ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

Safety-Oriented Pruning and Interpretation of Reinforcement Learning Policies*

Dennis Groß and Helge Spieker

Simula Research Laboratory
Oslo, Norway

Abstract. Pruning neural networks (NNs) can streamline them but risks removing vital parameters from safe reinforcement learning (RL) policies. We introduce an interpretable RL method called *VERINTER*, which combines NN pruning with model checking to ensure interpretable RL safety. VERINTER *exactly* quantifies the effects of pruning and the impact of neural connections on complex safety properties by analyzing changes in safety measurements. This method maintains safety in pruned RL policies and enhances understanding of their safety dynamics, which has proven effective in multiple RL settings.

1 Introduction

Reinforcement learning (RL) has transformed technology [1]. An RL agent learns a *policy* to achieve a set objective by acting and receiving rewards and observations from an environment. A *neural network (NN)* typically represents the policy, mapping environment state observations to action choices. Each observation comprises features characterizing the current environment state [2].

Unfortunately, learned policies are not guaranteed to avoid *unsafe behavior* [3], as rewards often do not fully capture complex safety requirements [4]. For example, an RL Taxi policy trained to maximize its reward for each passenger transported to their destination might not account for possible collisions.

To resolve the issue mentioned above, formal verification methods like *model checking* [2] have been proposed to reason about the safety of RL [5, 6, 7, 8]. Model checking is not limited by properties that can be expressed by rewards but supports a broader range of properties that can be expressed by *probabilistic computation tree logic (PCTL)* [9]. At its core, model checking uses mathematical models to verify a system's correctness concerning a given safety property.

Despite progress in applying verification to RL, the complexity of NNs still hides crucial details affecting safe decision-making [10]. This highlights the need for research on *interpretable and safety-focused RL methods* to enhance safe decision-making and promote responsible and interpretable RL development.

Pruning methods trim NN connections to analyze their impact on performance [11]. Yet, they lack a focus on safety.

Therefore, by integrating model checking and NN pruning, we propose a novel method named *VERINTER (VERify and INTERpret)* to *exactly interpret* neuron interconnections within NN policies concerning safety measurements. Ad-

*Funded by the EU under grant agreement number 101091783 (MARS Project) and as part of the Horizon Europe HORIZON-CL4-2022-TWIN-TRANSITION-01-03.

ditionally, VERINTER can be used as a safety-conscious pruning technique to eliminate unimportant connections from the NN while maintaining safety.

VERINTER takes three inputs: a *Markov Decision Process (MDP)* representing the RL environment, a trained policy, and a PCTL formula for safety measurements. We *incrementally build* only the reachable parts of the MDP, guided by the trained policy [2]. We then verify the policy's safety using the Storm model checker [12] and the PCTL formula.

In the case of a *safety violation* of the pruned RL policy, for instance, a collision likelihood above 1%, we can extract the information that the pruned interconnections are essential for safe decision-making. Otherwise, we may be able to prune more interconnections, only leaving the essential interconnections for safety.

Pruning an input neuron's connections removes its feature from decision-making, revealing its impact. For instance, if pruning the passenger sensor leads to significant changes like running out of fuel in a taxi RL scenario, it highlights its critical role in safe decision-making for the trained RL policy.

Our **main contribution**, VERINTER, safely prunes NN policies with formal verification, measures the impact of specific features and NN connections on safety, and is applicable across multiple benchmarks. *This study tries to bridge the gap between formal verification and interpretable RL, creating a unified method for safe and interpretable RL policies.*

Related Work Formal verification methods for RL policies are developed [5, 6, 7, 8, 13, 14] and RL policy pruning exists [15, 16, 17]. VERINTER differs by combining both formal verification and pruning in one *interpretable RL method* in the context of RL safety, accessing the impact of NN input features and connections on safety. Gangopadhyay et al. prune NN policies focusing on reachability while we *exactly verify complex safety PCTL properties* and set them into the context of *interpretable RL*. COOL-MC is a tool that verifies whether a policy violates a safety requirement or not [2]. We enhance COOL-MC by integrating it with VERINTER.

2 Background

Probabilistic model checking. A *probability distribution* over a set X is a function $\mu\colon X \to [0,1]$ with $\sum_{x\in X}\mu(x) = 1$. The set of all distributions on X is denoted $Distr(X)$.

Definition 1 (MDP). *A* MDP *is a tuple* $M = (S, s_0, Act, Tr, rew, AP, L)$ *where* S *is a finite, nonempty set of states;* $s_0 \in S$ *is an initial state;* Act *is a finite set of actions;* $Tr\colon S \times Act \to Distr(S)$ *is a partial probability transition function;* $rew\colon S \times Act \to \mathbb{R}$ *is a reward function;* AP *is a set of atomic propositions;* $L\colon S \to 2^{AP}$ *is a labeling function.*

We employ a factored state representation where each state s is a vector of features $(f_1, f_2, ..., f_d)$ where each feature $f_i \in \mathbb{Z}$ for $1 \leq i \leq d$ (d is the dimension of the state). The available actions in $s \in S$ are $Act(s) = \{a \in Act \mid Tr(s, a) \neq \bot\}$ where $Tr(s, a) \neq \bot$ is defined as action a at state s does not have a transition (action a is not available in state s). An MDP with only one action per state ($\forall s \in S : |Act(s)| = 1$) is a discrete-time Markov chain (DTMC) D.

Definition 2 (Policy)**.** *A memoryless deterministic policy for an MDP M is a function $\pi\colon S \to Act$ that maps a state $s \in S$ to action $a \in Act$.*

Applying a policy π to an MDP M yields an *induced DTMC D* where all nondeterminism is resolved. Storm [12] allows the verification of PCTL properties of induced DTMCs to make, for instance, safety measurements.

RL. The standard learning goal for RL is to learn a policy π in an MDP such that π maximizes the accumulated discounted reward, that is, $\mathbb{E}[\sum_{t=0}^{N} \gamma^t R_t]$, where γ with $0 \leq \gamma \leq 1$ is the discount factor, R_t is the reward at time t, and N is the total number of steps. In RL, an agent learns through interaction with its environment to maximize a reward signal [10].

NN policy pruning. A NN with d inputs and $|Act|$ outputs encodes a function $f\colon \mathbb{R}^d \to \mathbb{R}^{|Act|}$. Formally, the function f is given in the form of a sequence $\vec{W}^{(1)}, \ldots, \vec{W}^{(k)}$ of *weight matrices* with $\vec{W}^{(i)} \in \mathbb{R}^{d_i \times d_{i-1}}$, for all $i = 1, \ldots, k$. Pruning a weight $\vec{W}_{ij}^{(k)}$ sets it to zero, eliminating the connection between neuron i in layer k and neuron j in layer $k+1$.

We focus on the following types of pruning: *l_1-pruning* removes a specific fraction p of the weights $\vec{W}_{ij}^{(k)}$ starting with those of the smallest l_1-magnitude in layer k; *Random pruning* randomly eliminates a fixed fraction p of weights $\vec{W}_{ij}^{(k)}$; *Feature pruning* cuts all outgoing connections $\vec{W}_{ij}^{(1)}$ from a neuron linked to a specific NN policy observation feature f_i.

3 Methodology

We introduce *VERINTER's workflow*, where we first incrementally build the induced DTMC of the policy π and the MDP M as follows. For every reachable state s via the trained policy π, we query for an action $a = \pi(s)$. In the underlying MDP M, only states s' reachable via that action $a \in A(s)$ are expanded. The resulting DTMC D induced by M and π is fully deterministic, with no open action choices, and is passed to the model checker Storm for verification, yielding the *exact* safety measurement result m.

Next, the pruning procedure eliminates connections $\hat{W}$ within the NN based on predefined criteria and verifies the induced DTMC $\hat{D}$ of the pruned policy $\hat{\pi}$ and the MDP M to obtain the measurement result $\hat{m}$. Our framework remains independent of the specific pruning method used.

Then, with the completion of the pruning process, we can examine the difference between m and $\hat{m}$ to evaluate the relevance of the pruned connections.

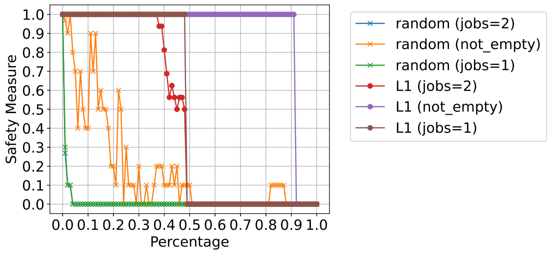


Fig. 1: Pruning methods across three Taxi events. The x-axis shows the percentage of pruned weights in the input layer, and the y-axis indicates the reachability probability of specified events (in brackets). Random pruning sample size: 10.

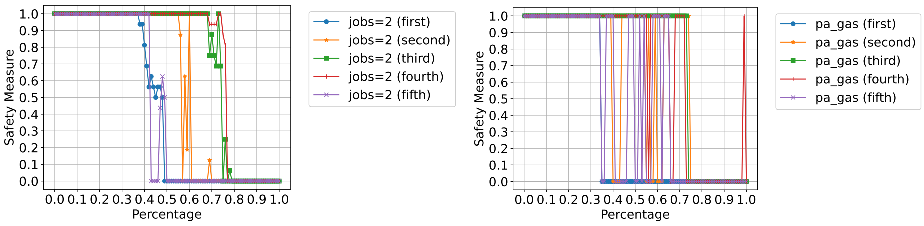
Safety feature pruning A feature f_i is important for an RL policy π and a specific measurement if its removal impacts policy safety. For instance, if all outgoing connections from the input layer receiving feature f_i are pruned, resulting in a pruned RL policy $\hat{\pi}$, we can assess if f_i is crucial for safety performance. In a taxi scenario where a passenger sensor is removed, and safety performance remains unaffected, such as the likelihood of running out of fuel remaining unchanged, we deduce that this feature does not influence this safety.

Limitations Our method supports memoryless NN policies within modeled MDP environments, only limited by state space and transition count [2]. VERINTER remains independent of the pruning method.

4 Experiments

We evaluate VERINTER in multiple model-based environments from [2] (*Taxi, Freeway, Crazy Climber, Avoidance, and Stock Market*). Experiments involve training RL policies using the deep Q-learning algorithm [1], achieving high safety success across the environments. The *Taxi policy* maintains non-empty full status, completes two jobs, and reaches a gas station with 100% success; the *Freeway policy* crosses 100% of the time safely; the *Crazy Climber policy* avoids falls; the *Avoidance policy* prevents collisions 68% of the time; and the *Stock Market policy* avoids bankruptcy.

Comparative analysis of pruning methods. This experiment compares two pruning methods on $W^{(1)}$, highlighting the model-agnostic nature of our method. In Figure 1, l_1-pruning removes more connections than random pruning while maintaining initial safety performance. Random pruning lacks consideration of connection weight, risking the removal of crucial connections and causing rapid performance degradation. Therefore, different pruning methods uniquely affect safety measurements due to varying connection pruning strategies.



(a) The reachability probability for finishing two jobs.

(b) The probability of picking up a passenger and then visiting a gas station.

Fig. 2: Each subfigure shows safety measurements for different NN layers, with the x-axis representing the percentage of pruned connections and the y-axis showing safety outcomes in the Taxi environment.

Environment	**Measure Label**	**Orig. Result**	**Pruned Feature**	**Result**
Taxi	$jobs = 2$	1	passenger_loc_x	1
Freeway	$crossed$	0.99	px_0	0.98
Crazy Climber	no_fall	0	px_1	0
Avoidance	$no\_collision_{100}$	0.68	x	0.25
Stock Market	$no\_bankruptcy$	1	$sell\_price$	1

Table 1: Safety feature prunings. The *Measure Label* refers to the safety measure. The *Orig. Result* and *Result* show the probability of conformance to the safety measure before and after pruning of the input feature *Pruned Feature*.

Effect of pruning different layers. We examine how pruning different layers of a trained NN policy affects safety in the Taxi scenario. We focus on safety measurements for completing two jobs (*jobs=2*) and the *complex probability measurement* [4] of picking up the passenger before reaching the gas station (*pa_gas*). In Figure 2, the pruning impact on safety does not consistently relate to specific layers, suggesting no dominance of low-level layers over high-level ones. Notable, pruning the first layer (shown by the blue line) slightly increased the reachability probability of completing two jobs with around 42% of neurons pruned, indicating that (further) pruning could enhance safety performance.

Safety feature pruning in different environments. Our method adapts to different RL environments, as shown in Table 1. The results vary; some pruned features maintain safety, while others compromise it.

5 Conclusion

VERINTER integrates model checking with NN pruning to refine RL policies, maintaining performance while identifying expendable features and NN connections. *Future research* could include multi-agent RL [18].

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[2] Dennis Gross, Nils Jansen, Sebastian Junges, and Guillermo A. Pérez. COOL-MC: A comprehensive tool for reinforcement learning and model checking. In *SETTA*. Springer, 2022.

[3] Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16:1437–1480, 2015.

[4] Peter Vamplew, Benjamin J. Smith, Johan Källström, Gabriel de Oliveira Ramos, Roxana Radulescu, Diederik M. Roijers, Conor F. Hayes, Fredrik Heintz, Patrick Mannion, Pieter J. K. Libin, Richard Dazeley, and Cameron Foale. Scalar reward is not enough: a response to silver, singh, precup and sutton (2021). *AAMAS*, 36(2):41, 2022.

[5] Yu Wang, Nima Roohi, Matthew West, Mahesh Viswanathan, and Geir E. Dullerud. Statistically model checking PCTL specifications on markov decision processes via reinforcement learning. In *CDC*, pages 1392–1397. IEEE, 2020.

[6] Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Deep reinforcement learning with temporal logics. In *FORMATS*, volume 12288 of *LNCS*, 2020.

[7] Tomás Brázdil, Krishnendu Chatterjee, Martin Chmelik, Vojtech Forejt, Jan Kretínský, Marta Z. Kwiatkowska, David Parker, and Mateusz Ujma. Verification of markov decision processes using learning algorithms. In *ATVA*, volume 8837 of *LNCS*, 2014.

[8] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Omega-regular objectives in model-free reinforcement learning. In *TACAS (1)*, volume 11427 of *LNCS*, pages 395–412. Springer, 2019.

[9] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects Comput.*, 6(5):512–535, 1994.

[10] Yanzhe Bekkemoen. Explainable reinforcement learning (XRL): a systematic literature review and taxonomy. *Mach. Learn.*, 113(1):355–441, 2024.

[11] Fan Ni and Min Luo. Interpretable analysis and pruning of modulation recognition network based on deep learning. In *ICDSP*, pages 35–42. ACM, 2022.

[12] Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. The probabilistic model checker Storm. *Int. J. Softw. Tools Technol. Transf.*, 24(4):589–610, 2022.

[13] Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban, and Roderick Bloem. Safe reinforcement learning using probabilistic shields (invited paper). In *CONCUR*, volume 171 of *LIPIcs*, pages 3:1–3:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[14] Dennis Gross and Helge Spieker. Probabilistic model checking of stochastic reinforcement learning policies. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART*, 2024.

[15] Jesús García-Ramírez, Eduardo F. Morales, and Hugo Jair Escalante. Model compression for deep reinforcement learning through mutual information. In *IBERAMIA*, volume 13788 of *Lecture Notes in Computer Science*, pages 196–207. Springer, 2022.

[16] Rui Xu, Siyu Luan, Zonghua Gu, Qingling Zhao, and Gang Chen. Lrp-based policy pruning and distillation of reinforcement learning agents for embedded systems. In *ISORC*, pages 1–8. IEEE, 2022.

[17] Briti Gangopadhyay, Pallab Dasgupta, and Soumyajit Dey. Pruver: Verification assisted pruning for deep reinforcement learning. In *PRICAI (1)*, volume 14325 of *Lecture Notes in Computer Science*, pages 137–149. Springer, 2022.

[18] Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. *Auton. Agents Multi Agent Syst.*, 38(1):4, 2024.