Counterexample-Guided Repair of Reinforcement Learning Systems Using Safety Critics

David Boetius [0000-0002-9071-1695] and Stefan Leue [0000-0002-4259-624X]

University of Konstanz, 78457 Konstanz, Germany {david.boetius,stefan.leue}@uni-konstanz.de

Abstract. Naïvely trained Deep Reinforcement Learning agents may fail to satisfy vital safety constraints. To avoid costly retraining, we may desire to repair a previously trained reinforcement learning agent to obviate unsafe behaviour. We devise a counterexample-guided repair algorithm for repairing reinforcement learning systems leveraging safety critics. The algorithm jointly repairs a reinforcement learning agent and a safety critic using gradient-based constrained optimisation.

Keywords: Reinforcement Learning · Safety · Repair.

1 Introduction

Deep Reinforcement Learning is at the core of several recent breakthroughs in AI [13,23,26]. With the increasing abilities of reinforcement learning agents, it becomes vital to effectively constrain such agents to avoid harm, particularly in safety-critical applications. A particularly effective class of constraints are formal guarantees on the non-occurrence of undesired behaviour (safety). Such guarantees are obtainable through formal verification [22].

Counterexample-guided repair is a successful iterative refinement algorithm for obtaining formally verified deep neural networks in supervised learning [6]. The repair algorithm alternates searching counterexamples and modifying the model under repair to remove counterexamples. A central component of the algorithm is a function that quantifies the safety of a neural network output. In reinforcement learning, the safety of an output depends on the interaction with the environment. Therefore, quantifying the safety of an output is expensive, making it challenging to apply counterexample-guided repair to reinforcement learning. To overcome this challenge, we propose to learn a safety critic [5,18,31] to quantify safety. Since safety critics are themselves imperfect machine learning models, we propose to repair the safety critic alongside the actual reinforcement learning agent.

In the spirit of actor-critic reinforcement learning algorithms [33], a safety critic learns to predict the safety of a state from gathered experience [5]. The idea of safety critics is analogous to the widely used [24,29] concept of value critics that learn to predict the value of a state from experience. We can use recorded unsafe trajectories, coupled with safe trajectories, as a starting point

for learning a safety critic. Since our aim is to *repair* a reinforcement learning agent, we can assume that the agent was already found to behave unsafely and, therefore, unsafe trajectories are available.

When using a safety critic in counterexample-guided repair, it is vital that the safety critic correctly recognises new counterexamples as unsafe. Otherwise, counterexample-guided repair can not proceed to repair the reinforcement learning agent. To achieve this, we propose to iteratively repair the safety critic alongside the reinforcement learning agent, such that the safety critic correctly recognises newly found counterexamples.

Our approach allows for great generality regarding the environments in which an agent operates. Similarly, our approach does not restrict the class of safety specifications that can be repaired. It solely requires the ability to falsify or verify the specification given an agent and an environment. Even in the absence of a falsifier or verifier, our approach can be used to repair a reinforcement learning agent whenever unsafe behaviour is detected.

The following section reviews the literature relevant to this paper. Following this, we formally introduce safe reinforcement learning, safety critics, and counterexample-guided repair. The subsequent section describes our approach of counterexample-guided repair using safety critics in more detail and rigour. We conclude with an outlook on future work. This includes an experimental evaluation.

2 Related Work

Reinforcement learning is the process by which an agent learns to solve a task by repeatedly interacting with an environment. The agent leans by maximising the return it receives during the interaction. State-of-the-art algorithms for reinforcement learning include Advantage Actor-Critic (A2C) [24], Asynchronous Advantage Actor-Critic (A3C) [24], and Proximal Policy Optimisation (PPO) [29]. These algorithms are based on deep neural networks as reinforcement learning agents. Due to the prevalence of deep neural networks in state-of-the-art reinforcement learning methods, this paper is primarily concerned with deep reinforcement learning. However, our approach is not limited to this class of models.

In safe reinforcement learning, the agent must also respect additional safety constraints. An overview of safe reinforcement learning is presented in [15]. More recent developments include shielding [1], safe reinforcement learning using abstract interpretation [20,32], and safe reinforcement learning via safety critics [5,18,31,39]. In contrast to safe reinforcement learning, the repair of reinforcement learning agents is concerned with making an existing reinforcement agent safe. In this paper, we apply safety critics to repairing reinforcement learning agents.

Verification of reinforcement learning systems is concerned with proving that a reinforcement learning agent behaves safely. Recent approaches for reinforcement learning verification build upon reachability analysis [3,19,37] and model checking [2,3,12]. A survey of the field is provided by [22].

In the domain of supervised learning, machine learning models are verified using Satisfiability Modulo Theories (SMT) solving [11,17,21], Mixed Integer Linear Programming (MILP) [36], and Branch and Bound (BaB) [7,14,38,40]. Many repair algorithms for supervised machine learning models are based on counterexample-guided repair [6]. The approaches for removing counterexamples range from augmenting the training set [28,34] and constrained optimisation [4,16] to specialised neural network architectures [8,16]. Non-iterative repair approaches for neural networks include [30,35].

3 Preliminaries

This section introduces Safe Reinforcement Learning, Safety Critics, Verification, Falsification, and Repair. While our algorithm's envisioned target is deep reinforcement learning agents (neural networks), our approach is not specific to a particular model class.

3.1 Safe Reinforcement Learning

Following [5], we adopt a perspective on safe reinforcement learning where unsafe behaviour may occur during training but not when an agent is deployed. Our assumption is that we can train the agent to behave safely in a simulation where unsafe behaviour is inconsequential. A safe reinforcement learning task is formalised as a Constrained Markov Decision Process (CMDP). We consider CMDPs with deterministic transitions.

Definition 1 (CMDP). A Constrained Markov Decision Process (CMDP) with deterministic transitions is a tuple (S, A, P, R, S_0, C) , where S is the state space, A is the set of actions, $P: S \times A \to S$ is the transition function, $R: S \times A \to \mathbb{R}$ is the reward, $C \subset S$ is a set of safe states, and $S_0 \subset C$ is a set of initial states.

For a finite time-horizon $T \in \mathbb{N}$, $\tau = s_0, a_0, s_1, a_1, \ldots, s_T$ is a trajectory of a CMDP if $s_0 \in \mathcal{S}_0$ and $s_t = P(s_{t-1}, a_{t-1}), \forall t \in \{1, \ldots, T\}$.

Definition 2 (Return). Given a discount factor $\gamma \in [0,1]$, the return G of a trajectory $\tau = s_0, a_0, s_1, a_1, \ldots, s_T$ is

$$G(\tau) = \sum_{t=0}^{T-1} \gamma^t R(s_t, a_t).$$

Definition 3 (Safe Trajectories). For a set of safe states C, a trajectory $\tau = s_0, a_0, s_1, a_1, \ldots, s_T$ is safe if $s_0, s_1, \ldots, s_T \in C$. We write $\tau \models C$ if τ is safe.

Assuming a uniform distribution $\mathcal{U}(\mathcal{S}_0)$ over the initial states, our goal is to learn a (deterministic) parametric policy $\pi_{\theta}: \mathcal{S} \to \mathcal{A}$ that maximises the expected return while maintaining safety

maximise
$$\mathbb{E}_{s_0 \sim \mathcal{U}(S_0)}[G(\tau(s_0, \pi_{\theta}))]$$

subject to $\tau(s_0, \pi_{\theta}) \models \mathcal{C} \quad \forall s_0 \in S_0,$ (1)

where $\tau(s_0, \pi_{\theta}) = s_0, a_0, s_1, a_1, \dots, s_T$ is a trajectory with $a_t = \pi_{\theta}(s_t), \forall t \in \{0, \dots, T-1\}.$

A parametric policy may be given, for example, by a neural network net_{θ} : $\mathbb{R}^n \to \mathcal{A}$ reading a numeric representation of a state $\mathbf{x}_s \in \mathbb{R}^n$, $n \in \mathbb{N}$, $s \in \mathcal{S}$ and returning an action $a \in \mathcal{A}$. In this paper, we use the terms policy and agent interchangeably.

3.2 Safety Critics

Safety critics learn the safety value function $V_{\mathcal{C}}^{\pi\theta}: \mathcal{S} \to \mathbb{R}$

$$V_{\mathcal{C}}^{\pi_{\boldsymbol{\theta}}}(s) = \min_{t \in \{0, \dots, T\}} c(s_t), \tag{2}$$

where $s_0 = s$, $s_t = P(s_{t-1}, \pi_{\theta}(s_{t-1}))$, $\forall t \in \{1, ..., T\}$, and $c : S \to \mathbb{R}$ is a satisfaction function [4] or robustness measure [9] for the safe set C.

Definition 4 (Satisfaction Function). A function $c: \mathcal{S} \to \mathbb{R}$ is a satisfaction function of a set $\mathcal{C} \subseteq \mathcal{S}$ if $\forall s \in \mathcal{S}: c(s) \geq 0 \Leftrightarrow s \in \mathcal{C}$.

The concept of a safety critic is analogous to (value) critics in actor-critic reinforcement learning [33]. Classical (value) critics learn the *value* of a state $V^{\pi\theta}$. The value is the expected return when starting in a state. Safety critics can be learned using the methods from [5,31,39].

3.3 Verification, Falsification, and Repair

Given a CMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \mathcal{S}_0, \mathcal{C})$ and a policy π_{θ} , we are interested in the question whether the policy guarantees safety for all initial states. A *counterex-ample* is an initial state for which following the policy leads to unsafe states.

Definition 5 (Counterexample). Given a policy π_{θ} , a counterexample is an initial state $s_0 \in \mathcal{S}_0$, such that the trajectory $\tau = s_0, a_0, s_1, a_1, \ldots, s_T, T \in \mathbb{N}$ with $a_t = \pi_{\theta}(s_{t-1})$, $\forall t \in \{1, \ldots, T-1\}$ contains an unsafe state $s_t \notin \mathcal{C}$ for some $t \in \{1, \ldots, T\}$.

Since counterexamples lead to unsafe states, the safety value function $V_{\mathcal{C}}^{\pi\theta}$ of a counterexample is negative.

When considering algorithms for searching counterexamples, we differentiate between *falsifiers* and *verifiers*. While falsifiers are *sound* counterexample-search algorithms, verifiers are *sound* and *complete*.

Definition 6 (Soundness and Completeness). A counterexample-search algorithm is sound if it only produces genuine counterexamples. Additionally, an algorithm is complete if it terminates and produces a counterexample for every unsafe policy.

Proposition 1. A policy π_{θ} is safe whenever a verifier does not produce a counterexample for π_{θ} .

Proof. Proposition 1 follows from contraposition on the completeness of verifiers.

Given an unsafe policy π_{θ} , the task of *repair* is to modify the policy to be safe while maintaining high returns. A successful repair algorithm for supervised learning is counterexample-guided repair [4,6,34]. The following section introduces a counterexample-guided repair algorithm for reinforcement learning.

4 Counterexample-Guided Repair using Safety Critics

Existing counterexample-guided repair algorithms repair supervised learning models by alternating counterexample search and counterexample removal. Algorithm 1 describes the algorithmic skeleton of counterexample-guided repair. This skeleton is akin to all counterexample-guided repair algorithms.

```
Algorithm 1: Counterexample-Guided Repair
```

```
Input: CMDP \mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \mathcal{S}_0, \mathcal{C}), Policy \pi_{\theta}

1 \mathcal{S}^c \leftarrow \text{find counterexamples}(\mathcal{M}, \pi_{\theta})

2 do

3 | \theta \leftarrow \text{remove counterexamples}(\mathcal{S}^c, \pi_{\theta}, \mathcal{M})

4 | \mathcal{S}^c \leftarrow \mathcal{S}^c \cup \text{find counterexamples}(\mathcal{M}, \pi_{\theta})

5 while \exists s_0 \in \mathcal{S}^c : s_0 is counterexample
```

When using a verifier to find counterexamples, Algorithm 1 is guaranteed to produce a safe policy if it terminates [6]. However, Algorithm 1 is not generally guaranteed to terminate [6]. Despite this, counterexample-guided repair has proven successful in repairing deep neural networks [4] and other machine learning models [34].

Algorithm 1 has two sub-procedures we need to instantiate for obtaining an executable algorithm: finding counterexamples and removing counterexamples. For finding counterexamples, we can use tools for verifying reinforcement learning systems [2,3,12,19,37] (see [22] for a survey). In the remainder of this paper, we address removing counterexamples using safety critics.

4.1 Removing Counterexamples

Similarly to the supervised setting [6], removing counterexamples corresponds to solving a constrained optimisation problem

maximise
$$\mathbb{E}_{s_0 \sim \mathcal{U}(\mathcal{S}_0)}[G(\tau(s_0, \pi_{\theta}))]$$

subject to $\tau(s_0, \pi_{\theta}) \models \mathcal{C} \quad \forall s_0 \in \mathcal{S}^c$, (3)

where $\tau(s_0, \pi_{\theta})$ is as in Equation (1) and \mathcal{S}^c is a finite set of counterexamples. In the supervised setting, we can remove counterexamples by directly solving the analogue of Equation (3) using gradient-based optimisation algorithms [4]. However, for repairing reinforcement learning policies, checking whether a set of parameters θ is feasible for Equation (3) is expensive, as it requires simulating the CMDP. Additionally, the term $\tau(s_0, \pi_{\theta}) \models \mathcal{C}$ suffers from exploding gradients [27] due to the repeated application of π_{θ} for obtaining the trajectory. These properties of Equation (3) hinder the application of gradient-based optimisation algorithms for removing counterexamples by solving Equation (3) directly.

To obtain an algorithm for removing counterexamples, we first note that Equation (3) can equivalently be reformulated using the safety value function $V_{\mathcal{C}}^{\pi_{\theta}}$ from Equation (2). Concretely, we can replace the constraint $\tau(s_0, \pi_{\theta}) \models \mathcal{C}$ by $V_{\mathcal{C}}^{\pi_{\theta}}(s_0) > 0$. Now, when approximating $V_{\mathcal{C}}^{\pi_{\theta}}$ using a safety critic $V_{\mathcal{C}}^{\pi_{\theta}}$, we obtain

$$\begin{array}{ll}
\text{maximise} & \mathbb{E}_{s_0 \sim \mathcal{U}(\mathcal{S}_0)}[G(\tau(s_0, \pi_{\boldsymbol{\theta}}))] \\
\text{subject to} & \widetilde{V}_{\mathcal{C}}^{\pi_{\boldsymbol{\theta}}}(s_0) \ge 0 \quad \forall s_0 \in \mathcal{S}^c,
\end{array} \tag{4}$$

While Equation (4) is not equivalent to Equation (3) due to using an approximation of $V_c^{\pi\theta}$, Equation (4) can be solved using techniques such as stochastic gradient descent/ascent [10] or the ℓ_1 penalty function method [4,25]. To ensure that solving Equation (4) actually removes counterexamples, we repair the safety critic $\widetilde{V}_c^{\pi\theta}$ alongside the policy.

4.2 Repairing Safety Critics

To allow us to remove counterexamples by solving Equation (4), the safety critic $\widetilde{V}_{\mathcal{C}}^{\pi_{\theta}}$ needs to correctly recognise the counterexamples in \mathcal{S}^{c} as counterexamples. By recognising a counterexample s_{0} as a counterexample, we mean that $\widetilde{V}_{\mathcal{C}}^{\pi_{\theta}}(s_{0}) < 0$. We can ensure that the safety critic recognises all counterexamples in \mathcal{S}^{c} by solving

minimise
$$J(\widetilde{V}_{\mathcal{C}}^{\pi_{\theta}})$$

subject to $\widetilde{V}_{\mathcal{C}}^{\pi_{\theta}}(s_0) < 0 \quad \forall s_0 \in \mathcal{S}^c \text{ with } V_{\mathcal{C}}^{\pi_{\theta}}(s_0) < 0,$ (5)

where $J(\widetilde{V}_{\mathcal{C}}^{\pi_{\theta}})$ is a loss function for training the safety critic [5,31,39]. Solving Equation (5) can itself be understood as removing counterexamples of the safety critic. As Equation (4), we can solve Equation (5) using stochastic gradient descent/ascent [10] or the ℓ_1 penalty function method [4,25].

Algorithm 2: Counterexample Removal

```
Input: CMDP \mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \mathcal{S}_0, \mathcal{C}), Unsafe policy \pi_{\boldsymbol{\theta}}, Safety Critic \widetilde{V}_{\mathcal{C}}^{\pi_{\boldsymbol{\theta}}}, Counterexamples \mathcal{S}^c 1 while \exists s_0 \in \mathcal{S}^c : s_0 is counterexample do

2 update \widetilde{V}_{\mathcal{C}}^{\pi_{\boldsymbol{\theta}}} by solving Equation (5)

3 update \pi_{\boldsymbol{\theta}} by solving Equation (4) using \widetilde{V}_{\mathcal{C}}^{\pi_{\boldsymbol{\theta}}}
```

4.3 Counterexample Removal Algorithm

We propose jointly modifying the safety critic and the unsafe reinforcement learning agent π_{θ} . Algorithm 2 summarises our approach. We first update the safety critic to recognise the available counterexamples. This corresponds to solving Equation (5). Using the updated safety critic, we update π_{θ} to remove the counterexamples by solving Equation (4).

Since the safety critic may fail to recognise a counterexample as a counterexample for the updated policy, we iterate the previous two steps until all counterexamples are removed.

In principle, this procedure may fail to terminate if the safety-critic "forgets" to recognise the counterexamples of the initial policy when being updated in the second iteration of Algorithm 2. Since the policy is updated in the first iteration of Algorithm 2, updating $\tilde{V}_{\mathcal{C}}^{\pi_{\theta}}$ in the second iteration does not consider counterexamples for the initial policy. Therefore, the policy may revert to its initial parameters in the second iteration to evade the updated safety critic. This leads to an infinite loop. However, this issue can be circumvented by including previous unsafe trajectories in Equation (5), similarly to how counterexamples are retained in Algorithm 1 for later iterations to counter reintroducing counterexamples.

5 Conclusion

We introduce a counterexample-guided repair algorithm for reinforcement learning systems. We leverage safety critics to circumvent costly simulations during counterexample removal. Our approach applies to a wide range of specifications and can work with any verifier and falsifier. The central idea of our approach is to repair the policy and the safety critic jointly.

Future work includes evaluating our algorithm experimentally and comparing it with abstract-interpretation-based safe reinforcement learning [20,32]. Since counterexample-guided repair avoids the abstraction error of abstract interpretation, we expect that counterexample-guided repair can produce less conservative, safe reinforcement learning agents. Additionally, our ideas are not inherently limited to reinforcement learning but can be applied whenever satisfaction functions are unavailable or costly to compute. Exploring such applications is another direction for future research.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

- Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: McIlraith, S.A., Weinberger, K.Q. (eds.) AAAI. pp. 2669–2678. AAAI Press (2018). https://doi.org/10.1609/AAAI.V32I1.11797
- Amir, G., Schapira, M., Katz, G.: Towards scalable verification of deep reinforcement learning. In: FMCAD. pp. 193–203. IEEE (2021). https://doi.org/10.34727/2021/ISBN.978-3-85448-046-4
- 3. Bacci, E., Giacobbe, M., Parker, D.: Verifying reinforcement learning up to infinity. In: Zhou, Z. (ed.) IJCAI. pp. 2154–2160. ijcai.org (2021). https://doi.org/10.24963/IJCAI.2021/297
- Bauer-Marquart, F., Boetius, D., Leue, S., Schilling, C.: SpecRepair: Counter-Example Guided Safety Repair of Deep Neural Networks. In: Legunsen, O., Rosu, G. (eds.) SPIN. Lecture Notes in Computer Science, vol. 13255, pp. 79–96. Springer (2022). https://doi.org/10.1007/978-3-031-15077-7 5
- 5. Bharadhwaj, H., Kumar, A., Rhinehart, N., Levine, S., Shkurti, F., Garg, A.: Conservative safety critics for exploration. In: ICLR. OpenReview.net (2021), https://openreview.net/forum?id=iaO86DUuKi
- Boetius, D., Leue, S., Sutter, T.: A robust optimisation perspective on counterexample-guided repair of neural networks. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) ICML. Proceedings of Machine Learning Research, vol. 202, pp. 2712–2737. PMLR (2023), https://proceedings.mlr.press/v202/boetius23a.html
- Bunel, R., Lu, J., Turkaslan, I., Torr, P.H.S., Kohli, P., Kumar, M.P.: Branch and Bound for Piecewise Linear Neural Network Verification. J. Mach. Learn. Res. 21, 42:1–42:39 (2020), http://jmlr.org/papers/v21/19-468.html
- 8. Dong, G., Sun, J., Wang, J., Wang, X., Dai, T.: Towards Repairing Neural Networks Correctly. In: QRS. pp. 714–725. IEEE (2021). https://doi.org/10.1109/QRS545444.2021.00081
- 9. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: Chatterjee, K., Henzinger, T.A. (eds.) FORMATS. Lecture Notes in Computer Science, vol. 6246, pp. 92–106. Springer (2010). https://doi.org/10.1007/978-3-642-15297-9_9
- Eban, E., Schain, M., Mackey, A., Gordon, A., Rifkin, R., Elidan, G.: Scalable Learning of Non-Decomposable Objectives. In: Singh, A., Zhu, X.J. (eds.) AIS-TATS. Proceedings of Machine Learning Research, vol. 54, pp. 832–840. PMLR (2017), http://proceedings.mlr.press/v54/eban17a.html
- 11. Ehlers, R.: Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks. In: D'Souza, D., Kumar, K.N. (eds.) ATVA. Lecture Notes in Computer Science, vol. 10482, pp. 269–286. Springer (2017). https://doi.org/10.1007/978-3-319-68167-2 19
- 12. Eliyahu, T., Kazak, Y., Katz, G., Schapira, M.: Verifying learning-augmented systems. In: Kuipers, F.A., Caesar, M.C. (eds.) SIGCOMM. pp. 305–318. ACM (2021). https://doi.org/10.1145/3452296.3472936

- Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov, A., R. Ruiz, F.J., Schrittwieser, J., Swirszcz, G., Silver, D., Hassabis, D., Kohli, P.: Discovering faster matrix multiplication algorithms with reinforcement learning. Nat. 610(7930), 47–53 (2022). https://doi.org/10.1038/s41586-022-05172-4
- 14. Ferrari, C., Mueller, M.N., Jovanović, N., Vechev, M.: Complete Verification via Multi-Neuron Relaxation Guided Branch-and-Bound. In: ICLR. OpenReview.net (2022), https://openreview.net/forum?id=l amHfloaK
- García, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. J. Mach. Learn. Res. 16, 1437–1480 (2015). https://doi.org/10.5555/2789272.2886795
- Guidotti, D., Leofante, F., Pulina, L., Tacchella, A.: Verification and Repair of Neural Networks: A Progress Report on Convolutional Models. In: AI*IA. Lecture Notes in Computer Science, vol. 11946, pp. 405–417. Springer (2019). https://doi.org/10.1007/978-3-030-35166-3 29
- Guidotti, D., Leofante, F., Tacchella, A., Castellini, C.: Improving reliability of myocontrol using formal verification. IEEE Trans. Neural Syst. Rehabilitation Eng. 27(4), 564–571 (2019). https://doi.org/10.1109/TNSRE.2019.2893152
- 18. Hans, A., Schneegaß, D., Schäfer, A.M., Udluft, S.: Safe exploration for reinforcement learning. In: ESANN. pp. 143–148 (2008), https://www.esann.org/sites/default/files/proceedings/legacy/es2008-36.pdf
- Ivanov, R., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verisig: verifying safety properties of hybrid systems with neural network controllers.
 In: Ozay, N., Prabhakar, P. (eds.) HSCC. pp. 169–178. ACM (2019). https://doi.org/10.1145/3302504.3311806
- Jin, P., Tian, J., Zhi, D., Wen, X., Zhang, M.: Trainify: A CEGAR-Driven Training and Verification Framework for Safe Deep Reinforcement Learning. In: Shoham, S., Vizel, Y. (eds.) CAV(1). Lecture Notes in Computer Science, vol. 13371, pp. 193–218. Springer (2022). https://doi.org/10.1007/978-3-031-13185-1 10
- 21. Katz, G., Barrett, C.W., Dill, D.L., Julian, K.D., Kochenderfer, M.J.: Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In: Majumdar, R., Kuncak, V. (eds.) CAV (1). Lecture Notes in Computer Science, vol. 10426, pp. 97–117. Springer (2017). https://doi.org/10.1007/978-3-319-63387-9 5
- 22. Landers, M., Doryab, A.: Deep reinforcement learning verification: A survey. ACM Comput. Surv. 55(14s), 330:1–330:31 (2023). https://doi.org/10.1145/3596444
- 23. Mankowitz, D.J., Michi, A., Zhernov, A., Gelmi, M., Selvi, M., Paduraru, C., Leurent, E., Iqbal, S., Lespiau, J.B., Ahern, A., Köppe, T., Millikin, K., Gaffney, S., Elster, S., Broshear, J., Gamble, C., Milan, K., Tung, R., Hwang, M., Cemgil, T., Barekatain, M., Li, Y., Mandhane, A., Hubert, T., Schrittwieser, J., Hassabis, D., Kohli, P., Riedmiller, M., Vinyals, O., Silver, D.: Faster sorting algorithms discovered using deep reinforcement learning. Nat. 618(7964), 257–263 (2023). https://doi.org/10.1038/s41586-023-06004-9
- 24. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: Balcan, M., Weinberger, K.Q. (eds.) ICML. JMLR Workshop and Conference Proceedings, vol. 48, pp. 1928–1937. JMLR.org (2016), http://proceedings.mlr.press/v48/mniha16.html
- Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, 2 edn. (2006). https://doi.org/10.1007/b98874
- OpenAI: Introducing ChatGPT (2022), https://openai.com/blog/chatgpt, accessed 14th March 2024

- Philipp, G., Song, D., Carbonell, J.G.: The exploding gradient problem demystified definition, prevalence, impact, origin, tradeoffs, and solutions. CoRR abs/1712.05577 (2017), http://arxiv.org/abs/1712.05577
- Pulina, L., Tacchella, A.: An Abstraction-Refinement Approach to Verification of Artificial Neural Networks. In: CAV. Lecture Notes in Computer Science, vol. 6174, pp. 243–257. Springer (2010). https://doi.org/10.1007/978-3-642-14295-6 24
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR abs/1707.06347 (2017), http://arxiv.org/abs/1707.06347
- Sotoudeh, M., Thakur, A.V.: Provable repair of deep neural networks. In: PLDI. pp. 588–603. ACM (2021). https://doi.org/10.1145/3453483.3454064
- 31. Srinivasan, K., Eysenbach, B., Ha, S., Tan, J., Finn, C.: Learning to be safe: Deep RL with a safety critic. CoRR abs/2010.14603 (2020), https://arxiv.org/abs/2010.14603
- 32. Sun, X., Shoukry, Y.: Provably correct training of neural network controllers using reachability analysis. CoRR abs/2102.10806 (2021), https://arxiv.org/abs/2102.10806
- 33. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. Adaptive computation and machine learning, MIT Press, Second edn. (2018), https://mitpress.mit.edu/9780262039246/reinforcement-learning/
- 34. Tan, C., Zhu, Y., Guo, C.: Building verified neural networks with specifications for systems. In: Gunawi, H.S., Ma, X. (eds.) APSys. pp. 42–47. ACM (2021). https://doi.org/10.1145/3476886.3477508
- 35. Tao, Z., Nawas, S., Mitchell, J., Thakur, A.V.: Architecture-preserving provable repair of deep neural networks. CoRR abs/2304.03496 (2023). https://doi.org/10.48550/arXiv.2304.03496
- 36. Tjeng, V., Xiao, K.Y., Tedrake, R.: Evaluating Robustness of Neural Networks with Mixed Integer Programming. In: ICLR (Poster). OpenReview.net (2019), https://openreview.net/forum?id=HyGIdiRqtm
- 37. Tran, H., Yang, X., Lopez, D.M., Musau, P., Nguyen, L.V., Xiang, W., Bak, S., Johnson, T.T.: NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems. In: CAV (1). Lecture Notes in Computer Science, vol. 12224, pp. 3–17. Springer (2020). https://doi.org/10.1007/978-3-030-53288-8_1
- 38. Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C., Kolter, J.Z.: Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Neural Network Robustness Verification. In: Ranzato, M., Beygelzimer, A., Dauphin, Y.N., Liang, P., Vaughan, J.W. (eds.) NeurIPS. pp. 29909–29921 (2021), https://proceedings.neurips.cc/paper/2021/hash/fac7fead96dafceaf80c1daffeae82a4-Abstract.html
- 39. Yang, Q., Simão, T.D., Tindemans, S.H., Spaan, M.T.J.: Safety-constrained reinforcement learning with a distributional safety critic. Mach. Learn. 112(3), 859–887 (2023). https://doi.org/10.1007/S10994-022-06187-8
- 40. Zhang, H., Wang, S., Xu, K., Li, L., Li, B., Jana, S., Hsieh, C., Kolter, J.Z.: General Cutting Planes for Bound-Propagation-Based Neural Network Verification. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) NeurIPS (2022), https://openreview.net/forum?id=5haAJAcofjc