization. Building on an abstraction-layer encoding of interval-based state abstraction, our approach leverages the induced finite abstract state space to efficiently compute per-state decision-robustness certificates and incorporates them into a certificate-guided robust objective. This design makes the robustness signal explicit at the abstract-state level and directly usable during policy optimization. We have implemented a prototype SaRob and extensively evaluated it on tasks from various platforms. The experimental results demonstrate that SaRob outperforms existing robust training approaches, and abstraction granularity is identified as an important factor in balancing system performance and robustness in DRL. Future work will extend our certificate-guided robust training to richer perturbation models and stronger robustness guarantees for safety-critical DRL applications.

References

[1] Y. Zhao, K. Wu, Z. Xu, Z. Che, Q. Lu, J. Tang, and C. H. Liu, "CADRE: A cascade deep reinforcement learning framework for vision-based autonomous urban driving," in *AAAI*, 2022, pp. 3481–3489.

[2] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 5068–5078, 2022.

[3] K. M. Oikonomou, I. Kansizoglou, and A. Gasteratos, "A hybrid reinforcement learning approach with a spiking actor network for efficient robotic arm target reaching," *IEEE Robotics Autom. Lett.*, vol. 8, no. 5, pp. 3007–3014, 2023.

[4] L. Garaffa, M. Basso, A. Konzen, and E. P. de Freitas, "Reinforcement learning for mobile robotics exploration: A survey," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 34, no. 8, pp. 3796–3810, 2023.

[5] A. Agrawal, A. S. Bedi, and D. Manocha, "RTAW: an attention inspired reinforcement learning method for multi-robot task allocation in warehouse environments," in *ICRA*, 2023, pp. 1393–1399.

[6] Y. Zhang, H. Zhu, D. Tang, T. Zhou, and Y. Gui, "Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems," *Robotics Comput. Integr. Manuf.*, vol. 78, p. 102412, 2022.

[7] J. Su, J. Huang, S. C. Adams, Q. Chang, and P. A. Beling, "Deep multi-agent reinforcement learning for multi-level preventive maintenance in manufacturing systems," *Expert Syst. Appl.*, vol. 192, p. 116323, 2022.

[8] M. Schneckenreither, S. Haeussler, and J. Peiró, "Average reward adjusted deep reinforcement learning for order release planning in manufacturing," *Knowl. Based Syst.*, vol. 247, p. 108765, 2022.

[9] T.-W. Weng, K. D. Dvijotham, J. Uesato, K. Xiao, S. Gowal, R. Stanforth, and P. Kohli, "Toward evaluating robustness of deep reinforcement learning with continuous control," in *ICLR*, 2020.

[10] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. S. Boning, and C. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on state observations," in *NeurIPS*, 2020, pp. 21 024–21 037.

[11] T. P. Oikarinen, W. Zhang, A. Megretski, L. Daniel, and T. Weng, "Robust deep reinforcement learning through adversarial loss," in *NeurIPS*, 2021, pp. 26 156–26 167.

[12] Y. Liang, Y. Sun, R. Zheng, and F. Huang, "Efficient adversarial training without attacking: Worst-case-aware robust reinforcement learning," in *NeurIPS*, 2022.

[13] A. Kumar, A. Levine, and S. Feizi, "Policy smoothing for provably robust reinforcement learning," in *ICLR*, 2022.

[14] B. Lütjens, M. Everett, and J. How, "Certified adversarial robustness for deep reinforcement learning," in *CoRL*, 2019, pp. 1328–1337.

[15] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," in *ICLR, Workshop Track Proceedings*, 2017.

[16] V. Behzadan and A. Munir, "Whatever does not kill deep reinforcement learning, makes it stronger," *CoRR*, vol. abs/1712.09344, 2017.

[17] M. Fischer, M. Mirman, S. Stalder, and M. T. Vechev, "Online robustness training for deep reinforcement learning," *CoRR*, vol. abs/1911.00887, 2019.

[18] X. Pan, C. Xiao, W. He, S. Yang, J. Peng *et al.*, "Characterizing attacks on deep reinforcement learning," in *AAMAS*, 2022, pp. 1010–1018.

[19] P. Jin, J. Tian, D. Zhi, X. Wen, and M. Zhang, "Trainify: A cegar-driven training and verification framework for safe deep reinforcement learning," in *CAV*, 2022, pp. 193 – 218.

[20] Z. Li, D. Zhu, Y. Hu, X. Xie, L. Ma, Y. Zheng, Y. Song *et al.*, "Neural episodic control with state abstraction," in *ICLR*, 2023.

[21] Z. Zhang, Z. Xue, Y. Chen, S. Liu, Y. Zhang, J. Liu, and M. Zhang, "Boosting verified training for robust image classifications via abstraction," in *CVPR*, 2023, pp. 16 251–16 260.

[22] M. Vechev, "Technical perspective: Beautiful symbolic abstractions for safe and secure machine learning," *Commun. ACM*, vol. 66, no. 2, p. 104, 2023.

[23] J. Tian, D. Zhi, S. Liu, P. Wang, G. Katz, and M. Zhang, "Taming reachability analysis of dnn-controlled systems via abstraction-based training," in *International Conference on Verification, Model Checking, and Abstract Interpretation*. Springer, 2023, pp. 73–97.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *ICLR*, 2016.

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR,abs/1707.06347*, 2017.

[27] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML*, vol. 48, 2016, pp. 1928–1937.

[28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *ICML*, vol. 80, 2018, pp. 1856–1865.

[29] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IROS*, 2012, pp. 5026–5033.

[30] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, 2013.

[31] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," in *ICML*, vol. 119, 2020, pp. 2048–2056.

[32] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016, arXiv:1606.01540.

[33] T. Johnson, D. Manzanas, P. Musau *et al.*, "Arch-comp20 category report: artificial intelligence and neural network control systems (AINNCS) for continuous and hybrid systems plants," *EPiC Series in Computing*, vol. 74, 2020.

[34] R. Ivanov, T. Carpenter, J. Weimer, R. Alur, G. Pappas, and I. Lee, "Verisig 2.0: Verification of neural network controllers using taylor model preconditioning," in *CAV*, 2021, pp. 249–262.

[35] G. Singh, M. Püschel, and M. T. Vechev, "A practical construction for decomposing numerical abstract domains," *Proc. ACM Program. Lang.*, vol. 2, no. POPL, pp. 55:1–55:28, 2018.

[36] M. Beheshti, A. Berrached, A. Korvin, C. Hu *et al.*, "On interval weighted three-layer neural networks," in *ASS*, 1998, pp. 188–194.

[37] W. Pedrycz and G. Vukovich, "Granular neural networks," *Neurocomputing*, vol. 36, no. 1-4, pp. 205–224, 2001.

[38] D. Abel, "A theory of abstraction in reinforcement learning," Ph.D. dissertation, Brown University, USA, 2020.

[39] S. Sinclair, T. Wang, G. Jain, S. Banerjee, and C. Yu, "Adaptive discretization for model-based reinforcement learning," in *NeurIPS*, vol. 31, 2020, pp. 3858–3871.

[40] L. Oala, C. Heiß, J. Macdonald, M. März, W. Samek, and G. Kutyniok, "Interval neural networks: Uncertainty scores," *arXiv preprint arXiv:2003.11566*, 2020.

[41] F. Wu, L. Li, Z. Huang, Y. Vorobeychik, D. Zhao, and B. Li, "CROP: certifying robust policies for reinforcement learning through functional smoothing," in *ICLR*, 2022.

[42] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *CAV*, 2017, pp. 97–117.

[43] E. Bacci, M. Giacobbe, and D. Parker, "Verifying reinforcement learning up to infinity," in *IJCAI*, 2021, pp. 2154–2160.

[44] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[45] A. Paszke, S. Gross, F. Massa, A. Lerer *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *NeurIPS*, vol. 32, 2019.