Probabilistic Verification of Recurrent Neural Networks for Single and Multi-Agent Reinforcement Learning

Luca Marzari[1] and **Enrico Marchesini**[2]
[1]TU Wien, Vienna, Austria
[2]Massachusetts Institute of Technology, Cambridge (MA), USA
luca.marzari@tuwien.ac.at, emarche@mit.edu

Abstract

History-dependent policies induced by recurrent neural networks (RNNs) rely on latent hidden state dynamics, making verification in partially observable reinforcement learning (RL) challenging. Existing RNN verification tools typically rely on restrictive modeling assumptions or coarse over-approximations of the hidden state space, which can lead to overly conservative or inconclusive results. We propose **RNN Pro**babilistic **Ve**rification (`RNN-ProVe`), a probabilistic framework that *estimates the likelihood* of undesired behaviors in RNN-based policies. `RNN-ProVe` uses policy-driven sampling to approximate the set of hidden states that are feasible under a trained policy, and derives statistical error bounds to produce bounded-error, high-confidence estimates of behavioral violations. Experiments on partially observable single-agent and cooperative multi-agent tasks show that `RNN-ProVe` yields more quantitative, feasibility-aware probabilistic guarantees than existing tools, while scaling to recurrent and multi-agent settings.

1 Introduction

Recurrent neural networks (RNNs) are widely used in reinforcement learning (RL) to handle partial observability and multiple agents [Aydeniz *et al.*, 2024; Marchesini *et al.*, 2025a]. Such RNN-based policies learn hidden-state dynamics that encode past observations and actions, making decision-making depend on the interaction *history* rather than only the current input. Verifying such policies is challenging, as most existing verification tools are designed for feedforward neural networks [Liu *et al.*, 2021; Wei *et al.*, 2025].

When applied to RNNs, verification is typically restricted to robustness analysis, where networks are checked over a single transition by bounding the effect of perturbations to a fixed input on the corresponding output. While effective for local robustness, this abstraction fails to capture the sequential and history-dependent nature of recurrent RL agents. In particular, assessing whether an RNN-based policy performs unsafe or undesired actions (or behaviors, interchangeably) requires reasoning about which hidden states can arise from *feasible, policy-induced histories*, that is, interactions the agent can actually encounter during execution (Fig. 1). Existing approaches addressing this challenge typically rely on explicit transition models to reason about recurrent policies, and verification remains computationally challenging even under these assumptions [Akintunde *et al.*, 2019; Carr *et al.*, 2021; Everett *et al.*, 2021]. Moreover, these methods are limited in both scalability and expressivity: they fail to scale to multiple agents or high-dimensional spaces [Marchesini *et al.*, 2025b; Marchesini *et al.*, 2026], and often yield only binary certificates (e.g., *safe or unsafe*). Such binary outcomes are frequently uninformative in practice, as policies may differ substantially in how often violations occur across different feasible histories. In principle, sound verification of RNN-based policies would require enumerating all reachable hidden-state configurations during execution and checking whether any of them lead to undesired behavior. However, this requirement is equivalent to computing neural network preimage measures for which a desired behavior holds [Kotha *et al.*, 2024; Zhang *et al.*, 2024], a problem that is #P-hard due to the nonlinearity and nonconvexity of neural networks [Marzari *et al.*, 2023a]. Hence, under standard complexity assumptions, obtaining an exact characterization of the feasible-history space is computationally infeasible. This observation suggests that, for recurrent policies in RL, exact worst-case verification is not only computationally prohibitive but also misaligned with the structure of history-dependent decision making.

Figure 1: (Left) Interval-based verification over-approximates the hidden state space, admitting infeasible histories. (Right) `RNN-ProVe` assesses policies over probabilistically feasible histories, yielding more precise and informative certificates.

These limitations motivate a probabilistic and quantitative perspective on verification for RNN-based policies. Rather than enumerating all feasible hidden states exactly, we aim to estimate the likelihood of undesired behavior by quantifying how much of the feasible-history space leads to behavioral violations under a trained policy. To this end, we introduce

RNN Probabilistic **Ve**rification (RNN-ProVe), a probabilistic framework that approximates the policy-induced feasible hidden state space using policy-driven sampling with explicit statistical error bounds, and produces bounded-error, high-confidence verification certificates. At its core, RNN-ProVe learns a *feasibility oracle* that distinguishes feasible from infeasible hidden state representations with respect to the distribution of histories induced by the policy during training. The oracle is implemented as a classifier trained on feasible representations encoded by the policy. RNN-ProVe then performs quantitative verification over probabilistically feasible histories identified by the classifier. Additionally, we derive statistical confidence bounds to determine the required sample sizes for verification, ensuring that the resulting certificates have provable error bounds and high statistical confidence.

We evaluate RNN-ProVe on single-agent and cooperative multi-agent partially observable tasks, demonstrating that the framework scales to settings in which existing verification tools become overly conservative or inapplicable, while providing informative probabilistic assessments of behavioral risk. Our results suggest that probabilistic verification over feasible histories is not merely a heuristic relaxation, but a principled response to fundamental computational barriers.

These outcomes motivate the following contributions: (i) We formulate the problem of *probabilistic verification* of recurrent RL policies under partial observability, explicitly accounting for the set of hidden states that are feasible under a trained policy. (ii) We propose RNN-ProVe, a policy-driven probabilistic verification framework that combines feasibility learning with Monte Carlo estimation to quantitatively assess behavioral violations in RNN-based policies. (iii) We extend the problem to fully cooperative multi-agent RL (MARL) and derive bounded-error, high-confidence guarantees for feasible-history identification and violation estimation, yielding a tractable approximation to an otherwise #P-hard verification problem.

More broadly, this work advocates probabilistic verification over feasible histories as a principled foundation for reasoning about safety and reliability in history-dependent decision-making systems.

2 Preliminaries and Related Work

At time t, an RNN produces an output y_t by using a hidden state representation h_t of an input sequence $\mathbf{x} = (x_1, \ldots, x_t)$. Following the illustration of Fig. 2, the transition function for a vanilla RNN is $h_t = \sigma(W_x x_t + W_h h_{t-1} + b_h)$, where W_x, W_h are the input and recurrent weight matrices, b_h is a bias vector, and σ is typically a non-linear activation function. The output is derived via a linear transformation as $y_t = W_y h_t + b_y$, where W_y is the output weight matrix and b_y is another bias vector. This recursive structure enables RNNs, as well as gated variants such as LSTMs [Hochreiter and Schmidhuber, 1997] and GRUs [Cho *et al.*, 2014], to model temporal dependencies over input sequences.

Figure 2: An explanatory RNN transition.

RNNs in RL. In partially observable RL, an agent typically learns a policy $\pi : \mathcal{H} \to \Delta(\mathcal{A})$; where $\mathcal{H}$ denotes the agent's action-observation history $(o_0, a_0, \ldots, o_{t-1})$, $\mathcal{A}$ is the set of actions modeling the RNN's output y, and $o \in \mathcal{O}$ are observations (i.e., RNN's input x) distributed following a stochastic transition function $T_{\mathcal{O}} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{O})$ (where $\mathcal{O}, \mathcal{S}$ are the finite sets of observations and states, respectively) [Åström, 1965]. In practice, at time t, π uses the RNN's hidden state h_t as a compressed history representation that depends on: (i) the previous hidden state h_{t-1}; and (ii) the current observation o_t, to output an action.[1] When tackling problems with $\mathcal{N}$ agents, we consider the setting where each agent $i \in \mathcal{N}$ has a local observation space $\mathcal{O}^i$ and conditions its policy on the local action-observation history [Oliehoek and Amato, 2016]. To achieve this decentralization, centralized training with decentralized execution (CTDE) is the de facto standard paradigm [Amato, 2024; Aydeniz *et al.*, 2025]. In CTDE, agents leverage privileged information during training (including the state $s \in \mathcal{S}$) while learning local history-based policies for execution [Papoudakis *et al.*, 2021]. Similarly, RNN-ProVe accesses the state while keeping decentralized execution. Thus, our framework is consistent with established practice in partially observable RL.

RNN Verification. Although RNNs are commonly used in partially observable RL tasks, existing RNN verification methods are developed primarily for robustness analysis. Robustness verification certifies the stability of network outputs under bounded input perturbations over a fixed time horizon.

Without loss of generality, consider verifying an RNN with a single scalar output, where the robustness requirement is to ensure that the output satisfies a given input-output property [Liu *et al.*, 2021]. Let f denote the RNN transition dynamics such that $h_t = f(h_{t-1}, x_t)$ for $t \in \{1, \ldots, T\}$, where T is the verification horizon. The property is encoded by appending a linear layer that outputs a margin value [Wang *et al.*, 2021]. Specifically, the final output is defined as $y_T = c^\top h_T + b_y$, where $y_T > 0$ indicates that the property is satisfied. In more detail, let $\mathcal{H}_0 = \{h_0\}$ be the initial hidden state set and let $\mathcal{X}_{1:T} = \{\mathcal{X}_1, \ldots, \mathcal{X}_T\}$ denote the admissible input sets, where each $\mathcal{X}_t = [\underline{x}_t, \overline{x}_t]$ represents an ℓ_∞-ball of radius ε around a nominal input. The over-approximated reachable hidden state sets are computed recursively as $\mathcal{H}_t = \{f(h, x) \mid h \in \mathcal{H}_{t-1},\ x \in \mathcal{X}_t\}$, and the corresponding output set is $\mathcal{Y}_T = \{y_T \mid h \in \mathcal{H}_T\}$.[2] Robustness verification then amounts to checking whether $\min(\mathcal{Y}_T) > 0$.[3]

Definition 1 (Robustness RNN-VERIFICATION)**.**
Input*: A tuple* $\mathcal{T} = \langle f, \mathcal{H}_0, \mathcal{X}_{1:T}, \pi \rangle$.

[1] For simplicity, we use $h, \mathcal{H}$ for both histories and hidden states.

[2] Reachability-based methods do not consider which hidden states can arise from feasible histories.

[3] We provide a numerical example in Appendix 6.

Output: $\texttt{robust} \iff \forall y \in \mathcal{Y}_T, y > 0$.
Equivalently, this requires proving that $\min(\mathcal{Y}_T) > 0$.

Related Work. Several methods have been proposed for RNNs robustness verification. [Akintunde *et al.*, 2019] unrolls the network over a finite time horizon, reducing it to a feedforward architecture. While effective, this strategy incurs exponential complexity in the horizon length. [Jacoby *et al.*, 2020] infers inductive invariants over hidden states to remove explicit horizon dependence. However, synthesizing precise invariants for nonlinear networks is computationally difficult and often yields conservative results. To improve scalability, interval- and relaxation-based reachability analyses are used within branch-and-bound frameworks [Ko *et al.*, 2019; Bunel *et al.*, 2020; Xu *et al.*, 2021; Mohammadinejad *et al.*, 2021; Du *et al.*, 2021; Tran *et al.*, 2023; Marzari *et al.*, 2025c]. These methods over-approximate reachable hidden states by assuming step-wise bounded input perturbations and targeting local robustness properties. Unlike robustness-based approaches that estimate output stability under bounded or stochastic input perturbations, our work quantifies how often undesired behaviors arise over the set of *policy-induced feasible histories*, addressing a fundamentally different source of uncertainty. This gap increases over long horizons, as over-approximation errors accumulate.

Beyond robustness, some works aim to verify recurrent policies. [Carr *et al.*, 2021] propose extracting finite-state automata or abstract MDPs from trained RNN policies. While conceptually powerful, these approaches rely on discretization heuristics that scale exponentially with the dimensionality of the hidden state, restricting their applicability to small-scale RNNs. [Everett *et al.*, 2021] perform backward reachability analysis directly in the state-action space. This method assumes full knowledge of the environment dynamics and deterministic transitions, assumptions that are rarely satisfied in complex, partially observable RL settings. Moreover, maintaining exact or even over-approximated reachable sets becomes intractable as the trajectory length increases.

Hence, existing approaches rely on strong modeling assumptions or suffer from severe scalability issues due to the combinatorial explosion inherent in abstracting continuous hidden state spaces. These problems highlight that current methods cannot effectively verify recurrent policies in RL.

3 RNN-Verification for RL

To verify RNN-based RL policies, we must move beyond input-perturbation-based RNN-VERIFICATION (Def. 1) and reason about uncertainty in the RNN's internal memory, namely the hidden state that encodes the agent's history. Specifically, instead of checking robustness against noisy inputs, we must verify actions over the set of feasible hidden states that could have arisen from past interactions. Hence, given an RNN-based policy f and the current observation o, our goal is to determine whether there exists any hidden state h, within the set of feasible ones $\mathcal{H}^* \subseteq \mathcal{H}$ up to o, such that $f(h, o)$ leads to a violation of a desired behavior (e.g, an action that leads to an unsafe situation, such as colliding into an obstacle). We formally define this verification problem as:

Definition 2 (RNN-VERIFICATION for RL)**.**

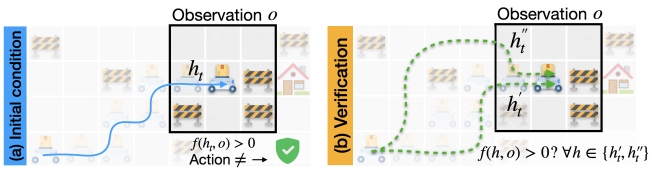

Figure 3: RNN-VERIFICATION for RL navigation: (a) The agent selects a safe action that avoids a collision. (b) The verification problem asks whether there exists a feasible history h (e.g., h_t', h_t'') such that the agent selects an undesired action, i.e., $f(h, o) \leq 0$.

Input: *A tuple* $\mathcal{T} = \langle f, \mathcal{H}, o \rangle$.
Output: $\texttt{violation} \iff \exists h \in \mathcal{H}^* \mid f(h, o) \leq 0$.

As in robustness verification, we assume that the behavior of interest can be expressed using a single scalar output whose sign indicates whether a violation occurs (i.e., the output is strictly positive if and only if the undesired action is not selected). This assumption can be enforced by appending a post-processing layer that encodes the action selection mechanism together with the desired behavioral constraint. In summary, Fig. 3 illustrates our verification problem, which checks whether there exists a feasible hidden state $h \in \mathcal{H}^*$ such that, when combined with the current observation o, the RNN selects an action that violates a desired behavior.

In contrast to robustness, RNN-VERIFICATION for RL requires characterizing the set of feasible hidden states $\mathcal{H}^*$ that reach a certain configuration (state). To this end, formal verification methods typically use coarse geometric over-approximations, such as independent intervals or hyperrectangles. However, as previously stated, these relaxations may admit spurious hidden states (i.e., combinations of hidden neuron values that cannot arise from any valid interaction history of an RL agent). As a result, propagating such inflated sets through the transition function f can yield spurious counterexamples, or false positives, that do not correspond to situations the agent can actually encounter. Sec. 3.1 discusses how `RNN-ProVe` addresses this issue.

Quantifying violations. Existing RNN verification tools typically return only a binary outcome and do not quantify the extent to which a trained RNN violates a desired requirement. In the context of RNN-based RL policies, this means that it is not possible to measure how much of the feasible hidden state space leads to violations of a desired behavior.

This limitation motivates our novel quantitative formulation, #RNN-VERIFICATION. In analogy with the feedforward network case [Kotha *et al.*, 2024; Marchesini *et al.*, 2023; Marzari *et al.*, 2025a], we invert the analysis perspective. Rather than characterizing all the inputs of the RNN that lead to violating the desired behavior, we characterize the set of feasible hidden states (histories) that violate the behavior for a certain situation at time t (i.e., for a fixed input observation). This formulation enables explicit reasoning about uncertainty over feasible histories and yields a characterization of the region of hidden states that induce undesired behavior.

Definition 3 (#RNN-VERIFICATION for RL)**.**
Input: *A tuple* $\mathcal{T} = \langle f, \mathcal{H}, o \rangle$.
Output: $Vol(\Gamma(\mathcal{T}))$.

With $\Gamma(\mathcal{T}) := \{h \in \mathcal{H}^* \mid f(h, o) \leq 0\} \subseteq \mathcal{H}$ *the set of feasible histories for a fixed* o *that result in undesired behaviors.*

Broadly speaking, the objective of this quantitative formulation is to compute the volume of $\Gamma(\mathcal{T})$ relative to $\mathrm{Vol}(\mathcal{H})$, which corresponds to the probability that the current policy exhibits an undesired behavior. We also note that exactly characterizing the feasible-history space $\mathcal{H}^*$ is at least as hard as computing the volume of inputs that violate a behavior for a neural network, a problem known as #DNN-VERIFICATION [Marzari *et al.*, 2023a].

Theorem 1. *The* #RNN-VERIFICATION *problem is #P-hard.*

In Appendix 7 we report a complete argument of this theoretical result. Under standard complexity assumptions, it implies that for any non-trivial RNN-based RL policy, obtaining an exact characterization of $\mathrm{Vol}(\Gamma(\mathcal{T}))$ is computationally infeasible. This observation motivates our shift to a probabilistic perspective on verification, which we develop in Sec. 3.1.

Verifying Cooperative MARL. To the best of our knowledge, we now extend the previous problem formulations for the first time to fully cooperative MARL. We consider a setting in which a decentralized set of $\mathcal{N}$ agents must coordinate to achieve a shared objective. In this setting, each agent $i \in \mathcal{N}$ acts according to its own RNN-based policy f_i, relying on its local observation o_i and hidden state representation $h_i \in \mathcal{H}_i$. Considering the agents' common goal, we define verification of a cooperative MARL problem as the conjunction of individual desired-behavior constraints, meaning the system does not exhibit violations if every agent satisfies its individual desired behavior. Conversely, the system has violations if *at least one* agent violates its behavior. We formally define the verification problem for this setting as follows:

Definition 4 (RNN-VERIFICATION for MARL)**.**
Input: $\mathbf{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_\mathcal{N}\}$, *where* $\mathcal{T}_i = \langle f_i, \mathcal{H}_i, o_i \rangle$.
Output: $\texttt{violation} \iff \exists i \in \{1, \ldots, \mathcal{N}\}, \exists h_i \in \mathcal{H}_i^* \mid f_i(h_i, o_i) \leq 0.$

We thus extend the quantitative formulation of Def. 3 to the multi-agent domain. Due to our interest in fully cooperative MARL, we are interested in quantifying the agent with the highest probability of violations given its current local history. Therefore, the #RNN-VERIFICATION problem for MARL aims to compute the maximum volume of undesired feasible histories across all agents.

Definition 5 (#RNN-VERIFICATION problem for MARL)**.**
Input: $\mathbf{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_\mathcal{N}\}$, *where* $\mathcal{T}_i = \langle f_i, \mathcal{H}_i, o_i \rangle$.
Output: $\max_{i \in \mathcal{N}} Vol(\Gamma(\mathcal{T}_i))$.

Since we adopt the standard decentralized setting in which each agent i acts based solely on its local information, the evaluation of $Vol(\Gamma(\mathcal{T}_i))$ for each agent is independent. The verification certificate is therefore determined by the worst-performing agent. This cooperative MARL formulation preserves tractability by decomposing into independent verification tasks, enabling parallel analysis without introducing additional conservatism. Accordingly, in the remainder of the paper, we present the proposed probabilistic framework and its theoretical guarantees for the single-agent case.

3.1 RNN-ProVe: a Novel Probabilistic Approach

Following the result in Theorem 1, `RNN-ProVe` solves a relaxed probabilistic version of the #RNN-VERIFICATION problem for RL. Intuitively, the following formalization asks for a scalar volume estimate that accurately approximates the volume measure of the true undesired set $\Gamma(\mathcal{T})$.

Definition 6 (Approximate #RNN-VERIFICATION for RL)**.**
Input: *A tuple* $\mathcal{T} = \langle f, \mathcal{H}, o \rangle$, *an error tolerance* $\epsilon \in (0, 1)$, *and a confidence parameter* $\delta \in (0, 1)$.
Output: $\tilde{V} \in [0, Vol(\mathcal{H})]$ *estimate of* $Vol(\Gamma(\mathcal{T}))$, *s.t.*

$$\Pr\left(\left|\frac{\tilde{V} - Vol(\Gamma(\mathcal{T}))}{Vol(\mathcal{H})}\right| \leq \epsilon\right) \geq 1 - \delta.$$

This probabilistic problem definition requires that, with high confidence $1 - \delta$, the deviation between the estimated $\tilde{V}$ and true volumes $Vol(\Gamma(\mathcal{T}))$, normalized by the total volume of the hidden space $Vol(\mathcal{H})$, remains within the specified error bound ϵ. Since computing $Vol(\Gamma(\mathcal{T}))$ exactly is computationally prohibitive, an alternative strategy is to perform a Monte Carlo estimation as $Vol(\Gamma(\mathcal{T})) \approx Vol(\mathcal{H}) \cdot \frac{1}{k} \sum_{i=1}^{k} \mathbb{1}_{f(h_i, o) \leq 0}$, where $h_1, \ldots, h_k$ are independent samples drawn from the hidden state domain $\mathcal{H}$, and $\mathbb{1}_{f(h_i, o) \leq 0}$ is the indicator function for undesired behaviors. However, this naive sampling strategy implicitly assumes that all histories within the domain $\mathcal{H}$ are feasible, while the set of feasible histories $\mathcal{H}^*$ is a complex, lower-dimensional manifold within $\mathcal{H}$. Consequently, a uniform sample from $\mathcal{H}$ inevitably includes unfeasible histories, leading to inaccurate verification results (as we will show in our experiments).

To address this issue, one would require an oracle that determines whether, for a given configuration (i.e., a state $s \in \mathcal{S}$), a history h is feasible, meaning that it can be realized as an internal hidden state by π. Since no exact analytical form of this oracle exists, `RNN-ProVe` leverages the universal approximation theorem [Hornik *et al.*, 1989] to learn an approximation of this oracle, as illustrated in Fig. 4. In more detail, during RL training, the agent's policy explores a wide range of environment states s and generates corresponding internal hidden state representations of the actual histories leading to s. `RNN-ProVe` records these state-hidden state pairs to construct a dataset used to train a *feasibility history*

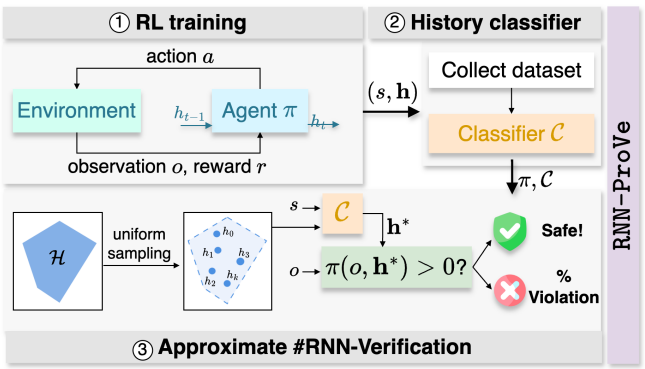

Figure 4: `RNN-ProVe` overview.

classifier $\mathcal{C} : \mathcal{S} \times \mathcal{H} \to \{0, 1\}$. Specifically, given a state s and a hidden state h, the classifier predicts whether h corresponds to a feasible hidden state for the trained policy (phases 1 and 2 in the upper part of Fig. 4). This data collection process leverages maximum entropy-based RL exploration mechanisms, which, in our experiments, lead to a sufficient coverage of the hidden state space for our dataset [Kaelbling *et al.*, 1996]. As a result, the dataset captures a wide variety of the feasible hidden state space, including both optimal and suboptimal but realizable interactions, which is essential for training a reliable feasibility classifier. Once the classifier $\mathcal{C}$ is trained, `RNN-ProVe` enters the quantitative verification phase (phase 3 in Fig. 4). For a given state of interest s where we want to verify a certain behavior, the method samples n candidate history vectors from the hidden state domain $\mathcal{H}$ and evaluates them using $\mathcal{C}$. The classifier acts as a feasibility oracle, retaining only those representations $\mathbf{h}^*$ that are predicted to be feasible hidden states of the RNN-based policy leading to state s. These validated histories are then used to evaluate the policy under the observation o associated with s. Hence, for each $h \in \mathbf{h}^*$, the method checks whether the action selected by the trained policy satisfies the desired behavior (i.e., selects a certain action), that is, whether $f(h, o) > 0$. If violations are observed, `RNN-ProVe` estimates the undesired volume $\tilde{V}$ as the ratio of violating hidden states to the total number of validated hidden states tested.

The reliability of the resulting certificate depends on two sources of uncertainty: (i) the accuracy of the feasibility classifier, which determines whether the identified set faithfully represents the true feasible hidden state manifold without admitting spurious histories; and (ii) sampling variance, which depends on the number of validated hidden states used to estimate the undesired volume. `RNN-ProVe` addresses both sources of uncertainty through rigorous statistical analysis, providing high-confidence and bounded-error certificates.

3.2 Theoretical Guarantees

`RNN-ProVe` guarantees rely on decomposing the estimation error ϵ into two components: (i) the approximation error of the histories classifier; and (ii) the statistical variance of the safety evaluation (i.e., the sampling error). We bound both terms using Hoeffding's inequality [Hoeffding, 1963] to derive a global probabilistic certificate.

Lemma 1 (Hoeffding's inequality [Hoeffding, 1963])**.** *Let $X_1, \ldots, X_n$ be independent random variables such that $X_i \in [0, 1]$, and let $\mu = \mathbb{E}[X_i]$ and its estimate $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} X_i$. Then, for any $\epsilon > 0$, $\Pr(|\hat{\mu} - \mu| \geq \epsilon) \leq 2\exp\big(-2n\epsilon^2\big)$.*

In our setting, let, without loss of generality, $\mathcal{H} \subseteq [0, 1]^n$ be the hidden state domain and $\epsilon, \delta \in (0, 1)$ be the total error tolerance and confidence parameters required by Def. 6. We distribute these budgets into classifier validation components $(\hat{e}, \epsilon_{clf}, \delta_{clf})$ and verification components $(\epsilon_{ver}, \delta_{ver})$, such that $\epsilon = \hat{e} + \epsilon_{clf} + \epsilon_{ver}$ and $\delta = \delta_{clf} + \delta_{ver}$.

Theorem 2 (`RNN-ProVe` Probabilistic Guarantees)**.** *Let $\hat{p}$ be the empirical probability of undesired behavior computed over N histories validated via the classifier $\mathcal{C}$. Let $\hat{e}$ be the empirical error rate of $\mathcal{C}$ measured on a held-out validation set of size M. If the sample sizes satisfy $N \geq \frac{\ln(2/\delta_{ver})}{2\epsilon_{ver}^2}$ and $M \geq \frac{\ln(2/\delta_{clf})}{2\epsilon_{clf}^2}$, then the volume estimate $\tilde{V} = Vol(\mathcal{H}) \cdot \hat{p}$ computed by `RNN-ProVe` satisfies $\Pr\left(\left|\frac{\tilde{V} - Vol(\Gamma(\mathcal{T}))}{Vol(\mathcal{H})}\right| \leq \epsilon\right) \geq 1 - \delta$.*

Proof. Let p^* denote the true probability of undesired behavior (i.e., the $Vol(\Gamma(\mathcal{T}))/Vol(\mathcal{H})$) and $p_{\mathcal{C}}$ denote the probability induced by the classifier $\mathcal{C}$. We bound the total deviation $|\hat{p} - p^*|$ by applying the triangle inequality $|\hat{p} - p^*| \leq |\hat{p} - p_{\mathcal{C}}| + |p_{\mathcal{C}} - p^*|$, representing the verification sampling and classifier error. To bound the classification error, we note that by applying Lemma 1, and enforcing that $2e^{-2M\epsilon_{clf}^2} \leq \delta_{clf}$, we obtain a validation set of size $M \geq \frac{\ln(2/\delta_{clf})}{2\epsilon_{clf}^2}$. By using this M we guarantee, with probability at least $1 - \delta_{clf}$, that $|p_{\mathcal{C}} - p^*| \leq \epsilon_{clf} \leq \epsilon_{clf} + \hat{e}$. Analogously, $\hat{p}$ is a Monte Carlo estimate of $p_{\mathcal{C}}$ obtained from N independent samples. By applying once again Lemma 1, if $N \geq \frac{\ln(2/\delta_{ver})}{2\epsilon_{ver}^2}$ we have that $|\hat{p} - p_{\mathcal{C}}| \leq \epsilon_{ver}$, with probability at least $1 - \delta_{ver}$. By union bound, these inequalities hold simultaneously with probability at least $1 - (\delta_{clf} + \delta_{ver}) = 1 - \delta$, giving $|\hat{p} - p^*| \leq \epsilon_{ver} + \hat{e} + \epsilon_{clf} = \epsilon$. Finally, substituting $\hat{p} = \frac{\tilde{V}}{\text{Vol}(\mathcal{H})}$ and $p^* = \frac{Vol(\Gamma(\mathcal{T}))}{Vol(\mathcal{H})}$, we obtain $\Pr\left(\left|\frac{\tilde{V}}{\text{Vol}(\mathcal{H})} - \frac{\text{Vol}(\Gamma(\mathcal{T}))}{\text{Vol}(\mathcal{H})}\right| \leq \epsilon\right) \geq 1 - \delta$, concluding the argument. □

This result shows that `RNN-ProVe` provides a bounded-error, high-confidence certificate for the volume of hidden states violating the desired behavior, providing a solution to the approximate #RNN-VERIFICATION for RL problem.

For example, consider a target confidence of $1-\delta = 99.9\%$ and an $\epsilon = 5\%$ error in the certified violation volume. Suppose the classifier $\mathcal{C}$ achieves an empirical error rate of $\hat{e} = 0.01$ (i.e., 99% accuracy). We allocate the remaining error budget evenly between classifier validation and Monte Carlo estimation by setting $\epsilon_{clf} = 0.02$ and $\epsilon_{ver} = \epsilon - (\hat{e} + \epsilon_{clf}) = 0.02$, with confidence parameters $\delta_{clf} = \delta_{ver} = \delta/2 = 5 \cdot 10^{-4}$. By Theorem 2, the required validation set size for the classifier and the sample histories for the verification are $M, N \geq \frac{\ln(2/\delta)}{2\epsilon^2} = \frac{\ln(2000)}{2 \cdot 0.02^2} \approx 10362$. Hence, `RNN-ProVe` certifies the violation volume with 99.9% confidence and a maximum error margin of 5% using $\approx 2 \cdot 10362$ total queries. Crucially, parallel execution on a GPU makes this computation take negligible time as shown in our experiments.

Limitations. While these guarantees are not restricted to discrete domains, extending `RNN-ProVe` to continuous settings requires identifying feasible hidden states when state and observation spaces are continuous. A natural extension, following standard practice in verification, is to discretize or quantize these spaces to obtain a finite abstraction over which feasibility can be learned and verified [Liu *et al.*, 2021; Marzari *et al.*, 2024; Marzari *et al.*, 2026a]. An alternative is to approximate feasibility via conservative reachable-set approximations of the hidden-state dynamics. Exploring such extensions is left for future work. `RNN-ProVe` may also be less effective for very small problem instances or when the rate of undesired behavior is extremely low. In these cases, the number of samples required to achieve high statistical

confidence may approach or exceed the number of feasible histories, making direct enumeration more appropriate. More generally, since the sample complexity scales as $O(\epsilon^{-2})$, estimating extremely rare violations (where $p^* \to 0$) requires very small error tolerances, leading to large sample requirements. This behavior reflects the complexity of the underlying counting problem rather than a limitation of our method.

4 Experiments

Our experiments address the following points: (i) *Can* `RNN-ProVe` *verify recurrent policies in partially observable single- and cooperative multi-agent tasks?* (ii) *Does* `RNN-ProVe` *scale with problem complexity, in regimes where existing RNN verification methods become computationally intractable?* (iii) *How does* `RNN-ProVe` *compare to naive sampling baselines, and how does incorporating* `RNN-ProVe` *dataset affect the linearization-based baseline?*

We evaluate `RNN-ProVe` on two classes of discrete environments: (i) single-agent navigation tasks of increasing size, with policies trained using the widely adopted DRQN algorithm [Hausknecht and Stone, 2017]; (ii) cooperative multi-agent policies trained on the BoxPushing (BP) domain [Xiao *et al.*, 2020], with increasing environment sizes, using the CTDE DRQN-based approach of [Xiao *et al.*, 2020].[4] Both environments are described in Fig. 5 and Appendix 8.

Since existing RNN verification tools are not designed to identify the feasible history space $\mathcal{H}^*$ and would therefore require an external oracle for this purpose, we adapt the linearization-based RNN verification approach of [Ko *et al.*, 2019] as a baseline for comparison with `RNN-ProVe`. In particular, we use the same feasibility oracle as in `RNN-ProVe` and perform an exact quantitative volume estimation. Employing the same feasibility oracle ensures a fair and controlled comparison. Our experiments thus aim to highlight the scalability limitations of exact solvers and the practical advantages of `RNN-ProVe` in challenging settings. Notably, both `RNN-ProVe` and the baseline are agnostic to the specific learning algorithm used to train the policies, as is standard in verification. In all experiments, the recurrent component of the policy is implemented using a GRU layer, selected for its strong empirical performance. Experiments are conducted on Xeon E5-2650 CPU nodes with 64 GB of RAM. We verify 10 independently trained policies per task, collected at convergence. The following curves show the average return over the independent runs, smoothed over 100 episodes, with shaded regions indicating 95% confidence intervals. We also trained three independent feasibility classifiers with different random initializations; as the resulting verification estimates exhibited negligible variation, we report only the averaged results in Tab. 1. The environmental impact of the experiments is discussed in Appendix 10.

Desired Behaviors. For single-agent navigation, we verify collision-avoidance behaviors (Def. 3), where the agent should not select an action that leads to a collision (as in [Marzari *et al.*, 2023b; Marzari *et al.*, 2025b; Marzari *et al.*, 2025d]). In the cooperative BP, we verify a joint behavior

[4]Complete hyperparameters are reported in Appendix 9.

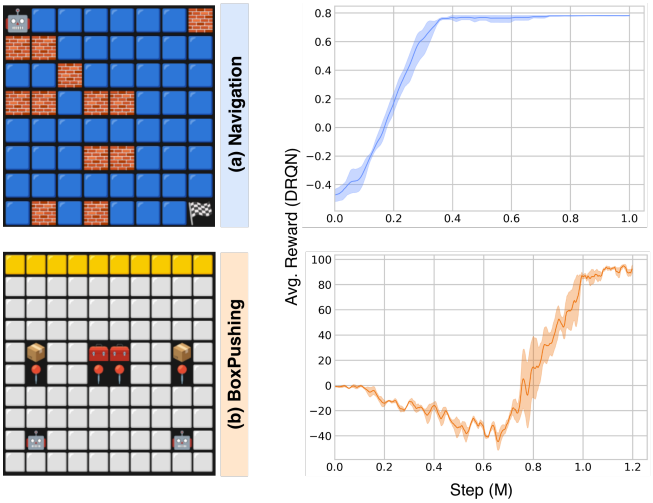

Figure 5: (Top) Single-agent navigation with red obstacles, where the agent must navigate to the goal location and the corresponding training performance. (Bottom) Cooperative multi-agent BP env, where agents must jointly push a box to the yellow goal, and the corresponding joint training performance of the agents.

(Def. 5), requiring that both agents pick the push action when positioned in front of the box. These are designed to assess both individual and cooperative violations. At a high-level, the resulting violation percentages can be interpreted as the likelihood that an agent exhibits undesired behavior when reaching a given state from arbitrary feasible histories, providing an indicator of policy generalization and robustness to history-dependent uncertainty, such as that induced by noisy or non-deterministic interactions.

4.1 Empirical Evaluation

Verification of RNN-based policies. `RNN-ProVe` verifies recurrent policies in all single-agent navigation and cooperative multi-agent BP. In navigation tasks, `RNN-ProVe` identifies non-trivial violation rates in smaller environments, with a mean violation of approximately 1% in the 4×4 grid and approximately 13% in the 8×8 grid (Table 1). In contrast, for the 16×16 environment, `RNN-ProVe` reports low violation rates below 1% under the specified (ϵ, δ) guarantees.

Fig. 6 also visualizes the spatial distribution of violations for the 8×8 environment. Despite strong training performance, the learned policies do not generalize uniformly across feasible hidden states. Specifically, the reported violation values correspond to the fraction of feasible histories from which the agent selects an action that eventually leads to a collision, revealing sensitivity to particular configurations near obstacles. These failures are not captured by standard empirical evaluations based on average return [Marzari *et al.*, 2026b]. In the cooperative BP task, `RNN-ProVe` certifies mean violation values of approximately 1.18% in the 10×10 environment and approximately 1.7% in the 20×20 environment, demonstrating that the framework naturally extends to cooperative settings with interacting recurrent agents while providing quantitative, probabilistic safety assessments.

Method	Task	Env size	GRU size	Feas. Oracle	$1-\hat{e}$	# samples	$\lvert\mathbf{h}^*\rvert$	Avg. Violation	$1-\delta$	ϵ	Time (s)
Baseline	Nav	4×4	4	Exact	-	-	-	1.44%	-	-	223.13
Baseline	Nav	4×4	4	Approx.	99.97	-	-	1.43%	-	-	215.58
Monte-Carlo	Nav	4×4	4	-	-	1M	1M	23.87%	-	-	0.0244
RNN-ProVe	Nav	4×4	4	Approx.	99.97	100k	12637	1.62%	99%	3.11%	0.0053
RNN-ProVe	Nav	4×4	4	Approx.	99.97	1M	127359	1.42%	99%	1.0%	0.0271
Baseline	Nav	8×8	8	Approx.	99.99	-	-	13.01%	-	-	1050.53
Monte-Carlo	Nav	8×8	8	-	-	1M	1M	21.24%	-	-	0.0244
RNN-ProVe	Nav	8×8	8	Approx.	99.99	100k	10890	13.04%	99%	3.33%	0.0024
RNN-ProVe	Nav	8×8	8	Approx.	99.99	1M	110293	13.37%	99%	1.05%	0.0207
Baseline	Nav	16×16	12	Approx.	98.36	-	-	-	-	-	Timeout
Monte-Carlo	Nav	16×16	12	-	-	1M	1M	4.94%	-	-	0.0244
RNN-ProVe	Nav	16×16	12	Approx.	98.36	100k	12775	0.64%	99%	4.7%	0.0028
RNN-ProVe	Nav	16×16	12	Approx.	98.36	1M	127015	0.79%	99%	2.61%	0.0223
RNN-ProVe	BP	10×10	16	Approx.	99.98 | 99.95	100k	20320	1.15%	99%	2.45%	0.0072
RNN-ProVe	BP	10×10	16	Approx.	99.98 | 99.95	1M	41061	1.18%	99%	1.76%	0.0179
RNN-ProVe	BP	20×20	32	Approx.	99.23 | 99.99	100k	32413	1.51%	99%	2.69%	0.0140
RNN-ProVe	BP	20×20	32	Approx.	99.23 | 99.99	1M	65027	1.73%	99%	2.12%	0.0476

Table 1: Quantitative verification results for RNN-based navigation (Nav) and BP tasks. Gray rows correspond to the baselines; RNN-ProVe reports probabilistic certificates with bounded error ϵ and confidence at least $1-\delta$.

Scalability Consideration. Results in Tab. 1 highlight the scalability limitations of exact linearization-based RNN verification when applied to recurrent RL policies. In navigation, the baseline requires several minutes of computation to verify the small 4×4 environment, increases its runtime to over 18 minutes in the 8×8 setting, and fails to complete the 16×16 task within the imposed 45-minute timeout. This behavior reflects the rapid growth of exact over-approximation-based RNN verification methods as both the environment size and the GRU hidden state dimension increase. In contrast, RNN-ProVe scales efficiently across all tasks, returning quantitative verification results (comparable with the baseline) in milliseconds. This efficiency stems from reasoning probabilistically over feasible histories. Moreover, existing RNN verification methods are not designed to handle cooperative multi-agent formulations and therefore cannot be directly applied to the BP task. In these settings, RNN-ProVe enables probabilistically quantitative verification of recurrent policies under decentralized execution, demonstrating its applicability beyond single-agent scenarios.

Ablation Study. We report two additional experimental variants for the navigation tasks in Table 1. For the 4×4 environment, we compare baseline results obtained with an exact feasibility oracle to those obtained with the approximate feasibility oracle learned by RNN-ProVe. In this environment, where an exact oracle is achievable, we notice that the verification results are nearly identical, indicating that the histories collected during training already cover a large portion of the feasible history space. When used within the RNN-ProVe pipeline, this data enables learning a feasibility classifier that closely approximates the true feasible-history distribution. We also compare our approach against a naive *Monte-Carlo* variant that removes the feasibility classifier and evaluates the policy on uniformly sampled hidden states. This leads to substantially higher violation rates, as most sampled hidden states do not correspond to realizable interaction histories and therefore induce spurious failures when fed into the policy. Overall, these experiments rule out two natural alternatives: naive linearization and bound-propagation–based hidden-state bound estimation, as well as naive Monte Carlo estimation without feasibility filtering. They further show that the proposed components are not only necessary to verify histories arising under policy execution, but also sufficient to produce accurate probabilistic estimates of behavioral violations. By operating over feasible combinations of hidden-state features, RNN-ProVe computes violation values that reliably reflect the true behavior of the policy.

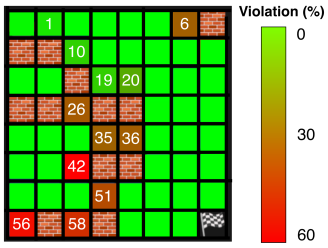

Figure 6: Violation heatmap for 8×8 navigation. Each cell (state) is colored based on the violation % computed via RNN-ProVe (red cells indicate higher violations).

5 Discussion

We presented RNN-ProVe, a probabilistic verification framework for recurrent policies in single- and cooperative multi-agent RL. RNN-ProVe addresses the challenge of reasoning over *feasible histories*, which is not captured by existing RNN verification methods. By combining policy-driven sampling with statistical analysis, we produce quantitative, high-confidence certificates that estimate how frequently undesired behaviors occur over feasible hidden states. RNN-ProVe enables scalable analysis in regimes where exact methods become intractable.

A key insight emerging from our analysis is that verification precision is governed not by tighter bounds, but by accurately capturing the combinatorial structure of feasible hidden state configurations induced by policy execution.

Acknowledgements

This work was supported in part by the "Fondo Italiano per la Scienza" project (FIS-2024-05614), and in part by the Austrian Science Fund (FWF) project 10.55776/ESP1944725, the Vienna Science and Technology Fund (WWTF) under Grant ICT22-023 (TAIGER) and the European Union, RobustifAI project, ID 101212818. The authors thank the reviewers for their insightful and constructive feedback, which has substantially improved the quality of this work.

References

[Akintunde *et al.*, 2019] Michael E Akintunde, Andreea Kevorchian, Alessio Lomuscio, and Edoardo Pirovano. Verification of rnn-based neural agent-environment systems. In *AAAI Conference on Artificial Intelligence*, pages 6006–6013, 2019.

[Amato, 2024] Christopher Amato. An introduction to centralized training for decentralized execution in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2409.03052*, 2024.

[Åström, 1965] Karl Johan Åström. Optimal control of markov processes with incomplete state information i. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

[Aydeniz *et al.*, 2024] Ayhan Alp Aydeniz, Enrico Marchesini, Christopher Amato, and Kagan Tumer. Entropy seeking constrained multiagent reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, page 2141–2143, 2024.

[Aydeniz *et al.*, 2025] Ayhan Alp Aydeniz, Enrico Marchesini, Robert Loftin, Christopher Amato, and Kagan Tumer. Safe entropic agents under team constraints. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, page 2411–2413, 2025.

[Bunel *et al.*, 2020] Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Philip HS Torr, Pushmeet Kohli, and M Pawan Kumar. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(42):1–39, 2020.

[Carr *et al.*, 2021] Steven Carr, Nils Jansen, and Ufuk Topcu. Verifiable rnn-based policies for pomdps under temporal logic constraints. In *International Conference on International Joint Conferences on Artificial Intelligence*, pages 4121–4127, 2021.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[Du *et al.*, 2021] Tianyu Du, Shouling Ji, Lujia Shen, Yao Zhang, Jinfeng Li, Jie Shi, Chengfang Fang, Jianwei Yin, Raheem Beyah, and Ting Wang. Cert-rnn: Towards certifying the robustness of recurrent neural networks. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 15–19, 2021.

[Everett *et al.*, 2021] Michael Everett, Golnaz Habibi, Chuangchuang Sun, and Jonathan P How. Reachability analysis of neural feedback loops. *IEEE Access*, 9:163938–163953, 2021.

[Hausknecht and Stone, 2017] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*, 2017.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hoeffding, 1963] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[Hornik *et al.*, 1989] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[Jacoby *et al.*, 2020] Yuval Jacoby, Clark Barrett, and Guy Katz. Verifying recurrent neural networks using invariant inference. In *International Symposium on Automated Technology for Verification and Analysis*, pages 57–74. Springer, 2020.

[Kaelbling *et al.*, 1996] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Katz *et al.*, 2017] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117, 2017.

[Ko *et al.*, 2019] Ching-Yun Ko, Zhaoyang Lyu, Lily Weng, Luca Daniel, Ngai Wong, and Dahua Lin. Popqorn: Quantifying robustness of recurrent neural networks. In *International Conference on Machine Learning*, pages 3468–3477, 2019.

[Kotha *et al.*, 2024] Suhas Kotha, Christopher Brix, J Zico Kolter, Krishnamurthy Dvijotham, and Huan Zhang. Provably bounding neural network preimages. *Advances in Neural Information Processing Systems*, 2024.

[Liu *et al.*, 2021] Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, Mykel J Kochenderfer, et al. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 4(3-4):244–404, 2021.

[Marchesini *et al.*, 2023] Enrico Marchesini, Luca Marzari, Alessandro Farinelli, and Christopher Amato. Safe deep reinforcement learning by verifying task-level properties. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023*, pages 1466–1475. ACM, 2023.

[Marchesini *et al.*, 2025a] Enrico Marchesini, Andrea Baisero, Rupali Bhati, and Christopher Amato. On stateful value factorization in multi-agent reinforcement learning.

In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, page 1445–1453, 2025.

[Marchesini *et al.*, 2025b] Enrico Marchesini, Benjamin Donnot, Constance Crozier, Ian Dytham, Christian Merz, Lars Schewe, Nico Westerbeck, Cathy Wu, Antoine Marot, and Priya L Donti. Rl2grid: Benchmarking reinforcement learning in power grid operations. *arXiv preprint arXiv:2503.23101*, 2025.

[Marchesini *et al.*, 2026] Enrico Marchesini, Eva Boguslawski, Alessandro Leite, Christopher Amato, Matthieu DUSSARTRE, Marc Schoenauer, Benjamin Donnot, and Priya L. Donti. MARL2grid-TR: A multi-agent RL benchmark in power grid operations. In *The Fourteenth International Conference on Learning Representations*, 2026.

[Marzari *et al.*, 2023a] Luca Marzari, Davide Corsi, Ferdinando Cicalese, and Alessandro Farinelli. The #dnn-verification problem: Counting unsafe inputs for deep neural networks. In *International Joint Conference on Artificial Intelligence, IJCAI*, pages 217–224, 2023.

[Marzari *et al.*, 2023b] Luca Marzari, Enrico Marchesini, and Alessandro Farinelli. Online safety property collection and refinement for safe deep reinforcement learning in mapless navigation. In *IEEE International Conference on Robotics and Automation, (ICRA)*, pages 7133–7139. IEEE, 2023.

[Marzari *et al.*, 2024] Luca Marzari, Davide Corsi, Enrico Marchesini, Alessandro Farinelli, and Ferdinando Cicalese. Enumerating safe regions in deep neural networks with provable probabilistic guarantees. In *Conference on Artificial Intelligence (AAAI)*, pages 21387–21394, 2024.

[Marzari *et al.*, 2025a] Luca Marzari, Ferdinando Cicalese, Alessandro Farinelli, Christopher Amato, and Enrico Marchesini. Verifying online safety properties for safe deep reinforcement learning. *ACM Trans. Intell. Syst. Technol.*, September 2025.

[Marzari *et al.*, 2025b] Luca Marzari, Priya L. Donti, Changliu Liu, and Enrico Marchesini. Improving policy optimization via ϵ-retrain. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025*, pages 1464–1472. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2025.

[Marzari *et al.*, 2025c] Luca Marzari, Isabella Mastroeni, and Alessandro Farinelli. Advancing neural network verification through hierarchical safety abstract interpretation. In *ECAI 2025 - 28th European Conference on Artificial Intelligence*, pages 1736–1743, 2025.

[Marzari *et al.*, 2025d] Luca Marzari, Francesco Trotti, Enrico Marchesini, and Alessandro Farinelli. Designing control barrier function via probabilistic enumeration for safe reinforcement learning navigation. *IEEE Robotics and Automation Letters*, 10(10):9630–9637, 2025.

[Marzari *et al.*, 2026a] Luca Marzari, Manuele Bicego, Ferdinando Cicalese, and Alessandro Farinelli. On the probabilistic learnability of compact neural network preimage bounds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 35707–35714, 2026.

[Marzari *et al.*, 2026b] Luca Marzari, Changliu Liu, Priya L Donti, and Enrico Marchesini. ε-retraining reinforcement learning algorithms. *Autonomous Agents and Multi-Agent Systems*, 40(1):24, 2026.

[Mohammadinejad *et al.*, 2021] Sara Mohammadinejad, Brandon Paulsen, Jyotirmoy V Deshmukh, and Chao Wang. Diffrnn: differential verification of recurrent neural networks. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 117–134. Springer, 2021.

[Oliehoek and Amato, 2016] Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

[Papoudakis *et al.*, 2021] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[Tran *et al.*, 2023] Hoang-Dung Tran, Taylor T Johnson, et al. Reachability analysis of recurrent neural networks using star sets. In *ACM International Conference on Hybrid Systems: Computation and Control*, 2023.

[Valiant, 1979] Leslie G Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[Wang *et al.*, 2021] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 34:29909–29921, 2021.

[Wei *et al.*, 2025] Tianhao Wei, Hanjiang Hu, Luca Marzari, Kai S. Yun, Peizhi Niu, Xusheng Luo, and Changliu Liu. Modelverification.jl: A comprehensive toolbox for formally verifying deep neural networks. In *Computer Aided Verification - 37th International Conference, CAV 2025*, volume 15932 of *Lecture Notes in Computer Science*, pages 395–408. Springer, 2025.

[Xiao *et al.*, 2020] Yuchen Xiao, Joshua Hoffman, and Christopher Amato. Macro-action-based deep multi-agent reinforcement learning. In *Conference on Robot Learning*, 2020.

[Xu *et al.*, 2021] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and Complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations*, 2021.

[Zhang *et al.*, 2024] Xiyue Zhang, Benjie Wang, and Marta Kwiatkowska. Provable preimage under-approximation for neural networks. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 3–23. Springer, 2024.