

# Reachability Verification Based Reliability Assessment for Deep Reinforcement Learning Controlled Robotics and Autonomous Systems

Yi Dong<sup>1</sup>, Xingyu Zhao<sup>1</sup>, Sen Wang<sup>2</sup> and Xiaowei Huang<sup>1</sup>

**Abstract**—Deep Reinforcement Learning (DRL) has achieved impressive performance in robotics and autonomous systems (RASs). A key impediment to its deployment in real-life operations is the spuriously unsafe DRL policies—unexplored states may lead the agent to make wrong decisions that may cause hazards, especially in applications where end-to-end controllers of the RAS were trained by DRL. In this paper, we propose a novel quantitative reliability assessment framework for DRL-controlled RASs, leveraging verification evidence generated from formal reliability analysis of neural networks. A two-level verification framework is introduced to check the safety property with respect to inaccurate observations that are due to e.g., environmental noises and state changes. Reachability verification tools are leveraged at the local level to generate safety evidence of trajectories, while at the global level, we quantify the overall reliability as an aggregated metric of local safety evidence, according to an operational profile. The effectiveness of the proposed verification framework is demonstrated and validated via experiments on real RASs.

## I. INTRODUCTION

Deep Reinforcement Learning (DRL) has achieved impressive experimental results in video game playing, in which DRL agents are deployed under a trial-error-replay model. However, safety critical applications normally are not able to trial and replay at will in the real-world, such as autonomous vehicles, power systems, and humanoid robots. A recent trend in autonomous navigation, including ground and underwater vehicles, is to use end-to-end controllers trained by reinforcement learning methods [1]–[4]. For these applications that require a high level of safety integrity, deploying unverified DRL policies can lead to catastrophic consequences. In the meantime, Deep Neural Networks (DNN) in DRL algorithms are known to be unrobust to adversarial examples, i.e., the output of a DNN may be subject to dramatic change under a minor input disturbance. Such issues have motivated this work to formally analyse the end-to-end DRL systems and assure their reliability before they are deployed.

DRL verification and testing are emerging in recent years, including safe exploration, run-time monitoring, adversarial

training, etc.. Pattanaik *et al.* first proposed adversarial attacks for DRL algorithms, which trains the DRL with engineered adversarial attacks to improve the DRL performance and obtain more robust DRL models [5]. However, these adversarial training methods cannot guarantee the safety during the training process. To this end, Dalal *et al.* designed a safety layer that solves an action correction formulation per state [6]. Felix *et al.* used the Lyapunov function to calculate the region of attraction for a specific policy, and applied statistical models to obtain high-performance DRL policies [7]. These algorithms focus on the safety property in the training process, but the testing environment may be different from the training environment. Therefore, run-time monitors were proposed to assure the safety during the operations. The shield structure is created to prevent the agent from making unsafe decisions [8]—it bans all unsafe actions for each state to achieve safe reinforcement learning.

Aforementioned methods answer the *binary/worst-case* question of whether there exists any safety violation in the presence of extreme perturbations/attacks. They do not, however, provide an *overall* understanding of how safe the DRL policy is whenever a violation can be found locally (in line with the insight gained from evaluating Deep Learning (DL) classifiers [9], [10]). In this regard, we introduce a reliability metric based on the *probabilistic* notion of proportions of safety violations in the global input space representing the environment and formulated by the Operational Profile (OP). Furthermore, most DRL algorithms are explored and learned in simulation environments for, e.g., efficiency and cost considerations. However, the gap between simulated and real environments may lead to violation of the safety property set in simulation. Observation in the simulation environment is assumed as an accurate and unbiased signal, which is an unacceptably strong assumption in safety critical context. To this end, we proposed a novel two-level verification framework to assess the reliability of DRL algorithms: i) at the local level of an initial state, reachability analysis tools are utilised to generate safety verification evidence that are interval based trajectories starting from an initial state; ii) at the global level, we borrow ideas from software reliability engineering to model the distribution of initial states as the OP (representing all possible operational scenarios), and then aggregate local safety evidence according to the OP to statistically estimate the overall reliability.

The main contributions of this paper include:

a) A two-level assessment framework is designed to assess the reliability of DRL algorithms. Reachability analysis formally verifies the safety of trajectories starting from an

\*This work is supported by UK Dstl through Safety Argument for Learning-enabled Autonomous Underwater Vehicles and UK EPSRC through End-to-End Conceptual Guarding of Neural Architectures [EP/T026995/1]. This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 956123.

<sup>1</sup>Yi Dong, Xingyu Zhao and Xiaowei Huang are with the Department of Computer Science, University of Liverpool, Liverpool, UK {yi.dong, xingyu.zhao, xiaowei.huang}@liverpool.ac.uk

<sup>2</sup>Sen Wang is with the Department of Electrical and Electronic Engineering, Imperial College London, London, UK sen.wang@imperial.ac.uk

initial state at the local level, while overall reliability claims are supported statistically at the global level across initial states.

b) At the local level, the state-of-the-art reachability verification tool POLAR [11] is integrated and optimised for faster computation and tighter bounded results. Meanwhile, OP is introduced and approximated to support a global level reliability claims.

c) A publicly accessible repository of our method with all source code, datasets, experiments and a real-world case study based on BlueRov2 unmanned underwater vehicles (UUVs).

The rest of this paper is organised as follows. The mathematical preliminaries used in this paper are summarised in Section II. A detailed two-level verification algorithm is demonstrated in Section III, including local-level reachability analysis and global-level statistical analysis. Simulation results and corresponding discussion are presented in Section IV. Finally, Section V concludes this paper.

## II. PRELIMINARIES

### A. Deep Reinforcement Learning

We use discounted infinite-horizon Markov Decision Process (MDP) to model the interaction of an agent with the environment  $E$ . An MDP is a 5-tuple  $\mathcal{M}^E = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ , where  $\mathcal{S}$  is the state space,  $\mathcal{A}$  is the action space,  $\mathcal{P}(\mathbf{x}'|\mathbf{x}, \mathbf{a})$  is a probabilistic transition,  $\mathcal{R}(\mathbf{x}, \mathbf{a}) \in \mathbb{R}_{\geq 0}$  is a reward function, and  $\gamma \in [0, 1)$  is a discount factor. We use  $\mathbf{x}$  to range over the state space  $\mathcal{S}$  because it not only is a state but also will later be used as input to a policy neural network. We consider different DRL algorithms, such as DDPG [12], TD3 [13], PPO [14], which return an optimal policy  $\pi^*$  includes a mapping  $\mu: \mathcal{S} \rightarrow \mathcal{A}$  that maps from states to actions.

Based on  $\mathcal{M}^E$ , a policy  $\pi$  induces a trajectory distribution  $\rho^{\pi, E}(\zeta)$  where  $\zeta = (\mathbf{x}_0, \mathbf{a}_0, \mathbf{x}_1, \mathbf{a}_1, \dots)$  denotes a random trajectory. The state-action value function of  $\pi$  is defined as

$$Q^\pi(\mathbf{x}, \mathbf{a}) = \mathbb{E}_{\zeta \sim \rho^{\pi, E}} \left[ \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(\mathbf{x}_t, \mathbf{a}_t) \right] \quad (1)$$

and the state value function of  $\pi$  is  $V^\pi(\mathbf{x}) = Q^\pi(\mathbf{x}, \pi(\mathbf{x}))$ .

We consider a DRL driven robot that navigates, and avoids collisions, in a complex environment where there are static and dynamic objects (or obstacles). The interaction of the robot with the environment can be modelled as an MDP. At each time  $t$ , the robot has its sensor observation from the environment, namely state  $\mathbf{x}_t$ , i.e.,  $\mathbf{x}_t = (o_t^1, o_t^2, \dots, o_t^n)^T$  where  $o_t^1, o_t^2, \dots, o_t^n$  are observable sensor signals at time  $t$ . It is worth noting that the robot's parameters are also formatted as observations because these variables are relative to the environment, such as the pose in the global coordinate system and the velocity relative to the stationary ground. The sensors can only observe partial information of the environment, e.g., by scanning the environment within a certain distance. For example, the observation range is within 3.15 metres in Turtlebot Waffle Pi [15] for a distance sensor.

An action  $\mathbf{a}_t \in \mathcal{A}$  consists of several decision variables. With the Proportional-Integral-Derivative (PID) controller on the robot, we consider two action variables, representing line velocity and angle velocity, respectively, i.e.,  $\mathbf{a}_t = (v_t^{line}, v_t^{angle})^T$ . At each time  $t$ , the DRL policy outputs an action  $\mathbf{a}_t$  from the action set  $\mathcal{A}$ .

A fundamental functionality of an autonomous robot is to avoid the obstacles and reach a goal area. On every state  $\mathbf{x}_t$ , the sensory input  $o_t^i$  can be utilised to e.g., predict the distance to the obstacles when they are within the sensing range. To implement the functionality, the environment may impose a reward function  $\mathcal{R}$  on the states, the actions or both. A reward on the states can be, e.g., with respect to the distance to obstacles, and a reward on the actions can be, e.g., with respect to the acceleration in linear or angular speeds.

### B. Reachability Analysis and Verification

Reachability analysis has been developed recently for the verification of safety and robustness of DNNs [16]–[19]. Adapted to the context described in Section II-A, reachability analysis determines the set of states that a system can reach, starting from a set of initial states and considering the interaction between the DRL policy and the MDP. Safety verification, which is to determine whether a given DRL policy may lead to any unsafe state over an MDP, can be reduced to the reachability problem of whether an unsafe state is reachable.

In this paper, we verify a safety property on a model-free DRL algorithm by computing its reachable set over a full trajectory. Similar to POLAR [11], the following two mathematics are employed to calculate the reachable set: Taylor Arithmetic and Bernstein Polynomial.

1) *Taylor Arithmetic*: Followed by [11], [20], [21], any interval could be transferred into a Taylor model. A Taylor model is combined by a polynomial approximation  $p$  and an interval error bound  $I$ :

$$TM = p(\mathbf{x}) + I, \mathbf{x} \in D \quad (2)$$

where  $D$  is the input domain of the Taylor model.  $I$  is the remainder of the Taylor model.

Given two Taylor Models:  $TM_1 = (p_1, I_1)$  and  $TM_2 = (p_2, I_2)$ , the addition and multiplication are computed as:

$$TM_1 + TM_2 = (p_1 + p_2, I_1 + I_2)$$

$$TM_1 \times TM_2 = (p_1 \times p_2 - r_k, I_1 \times I_2 + Int(p_1) \times I_2 + Int(p_2) \times I_1 + Int(r_k))$$

where  $Int(\cdot)$  is the interval of the polynomials,  $k$  is the maximum order of the Taylor models, and  $r_k$  is the truncated polynomial of the Taylor models.

2) *Bernstein Polynomial*: There are several different activation functions in DNNs, e.g., Sigmoid, ReLU, and Tanh. The Taylor model can only propagate with continuous activation functions, but DNNs may use piece-wise activation functions to fit the data, such as the ReLU function. Inspired by [11], the Bernstein polynomial can be applied to calculate the Taylor models with discrete activation functions.

To achieve over-approximation for safety, the conservative remainder for Bernstein polynomial should be considered:

$$p_\sigma = \sum_{i=0}^k \left( \sigma \left( a + \frac{b-a}{k} i \right) C_i^k \frac{(y_i - a)^k (b - y_i)^{k-i}}{(b-a)^k} \right) \quad (3)$$

where  $a, b$  are the infimum and supremum of the activation function input,  $k$  is the maximum order of the Bernstein polynomial,  $y$  is the sampled point between  $a$  and  $b$ .

$$\epsilon = \max_{i=0, \dots, m} \left( \left| p_\sigma \left( \frac{b-a}{m} \left( i + \frac{1}{2} \right) + a \right) - \sigma \left( \frac{b-a}{m} \left( i + \frac{1}{2} \right) + a \right) \right| + \frac{b-a}{m} \right) \quad (4)$$

where  $m$  is the sampling steps and  $\epsilon$  is the conservative error bound of the Bernstein polynomial  $I_\sigma = [-\epsilon, +\epsilon]$ .

### C. Operational Profile based Reliability Assessment

In real world scenarios, the agent can take different trajectories from different initial states under a given policy and an operational environment. To support reliability claims for DRL in safety critical applications, all possible trajectories need to be considered. In real applications, the initial state (and its trajectory) of the agent usually obeys some distribution that can be approximated from data.

The OP has been widely modelled in reliability assessment, which is applied to represent the occurrence probabilities of function calls and the distributions of parameter values [22], [23]. In other words, the OP is a Probability Density Function over the whole input domain  $D$ , and returns the probability of  $\mathbf{x} \in D$  being selected as the next random input. Later, we formally define the reliability metric of a DRL policy in a given environment, considering the OP.

## III. ALGORITHM DESIGN

In this paper, we design a novel two-level framework for assessing the reliability of DRL controlled Robotics and Autonomous Systems (RAS) based on the reachability verification tools and statistical analysis technologies. At the local-level, safety verification is reduced to a reachability problem of checking whether an unsafe state is reachable from an initial state, while the global-level claims on reliability can be obtained by OP-based statistical methods.

### A. Local-level Safety Verification

At the local level, we introduce an interval based method to calculate the reachable set of a DRL policy. DDPG algorithm is composed of an actor network and a critic network. We reduce the verification problem to a reachability problem, which is to calculate the overall reachable set of actor neural network and environment, cf. Fig. 1.

The observation from the environment is normally noisy due to, e.g., inaccurate sensor signal and external disturbance. These noises could be ignored under safe conditions and in wide spaces but will cause safety issues in some corner cases, as illustrated in subfigures (A-D) of Fig. 2. It is obvious that the UUV is safe in case A because the real paths and observable path are safe. In cases B, C, and D, the UUV is

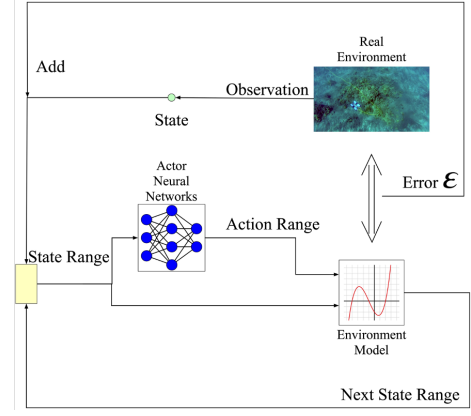


Fig. 1: Reachability verification framework.

unsafe since at least one signal shows that a crash occurs. If the sensor did not observe the correct distance perception in a corner case, the decision of the current policy will lead to a crash with high probability. Consequently, these errors should also be considered in safety verification. Following the recommendation from [24], the interval methods are suitable to deal with the uncertain sensor noises. Based on the reachability verification tools, we use an interval to bound the real paths around the observable path. Subfigure (E) in Fig. 2 shows that all the real paths are bounded in green intervals.

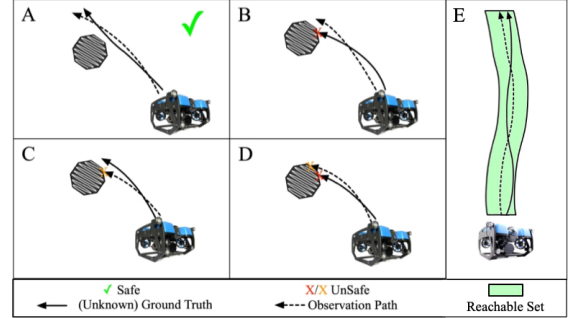


Fig. 2: Different corner cases and illustration diagram.

Specifically, the robot's initial states are over-approximated with a range  $\eta$  and all possible true states are theoretically bounded in this green interval. Then we have

$$\mathbf{a}_t \in NN_{actor}(\mathbf{x}), \quad \mathbf{x} \in \eta \quad (5)$$

where  $NN_{actor}$  is the actor neural network of DDPG. Transferring the initial input domain into a Taylor model:

$$TM_i^0 = (p_i(x_i), I_i), \quad x_i \in \eta_i \quad (6)$$

where  $x_i$  is the  $i$ th state value of the observation. There is no activation function after the first layer of the neural network. The Taylor model can be directly propagated to the next layer.

$$tm_i^j = \sum_{k=1}^N \mathbf{w}_k \times TM_k^{j-1} + \mathbf{b}_i \quad (7)$$

where  $tm_i^j$  is the temporary Taylor model that has not passed the activation functions.  $\mathbf{w}, \mathbf{b}$  are the weights and biases of  $NN_{actor}$ , respectively.  $j, k$  are the indices of neural network layers and neurons,  $N$  is the total number of neurons.

For the rest layers of  $NN_{actor}$ , there are activation functions. Meanwhile, Taylor arithmetic cannot process the interval input with the noncontinuous activation functions. Therefore, we employ the Bernstein polynomial to deal the activation functions. Although the transformation will yield errors, we can summarise these errors into the remainder of the Taylor models by equations (3) and (4).

$$TM_i^j = p_\sigma \left( \sum_{k=1}^N \mathbf{w}_k \times TM_k^{j-1} + \mathbf{b}_i \right) + Int(r_k) + I_\sigma, \quad j > 0 \quad (8)$$

It is noticed that the maximum order of the Taylor models will be increased after the activation functions, and therefore we truncate the part polynomial  $r_k$  of the output Taylor model to maintain the complexity of the Taylor models. After several layers propagation, the output Taylor model is

$$TM_i^{out} = \sum_{k=1}^N \mathbf{w}_k \times TM_k^{out-1} + \mathbf{b}_i \quad (9)$$

Here, the output Taylor model could be reverse to the interval by calculating the upper and lower bound of the Taylor model regarding to the input domain  $D$ .

*a) POLAR Algorithm Optimisation:* In this paper, POLAR algorithm is selected to test DRL algorithm due to its advantages of tighter remainder bounds [11]. Different from other interval arithmetic methods, POLAR algorithm generate better performance in reachable set calculation, especially for DNNs. The deeper neural network layers cost POLAR longer time to do Bernstein polynomial sampling. In consideration of the characteristics of ReLU activation function, the propagation law of the Taylor models is separated into three parts, which depends on the interval ranges  $TM \in [a, b]$ .

Furthermore, it is time consuming to calculate the accurate bound of the high order Taylor models. Similar to [11], [25], we calculate the conservative bound for the Taylor models with the Minkowski addition [26]:

$$P \oplus Q = \{\mathbf{p} + \mathbf{q} | \mathbf{p} \in P, \mathbf{q} \in Q\} \quad (10)$$

Consequently, to accelerate the testing speed and further enhance the testing accuracy of ReLU activation functions, we define the following propagation law for the Taylor models:

$$TM^o = \begin{cases} 0, & \text{if } b \leq 0, \\ p_\sigma(TM^i) + Int(r_k) + I_\sigma, & \text{if } a \leq 0 \ \& \ b \geq 0, \\ TM^i, & \text{if } a \geq 0, \end{cases} \quad (11)$$

where  $TM^i$  and  $TM^o$  are the Taylor models before and after the Relu activation function, respectively.

We enhance the original POLAR algorithm by first splitting the polynomial fit on each neuron into three piecewise functions based on the characteristics of the ReLU function, and then using the Bernstein polynomial to calculate the TMs if and only the interval of TMs contains 0. It is obvious that

the proposed approach (11) is much faster than the original POLAR algorithm since the sampling steps of the Bernstein polynomial are excused when the lower and upper bounds of TMs are positive or negative simultaneously. Nevertheless, the proposed method (11) can get the results without error (when the interval is positive-definite) or even eradicate the errors passed from the previous layer to zero (when the interval is negative-definite). Therefore, this optimisation step can reduce the computation time and increase accuracy.

## B. Global-level Reliability Assessment

The execution of a DRL-driven RAS in an environment leads to a trajectory distribution (modelled as a Discrete-Time Markov Chain, discussed later), where the uncertainty (modelled with probability distribution) is from the environment<sup>1</sup>. Formally, given an environment  $E$ , a policy  $\pi$ , and an initial state  $\mathbf{x}_0$ , we can construct a model  $\mathcal{M}^E(\pi, \mathbf{x}_0)$  representing the probability distribution of a set of trajectories. Assume that we have a verification technique  $g$ , as discussed in Sections III-A.

*Definition 1:* The verification problem is, given a constructed model  $\mathcal{M}^E(\pi, \mathbf{x}_0)$  and a verification tool  $g$ , to determine whether the model is safe with respect to certain property  $\phi$ , written as  $\mathcal{M}^E(\pi, \mathbf{x}_0) \models^g \phi$ . We may omit the superscript  $g$  and write  $\mathcal{M}^E(\pi, \mathbf{x}_0) \models \phi$ , if it is clear from the context. We can also assume that  $g$  returns a probability value – a Boolean answer can be converted into a Dirac probability. Then, the verification problem is to compute  $Pr(\mathcal{M}^E(\pi, \mathbf{x}_0), \phi)$ , i.e., the probability of safety.

In the following, we discuss how the above verification problem may contribute to the computation of the reliability. Similar as [28], [29], we partition the space of initial states into  $m$  sets, each of which is represented as a constraint  $\mathcal{C}_i$ , for  $i = 1..m$ . Then, we can also define the empirical distribution  $p_{op}$  over the partitions, and find a model  $G_\theta$  that is as close as possible to  $p_{op}$ . Formally, assume that  $G_\theta$  is a generative model over parameters  $\theta$ , we have

$$\theta^* = \arg \min_{\theta} \text{KL}(G_\theta, p_{op}) \quad (12)$$

where  $\text{KL}(\cdot, \cdot)$  is the KL divergence between two distributions.

Based on these, we can estimate the reliability (defined as the probability of failure in satisfying  $\phi$  with the policy  $\pi$  in the environment  $E$ ) as

$$\text{Reliability}(E, \pi, \phi) = \sum_{i=1}^m G_\theta(\mathcal{C}_i) (1 - Pr(\mathcal{M}^E(\pi, \mathbf{x}_{\mathcal{C}_i}), \phi)) \quad (13)$$

where  $G_\theta(\mathcal{C}_i)$  returns the probability density of the partition  $i$  that is represented as the constraint  $\mathcal{C}_i$ ,  $\mathbf{x}_{\mathcal{C}_i}$  denotes the central point (i.e., a representative) of  $\mathcal{C}_i$ , and  $1 - Pr(\mathcal{M}^E(\pi, \mathbf{x}_{\mathcal{C}_i}), \phi)$  returns the failure rate of the DRL agent  $\pi$  working on inputs satisfying the constraint  $\mathcal{C}_i$  under the environment  $E$ . Note that,  $G_\theta$  can be estimated in the same way as the data distribution in [28], [29].

<sup>1</sup>For simplicity and without loss of generality, we assume DRL policy is deterministic, while our method can be adapted for probabilistic policies.

TABLE I: Connections between reliability assessment for traditional software, DL classifiers and DRL controllers.

	Traditional safety critical software	DL classifiers	<b>DRL controllers</b>
Metric	Probability of failure per random demand ( <i>pdf</i> )	Probability of misclassification per random image ( <i>pmi</i> )	<b>Probability of crash per random initial state</b>
OP	Distribution of independent demands	Distribution of independent images	<b>Distribution of independent initialisations</b>
Event of interest	Failure that has safety impact	Misclassification that has safety impact	<b>Failure that leads to hazards, e.g., crash</b>
Partitions	Classes of input demands (“bins”) based on functionalities	Norm-balls in the input space with certain radius (e.g., the <i>r</i> -separation distance [27])	<b>Norm-balls in the input space with certain radius representing a set of initial states</b> (e.g., a specified bound of errors)
V&V each partition	Estimation on conditional <i>pdf</i> of each bin (e.g., by stress testing of a “bin”)	Local robustness estimation within a given norm-ball	<b>Reachability verification</b> (a “strip” of bounded trajectories in the input space starting with a norm-ball)
Oracle of each partition	By observation or given by the specification	Label of the central-point (seed) of the norm-ball	<b>Emptiness of the intersection of the “strip” and predefined unsafe-areas</b>

To better position the scope of our work, we summarise our approach for DRL, compared with reliability assessment methods designed for traditional software and DL classifiers, in Table I, where the proposed method is written in bold font.

#### IV. EXPERIMENT RESULTS

This section presents experimental results regarding the following research questions (RQs):

**RQ1:** How effective the proposed local-level reliability assessment is of a single trajectory (reachability verification)?

**RQ2:** How accurate the proposed method is compared to original POLAR algorithm?

**RQ3:** How conservative it is between our interval-based method and traditional sampled point-based method?

In the following, we will first introduce our experimental environment, and then address the **RQs** individually.

##### A. Experiment Setup

The mission of this experiment example is to automatically dock the UUV based on DRL algorithm. We have both simulation and physical experiments for Bluerov2 UUVs<sup>2</sup>. For the simulation environment, we use ROS [30] and Gazebo [31] as shown in the left side of Fig. 3.

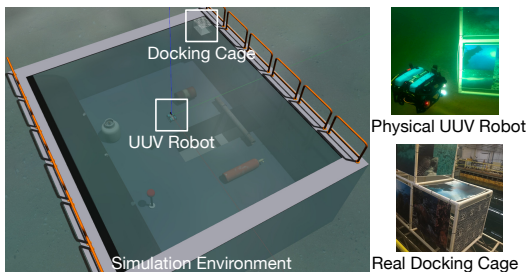


Fig. 3: Simulation and physical experiment environments.

For the physical experiment environment, the training process is conducted on a real-world water tank with a docking cage as shown in the right side of Fig. 3. In the real-world experiment environment, we randomly initialise the UUV from different start points, which naturally generate the

OP in our reliability assessment framework. Theoretically, we will accurately estimate the model’s safety if the verified all initial intervals can cover whole initial space. Due to the limitation of computing power and the scalability problem, we use the sampling method to estimate the safety of the entire input space in this paper. It can be obtained from Fig. 4, the distribution of the approximated OP converges as the number of samples increases.

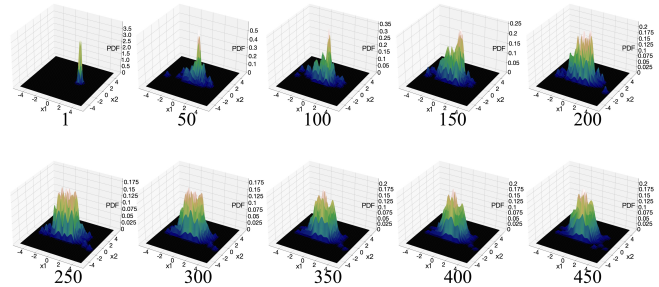


Fig. 4: Approximating the OP as more data is sampled. Multivariate distribution is projected to a 2D space for visualisation.

##### B. Verification of an Initial State

In this subsection, we show the local-level reachability verification process of an initial interval. In this paper, we use POLAR [11] as the reachability verification tool to over-approximate the reliability of an initial state. Here, we set a deadzone  $([-0.1, +0.1])$  around the target to accelerate the UUV’s stability and improve the missions’ success rate. It can significantly solve the scalability problem of the verification tool and the over-conservative problem caused by the error accumulation. The verification results are shown in Fig. 5.

From the figure, the reachable sets of the robot converges to 0, which means the robot from that initial state interval will reach its destination. Due to the existence of the deadzone, if the variables tend to the target value, then all the variables will eventually converge to the deadzone, that is, the UUV will eventually be parallel to the dock frame. Note that state variables 3 (roll velocity) and 4 (pitch velocity)

<sup>2</sup>All source code, DRL models, datasets, and experiment results are available at solitude website <https://github.com/Solitude-SAMR>

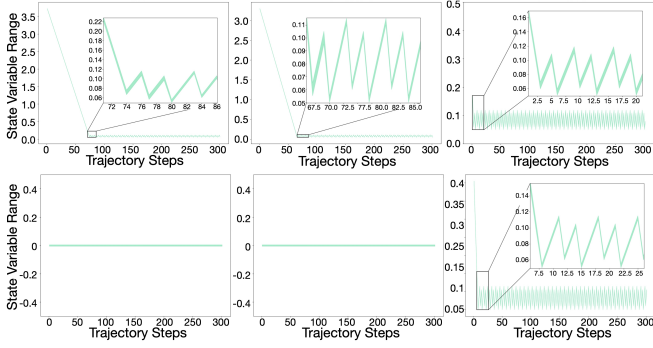


Fig. 5: The reachable set of six state variables (Top 3 figures: x, y, z velocity; Bottom 3 figures: roll, pitch, yaw velocity). Each plot shows a “strip” of bounded trajectories.

are not changed ideally since they are uncontrollable by the DRL policy in this experiment.

### C. Comparison with Original POLAR Algorithm

Here, we show the over-approximated range between our method and POLAR [11] to illustrate the advantages of the proposed algorithm. The experiments are performed using Python on a same computer equipped with a AMD core *EPYC* 7452. The initial states, system dynamics, neural network models and weights are same. In the experiment, we found that the POLAR algorithm is too loose after iterating 50 steps. To better show the performance, we update the input state interval on every single steps. The output Q ranges of each step in a trajectory are compared in Fig. 6.

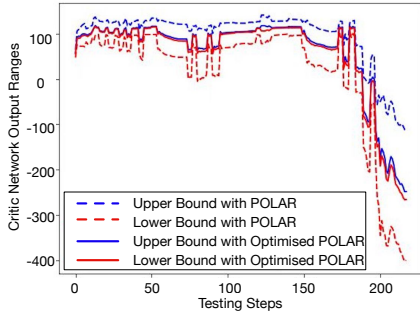


Fig. 6: Comparisons between the original POLAR and our optimised POLAR—the latter yields tighter bounds.

In Fig. 6, our optimisation for the original POLAR greatly improves its performance. Although both algorithms can obtain the reachable set of the DNNs, the proposed approach yields tighter thus more accurate results.

### D. Comparison with Point-based Assessment

After the local-level verification, we show the reliability assessment of different DRL policies to validate the effectiveness of the proposed framework in this subsection. For a given DRL policy and a set of initial spaces, we can estimate the reliability of the system based on equation (13) and reachability verification tools. In this paper, we collected the data from the real world water tank environment and trained

a DRL model to finish the mission task. After we sampled 500 initial states in the real-world environment, we calculate the reachable set by the proposed method and estimate the reliability of the UUV system between our interval-based method and traditional point-based methods, where we use the same OP on the global-level assessment. The comparison results are summarised in Fig. 7:

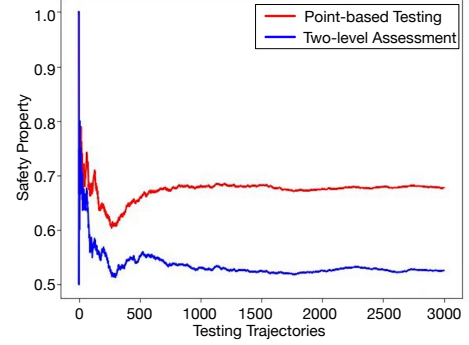


Fig. 7: Comparisons between point-based and our two-level interval-based assessments—the latter yields more conservative results.

As can be seen from the Fig.7, the proposed interval-based approach is more conservative than the point-based testing method. This is because the point-based method is equivalent to a sample from our interval. Therefore, the proposed local-level reachability verification algorithm considers the entire environment and is an over-approximation of all possible points. The result of original POLAR algorithm is not shown here, because the result is too conservative, where the reachable set covers all safe and unsafe area.

It can be seen from the Fig. 7, as the number of sampled initial states (representing trajectories starting from them) increases, the predicted reliability of UUV system converges. We found that the reliability of the system tested here is not very high. This is mainly due to the fact that safety factors are not considered in the training of the DRL model. It is conceivable that the goal of the reinforcement learning model is to dock the UUV at the harbour, which will cause the UUV always attempt to approach the destination in a straightforward route to achieve maximum reward and speed.

## V. CONCLUSION

This paper studies the probability of failures that can cause hazards in DRL controlled RASs. A two-level reliability assessment framework is proposed, using reachability verifications at the local-level and statistically supporting a probabilistic reliability claims based on the OP at the global-level. An optimisation on the local-level reachability analysis algorithm is applied to enhance the verification speed and accuracy. The results in the simulation and the real world manifest the effectiveness of the proposed framework.

## ACKNOWLEDGMENT

We thank Vibhav Bharti for his support in the experiments.

## REFERENCES

- [1] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yagamani, and P. Pérez, “Deep reinforcement learning for autonomous driving: A survey,” *IEEE Transactions on Intelligent Transportation Systems*, 2021.
- [2] J. Chen, S. E. Li, and M. Tomizuka, “Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning,” *IEEE Transactions on Intelligent Transportation Systems*, 2021.
- [3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.
- [4] F. Ye, S. Zhang, P. Wang, and C.-Y. Chan, “A survey of deep reinforcement learning algorithms for motion planning and control of autonomous vehicles,” in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 1073–1080.
- [5] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, “Robust deep reinforcement learning with adversarial attacks,” in *Proceedings of the 17th International Conference on Autonomous Agents and Multi-Agent Systems*, ser. AAMAS '18. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2018, p. 2040–2042.
- [6] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, “Safe exploration in continuous action spaces,” *arXiv preprint arXiv:1801.08757*, 2018.
- [7] F. Berkenkamp, “Safe exploration in reinforcement learning: Theory and applications in robotics,” Ph.D. dissertation, ETH Zurich, 2019.
- [8] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, “Safe reinforcement learning via shielding,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [9] S. Webb, T. Rainforth, Y. W. Teh, and M. P. Kumar, “A statistical approach to assessing neural network robustness,” in *7th Int. Conf. on Learning Representations (ICLR'19)*. OpenReview.net, 2019.
- [10] B. Wang, S. Webb, and T. Rainforth, “Statistically robust neural network classification,” in *Proc. of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, ser. PMLR, vol. 161 of UAI'21, 2021, pp. 1735–1745.
- [11] C. Huang, J. Fan, X. Chen, W. Li, and Q. Zhu, “Polar: A polynomial arithmetic framework for verifying neural-network controlled systems,” in *the 20th Int. Symp. on Automated Technology for Verification and Analysis*, 2022.
- [12] 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'16*, 2016.
- [13] S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.
- [14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [15] Robotis, “Robotis(2019) turtlebot3 – e-manual, waffle pi,” [Online] <https://manual.robotis.com/docs/en/platform/turtlebot3/overview/>. (Accessed on 02 August 2021).
- [16] W. Ruan, X. Huang, and M. Kwiatkowska, “Reachability analysis of deep neural networks with provable guarantees,” in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, ser. IJCAI'18. AAAI Press, 2018, p. 2651–2659.
- [17] C. Huang, J. Fan, W. Li, X. Chen, and Q. Zhu, “Reachnn: Reachability analysis of neural-network controlled systems,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–22, 2019.
- [18] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, “A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability,” *Computer Science Review*, vol. 37, p. 100270, 2020.
- [19] M. Althoff, “Reachability analysis and its application to the safety assessment of autonomous cars,” Ph.D. dissertation, Technische Universität München, 2010.
- [20] K. Makino and M. Berz, “Taylor models and other validated functional inclusion methods,” *International Journal of Pure and Applied Mathematics*, vol. 6, pp. 239–316, 2003.
- [21] 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 *Int. Conf. on Computer Aided Verification*. Springer, 2021, pp. 249–262.
- [22] J. D. Musa, “Operational profiles in software-reliability engineering,” *IEEE Software*, vol. 10, no. 2, pp. 14–32, 1993.
- [23] C. Smidts, C. Mutha, M. Rodriguez, and M. J. Gerber, “Software testing with an operational profile: Op definition,” *ACM Comput. Surv.*, vol. 46, no. 3, 2014.
- [24] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare, “Deep reinforcement learning at the edge of the statistical precipice,” *Advances in neural information processing systems*, vol. 34, pp. 29 304–29 320, 2021.
- [25] G. Yang, G. Qian, P. Lv, and H. Li, “Efficient verification of control systems with neural network controllers,” in *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, 2019, pp. 1–7.
- [26] P. Gritzmann and B. Sturmfels, “Minkowski addition of polytopes: computational complexity and applications to gröbner bases,” *SIAM Journal on Discrete Mathematics*, vol. 6, no. 2, pp. 246–269, 1993.
- [27] Y.-Y. Yang, C. Rashtchian, H. Zhang, R. R. Salakhutdinov, and K. Chaudhuri, “A closer look at accuracy vs. robustness,” *Advances in neural information processing systems*, vol. 33, pp. 8588–8601, 2020.
- [28] Y. Dong, W. Huang, V. Bharti, V. Cox, A. Banks, S. Wang, X. Zhao, S. Schewe, and X. Huang, “Reliability assessment and safety arguments for machine learning components in system assurance,” *ACM Tran. on Embedded Computing Systems*, 2022.
- [29] X. Zhao, W. Huang, A. Banks, V. Cox, D. Flynn, S. Schewe, and X. Huang, “Assessing the Reliability of Deep Learning Classifiers Through Robustness Evaluation and Operational Profiles,” in *AI Safety'21 Workshop at IJCAI'21*, 2021.
- [30] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, “Ros: an open-source robot operating system,” in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.
- [31] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 3. IEEE, 2004, pp. 2149–2154.