A Survey on the Verification of Reinforcement Learning Policies

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

Abstract

Reinforcement learning (RL) is increasingly applied in complex, safety-critical domains, yet the lack of rigorous behavioral guarantees for neural network-based policies remains a major barrier to deployment. Recent advances in policy expressiveness and scale have intensified this challenge, leading to a rapidly growing but conceptually fragmented body of work on RL policy verification. This survey provides a unifying perspective on RL verification methods. We introduce a taxonomy that clarifies relationships among existing approaches along three axes: verification paradigm (formal versus probabilistic), temporal scope (step-wise versus multi-step), and guarantees strength. Beyond taxonomy, we unify underlying theoretical foundations, make implicit assumptions and limitations explicit, and identify emerging directions.

1 Introduction

Over the past decade, reinforcement learning (RL) has emerged as a powerful framework for sequential decision making in complex, high-dimensional environments [Mnih *et al.*, 2015]. Recent advances in expressiveness and scalable training have accelerated interest in deploying RL agents in safety-critical domains such as autonomous driving and energy systems [Lichtlé *et al.*, 2024; Marchesini *et al.*, 2026]. Despite these advances, the deployment of RL systems in high-stakes settings remains limited by the lack of rigorous *behavioral* guarantees. RL policies are typically realized as deep neural networks (DNNs) with complex, non-linear decision boundaries, which can exhibit unsafe behavior under rare conditions, distribution shift, or interaction effects [Szegedy *et al.*, 2013; Aydeniz *et al.*, 2025]. As empirical validation alone cannot rule out such failures [Marchesini *et al.*, 2023; Marzari *et al.*, 2025c], formal and systematic verification becomes essential for safety-critical RL applications.

This requirement has given rise to the application of *formal verification of deep neural networks* methods to RL, which seek to provably certify post-training, that a policy satisfies desired behavioral properties prior to deployment [Liu *et al.*, 2021]. Unlike supervised learning, RL induces

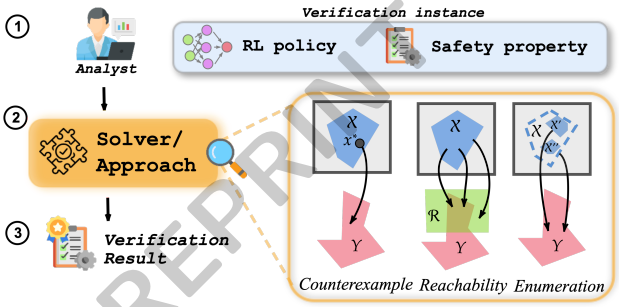

Figure 1: Pipeline for post-training verification of RL policies, highlighting solver-based approaches used to establish safety guarantees.

correctness properties that fundamentally depend on interaction, feedback, and distribution shift under policy execution. An RL policy does not operate on a fixed input distribution, but actively shapes the state distribution it encounters through closed-loop control, leading to error accumulation, rare-event failures, and safety violations that may only arise after long horizons (e.g., a policy that is locally robust at every time step may still violate safety through compounding errors over long horizons) [Sutton and Barto, 2018]. As a result, verification objectives in RL cannot be reduced to input-output robustness alone, but must account for temporal dependence, policy-environment coupling, and deployment-time uncertainty. These characteristics fundamentally shape both what can be verified and which abstractions are meaningful. In detail, we verify properties including safety guarantees (e.g., collision avoidance), robustness guarantees (e.g., resilience to noise, or adversarial perturbations), and reachability guarantees (e.g., task completion). Providing these certificates has become increasingly challenging as RL policies grow in expressiveness and scale.

Fig. 1 summarizes the post-training RL verification pipeline, where a safety property and trained policy are analyzed by a verification solver to either produce a counterexample, establish a reachability-based guarantee, or enumerate unsafe regions of the state space. Drawing on formal methods, control theory, and statistical analysis, a wide range of approaches exists to certify policy behavior after training. These include deterministic techniques that provide sound guarantees via satisfiability (SAT) [Barrett and

Paradigm	Tool / Paper	Approach / Solver	Horizon	Arch.	Rob.	Safe.	Enum.	Guarantee	RL-Ready
Formal	α, β-CROWN [Xu *et al.*, 2020]	Bound Prop. / Linearization	Step-wise	MLP, CNN, RNN	✓	✓	×	Sound & Complete*	Yes
			Step-wise	Transformer	✓	×	×	Sound	No
	nnenum [Bak, 2021], PyRAT [Lemesle *et al.*, 2024]	Abs. Int. / Reachability	Step-wise	MLP, CNN	✓	×	×	Sound	Yes
	NNV 2.0 [Lopez *et al.*, 2023]		Step-wise	MLP, Neural-ODE	✓	✓	×	Sound	Yes
	ModelVerification.jl [Wei *et al.*, 2025]	Bound Prop. / Linearization	Step-wise	MLP, CNN, Neural-ODE	✓	✓	×	Sound & Complete*	Yes
	Marabou (Reluplex extension) [Wu *et al.*, 2024]	SMT / MIP Solving	Step-wise	MLP, CNN	✓	✓	×	Sound & Complete	Yes
	NeuralSAT [Duong *et al.*, 2024]		Step-wise	MLP	✓	✓	×	Sound & Complete	Yes
	U/L-Dist [Kofnov *et al.*, 2025]	Bound Prop. / Reachability	Step-wise	MLP,CNN	✓	×	×	Sound & Complete*	Partial
	Neural Lyapunov [Yang *et al.*, 2024]	Control Theory	Multi-Step	Neural CBF	×	✓	×	Sound & Complete*	Yes
	INVPROP [Kotha *et al.*, 2024]	Bound Prop. / Linearization	Step-wise	MLP	×	✓	✓	Sound	Yes
	Exact Enum [Matoba and Fleuret, 2020]	Preimage Enumeration	Step-wise	MLP	×	✓	✓	Sound & Complete	Yes
Probabilistic	PT-LiRPA [Marzari *et al.*, 2025b]	Probabilistic Linear Relaxation	Step-wise	MLP, CNN	✓	✓	×	Probabilistic	Yes
	Online CBF [Marzari *et al.*, 2025d]	Runtime Monitoring	Multi-Step	MLP + CBF	×	✓	✓	Probabilistic	Yes
	PL-MDP [Hasanbeig *et al.*, 2019]	Temporal Logic	Multi-step	MLP (abstracted)	×	✓	×	Probabilistic	Partial
	FSC-based [Carr *et al.*, 2021]	Synthesis	Step-wise	MLP, RNN	×	✓	×	Probabilistic	Partial
	PREMAP [Zhang *et al.*, 2025]	Bound Prop. / Linearization	Step-wise	MLP	×	✓	✓	Sound / Probabilistic	Partial
	Prob. Enum [Marzari *et al.*, 2025a]	Abs. Int. / Reach. + Sampling	Step-wise	MLP	×	✓	✓	Probabilistic	Yes

* Not for all the methods, completeness holds only when combined with exhaustive branching or linear programming solvers.

Table 1: A unified taxonomy of verification methods for deep RL, highlighting trade-offs between guarantee strength and expressivity across step-wise, multi-step, probabilistic, and enumeration-based formulations.

Tinelli, 2018] and reachability analysis [Lomuscio and Maganti, 2017], probabilistic methods [Valiant, 1984] that quantify risk in stochastic environments, and hybrid neuro-symbolic approaches that combine learned policies with control theory [Zhao *et al.*, 2020]. Despite the growing volume of results, the literature remains conceptually fragmented. Methods are often developed under different assumptions, target different notions of correctness, and operate over different temporal scopes, making it difficult to compare approaches or understand their fundamental relationships.

Contributions. We address this fragmentation by introducing a unified taxonomy of post-training verification methods for neural network-based policies. We organize existing approaches along three core axes: the verification paradigm, the temporal scope of properties, and the guarantees strength of different solvers. Beyond taxonomy, we formalize the main RL verification problems using a unified notation and provide clear, illustrative examples to clarify the underlying ideas. We also analyze the limitations of existing methods and the key trade-offs among expressiveness and soundness. Finally, we discuss open challenges in verifying modern policy architectures, multi-agent systems, and outline directions for extending verification techniques to these settings.

2 Preliminaries

Reinforcement learning is commonly formalized as a Markov decision process (MDP) $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ denote finite state and action spaces, respectively, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ specifies the state transition dynamics, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. An agent's behavior is determined by a policy mapping states to actions, which is parameterized by a DNN for the context of this survey. When deployed, the policy induces a distribution over state-action trajectories.

Most verification methods studied here reason over abstractions of this policy-environment loop, differing in how explicitly they model feedback, uncertainty, and temporal dependence. Understanding which aspects of RL execution are preserved or abstracted away is thus essential for interpreting the guarantees provided by different verification methods.

3 A Unified Taxonomy on RL Verification

While many underlying solver techniques originate from classical neural network verification, their application to RL raises distinct semantic challenges due to feedback, temporal dependence, and interaction with uncertain environments. Table 1 organizes RL verification approaches along three main axes, plus an "RL readiness" one. *Paradigm* characterizes whether the method is formal or probabilistic. *Temporal scope* distinguishes step-wise methods, which analyze a single policy evaluation under bounded inputs, from multi-step methods, which reason about trajectory-level or closed-loop behavior. *Guarantees* indicates the strength of the assurance provided. *RL readiness* then capture practical applicability, indicating whether a method can be applied directly to trained RL policies without substantial abstraction or reformulation.

Reader's Roadmap. This section progresses through increasingly expressive RL verification abstractions. We begin with step-wise analysis (3.1), reasoning about 1-step individual policy decisions. We then consider multi-step formulations (3.2), capturing closed-loop dynamics and temporal properties at higher computational cost. Hence, enumeration-based approaches (3.3) that characterize unsafe regions of the state space. Table 1 provides a unifying reference.

3.1 Step-wise RL Verification

Step-wise RL verification *asks whether a policy selects an unsafe action at a given state*, abstracting away future dynamics. Yet, it underpins most scalable approaches and remains the most mature class of RL verification tools.

Problem Formulation. Consider a trained RL policy represented by a DNN $f : \mathcal{I} \to \mathcal{O}$, where inputs encode the agent's state and outputs represent action values, probabilities, or control commands. The step-wise verification problem is specified by a tuple $\mathcal{T} = \langle f, \mathcal{X}, \mathcal{Y} \rangle$, where the precondition $\mathcal{X} \subseteq \mathcal{I}$ defines a set of admissible inputs (typically

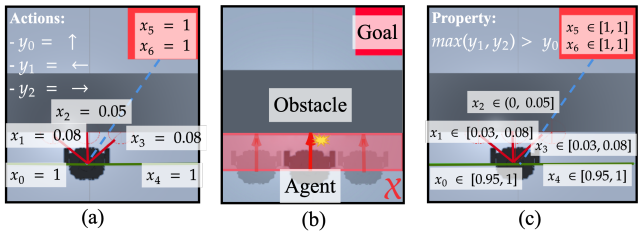

Figure 2: Explanatory behavioral property. (a) Unsafe situation to be verified. (b) Encoding of the unsafe region using $\mathcal{X}$ as geometric set. (c) Property encoded using intervals and predicate on the output.

a neighborhood around a nominal state capturing uncertainty due to sensor noise, or state estimation error). The postcondition $\mathcal{Y} \subseteq \mathcal{O}$ encodes unsafe actions or violations of control constraints. The pair $(\mathcal{X}, \mathcal{Y})$ thus defines a *local behavioral property* to verify for the policy, evaluated independently of the environment [Liu *et al.*, 2021].

Example 1 (Local behavioral property)**.** *Consider the navigation task in Fig. 2, where a policy maps lidar and goal features to actions $\{y_0, y_1, y_2\}$ (forward, left, right). When an obstacle is directly ahead, safety requires that the forward action is never selected under bounded sensor noise. This is encoded by defining $\mathcal{X}$ as an ℓ_∞ ball around the nominal obstacle state and the postcondition $\mathcal{Y}$ as $\max(y_1, y_2) > y_0$. The property is violated if any state in $\mathcal{X}$ induces y_0.*

Notably, in step-wise RL verification, robustness and safety may coincide: $\mathcal{X}$ defines a local perturbation region around a given state, and verifying that no input in $\mathcal{X}$ induces an unsafe action may be equivalent to certifying adversarial robustness of the policy at that state. This equivalence enables a standardized encoding using a specification language such as VNN-LIB [Demarchi *et al.*, 2023], where $\mathcal{X}$ is specified by linear input constraints and $\mathcal{Y}$ by linear output predicates (e.g., action-ranking constraints). Step-wise RL verification can thus be addressed directly by off-the-shelf neural network verifiers without modeling the environment, allowing methods ranging from exact mixed-integer programming [Winston, 2004] to scalable over-approximate techniques such as abstract interpretation [Gehr *et al.*, 2018] and linear relaxation [Zhang *et al.*, 2018] to be applied in a unified and reproducible manner. This equivalence admits two definitions for step-wise verification: an explicit search for a violating input in $\mathcal{X}$ with SAT analysis [Katz *et al.*, 2017; Wu *et al.*, 2024; Tjeng *et al.*, 2017], or the computation of conservative output bounds over all inputs in $\mathcal{X}$ with the reachability-based formulation [Lomuscio and Maganti, 2017; Wang *et al.*, 2018; Kofnov *et al.*, 2025], which we discuss next.

Satisfiability Setting. In SAT-based verification (Def. 1), the policy and the negation of the safety property are encoded as a set of logical constraints [Duong *et al.*, 2024]. Verification reduces to checking whether there exists an input in $\mathcal{X}$ such that the corresponding output lies in $\mathcal{Y}$. When a solution exists, the verifier returns a counterexample, corresponding to the state where the policy violates the behavioral property.

Definition 1 (SAT-Based RL Verification)**.**
Input: *A tuple* $\mathcal{T} = \langle f, \mathcal{X}, \mathcal{Y} \rangle$
Output: $\texttt{violate} \iff \exists x \in \mathcal{X} \mid f(x) \in \mathcal{Y}$

SAT-based methods are particularly valuable for falsification and debugging of learned policies. However, their worst-case computational complexity grows exponentially with network size (non-linearities), limiting scalability. In particular, this is an NP-hard problem [Katz *et al.*, 2017], which motivates the development of over-approximation techniques and branch-and-bound strategies [Bunel *et al.*, 2018].

Reachability Setting. This alternative strategy (Def. 2) propagates bounds through the policy network to compute an over-approximation of the reachable output set $\mathcal{R}(\mathcal{X})$ (e.g., by using interval bound propagation [Lomuscio and Maganti, 2017]). When $\mathcal{R}(\mathcal{X})$ is fully contained within the unsafe output region $\mathcal{Y}$, the policy violates the behavioral property.

Definition 2 (Reachability-Based RL Verification)**.**
Input: *A tuple* $\mathcal{T} = \langle f, \mathcal{X}, \mathcal{Y} \rangle$
Output: $\texttt{violate} \iff \mathcal{R}(\mathcal{X}) \subseteq \mathcal{Y}$

From an RL perspective, reachability-based verification is attractive as it provides guarantees that hold uniformly over regions of the state space, rather than individual counterexamples [Bak, 2021; Lemesle *et al.*, 2024; Lopez *et al.*, 2023]. This is particularly well suited to RL settings where uncertainty in $\mathcal{X}$ reflects epistemic sources (e.g., perception noise, or state estimation error) rather than adversarial perturbations. In such cases, region-level guarantees better align with how policies are deployed and evaluated. The main challenge is over-approximation: conservative reachable-set bounds may be too coarse to certify or refute a property. A common remedy is domain refinement via branch-and-bound, where the input region is recursively split using heuristic criteria [Wang *et al.*, 2018], yielding tighter output bounds on smaller subregions until a conclusive result is obtained (Fig. 3). However, in the worst case, this process requires an exponential number of verification calls, reflecting the NP-hardness of verifying piecewise-linear neural networks.

Modern verifiers also combine linear relaxations (e.g., linear relaxation-based perturbation analysis (LiRPA) [Xu *et al.*,

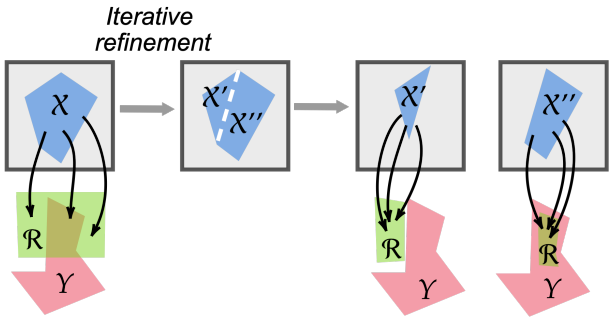

Figure 3: BaB refinement. (Adapted from [Marzari *et al.*, 2024].)

2020]) with refined branching strategies that split either the input space or internal activation patterns (e.g., α, β-CROWN [Zhang *et al.*, 2018; Xu *et al.*, 2021; Wang *et al.*, 2021], ModelVerification.jl [Wei *et al.*, 2025]). In RL, these refinements tighten bounds on policy outputs over local neighborhoods of states, improving the precision of action-level guarantees. However, the resulting scalability-precision trade-off is especially acute for high-dimensional RL policies, motivating a probabilistic perspective.

Probabilistic Step-Wise Verification. To improve scalability, probabilistic methods estimate the likelihood that an RL policy selects an unsafe action under uncertainty, rather than requiring safety for all inputs in $\mathcal{X}$. Early approaches, such as PROVEN [Weng *et al.*, 2019] and randomized smoothing [Cohen *et al.*, 2019], bound violation probabilities under input perturbation distributions but rely on restrictive noise assumptions or focus on classification. More recently, `PT-LiRPA` [Marzari *et al.*, 2025b] extends linear relaxation with probabilistic reasoning to bound the probability of unsafe actions within an uncertainty region without assuming a specific noise model, enabling scalable verification of RL policies. Despite their maturity and strong tooling support, step-wise methods cannot reason about closed-loop execution or long-horizon effects, motivating the multi-step and enumeration-based paradigms discussed next.

3.2 Multi-step RL Verification

Multi-step RL verification *asks whether a policy can incur unsafe behavior over time*. Such properties are inherently temporal and depend on closed-loop trajectories, but their expressiveness comes at a high computational cost: modeling dynamics, temporal specifications, and long-horizon uncertainty significantly limits scalability. As a result, existing approaches rely on abstractions and relaxations, giving rise to several distinct multi-step verification paradigms.

Problem Formulation. Multi-step verification generalizes step-wise analysis by reasoning about the interaction between the policy and the environment over finite or infinite horizons. The objective is to determine whether any closed-loop trajectory can reach an unsafe state, shifting the object of analysis from the policy in isolation to the policy-environment system. A trajectory $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_t\}$ is simply a sequence of state-action interactions. When the transition dynamics are known or can be conservatively approximated, multi-step RL verification is a trajectory-level reachability problem that verifies a *trajectory-level behavioral property*.

Example 2 (Trajectory-level behavioral property)**.** *Consider the navigation task in Fig. 2, where an agent follows a fixed policy in a known environment with deterministic dynamics. Let $\mathcal{S}_{\text{unsafe}} \subseteq \mathcal{S}$ denote collision states. Given an initial set $\mathcal{X}_0 \subseteq \mathcal{S}$, safety requires that no closed-loop trajectory reaches $\mathcal{S}_{\text{unsafe}}$, i.e., $\forall s_0 \in \mathcal{X}_0,\ \forall t \geq 0,\ s_t \notin \mathcal{S}_{\text{unsafe}}$. The property is violated if some $s_0 \in \mathcal{X}_0$ reaches an unsafe state.*

A representative approach is the reachability analysis of neural feedback loops proposed by [Everett *et al.*, 2021], which embeds the policy within a dynamical system and propagates over-approximations of reachable states over time using linearization-based abstractions. While effective, the resulting conservativeness grows with the time horizon, and scalability is limited by the need for domain splitting and refinement to balance precision and computational tractability.

Multi-step RL verification also admits multiple definitions, which differ in their treatment of dynamics, temporal specifications, and uncertainty, and which we review next.

Labeled MDPs and Temporal Logic Verification. Many long-horizon RL objectives are naturally expressed using temporal logics such as linear temporal logic (LTL) [Pnueli, 1977], which enable explicit specification of properties over trajectories [Carr *et al.*, 2021]. To reason about these, the environment is modeled as a labeled MDP, where states are annotated with atomic propositions via a labeling function L. The LTL specification φ is translated into an automaton M, and verification is performed over the product of the policy-induced MDP and the automaton. Combined with reachability analysis and linearization-based over-approximations, this yields a conservative characterization of trajectories that may violate φ. This formulation substantially increases verification complexity, as the product MDP-automaton can be exponentially larger than the original system, making formal verification intractable for high-dimensional or long-horizon settings. Formally, given an initial state set $\mathcal{X}_0$ and an LTL formula φ, we define this type of verification in Def. 3.

Definition 3 (LTL-BASED MULTI-STEP RL VERIFICATION)**.**
Input*: A tuple $\mathcal{T} = \langle f, M, \mathcal{X}_0, \varphi \rangle$*
Output*: $\texttt{violate} \iff \exists \tau \in \langle f, M, \mathcal{X}_0 \rangle \mid \tau \not\models \varphi$*

Neuro-Symbolic Certificates. On a different direction, neuro-symbolic approaches combine RL policies with symbolic safety certificates from control theory, most notably control barrier functions and Lyapunov functions [Ames *et al.*, 2016]. Shielding-based neuro-symbolic approaches can also guarantee safety at runtime, but have so far been demonstrated primarily in small-scale, discrete RL domains [Melcer *et al.*, 2022]. In these paradigms, the policy proposes actions, while the certificate enforces safety by modifying or rejecting unsafe actions at runtime, enabling long-horizon guarantees.

When available analytically, such certificates provide strong model-based guarantees over continuous-time trajectories. In many RL settings, however, the safe set or certificate must be learned from data, leading to neural barrier or Lyapunov functions whose verification becomes a problem in its own right. Recent work focuses on certifying that learned certificates satisfy invariance or stability conditions over the relevant state space [Hu *et al.*, 2025; Yang *et al.*, 2024]. While effective, neuro-symbolic methods shift rather than eliminate the verification burden, inheriting the scalability challenges of neural network verification and introducing additional modeling assumptions. Probabilistic variants further relax worst-case guarantees in favor of high-confidence safety certificates over long horizons [Marzari *et al.*, 2025d].

Probabilistic Multi-Step Verification. Even in the multi-step case, probabilistic verification relaxes worst-case guarantees by estimating the likelihood that an RL policy satis-

fies a temporal property over closed-loop trajectories. These methods achieve scalability by sampling trajectories or propagating probabilistic bounds through labeled MDPs, providing formal confidence guarantees rather than absolute safety. A representative example is the probabilistic reachability framework of [Hasanbeig *et al.*, 2019], which computes bounds on the probability of satisfying temporal specifications in discrete-time stochastic systems.

Position within the taxonomy. Multi-step verification methods extend step-wise verification along the temporal axis. They expose a fundamental trade-off between expressivity and scalability: richer behavioral guarantees require reasoning about policy-environment interaction over time, but quickly lead to state-space explosion. Understanding and managing this trade-off is central to RL verification and motivates the emerging problem we discuss next.

3.3 Emerging Verification Formulations

An emerging *enumeration-based verification* paradigm (or neural network preimage analysis) (Def. 4) *asks where in state space does the policy systematically fail?* In RL, where failures often arise from structured regions rather than isolated states, such region-level information supports policy debugging, targeted data collection, retraining, and deployment-time mitigation strategies [Matoba and Fleuret, 2020; Marzari *et al.*, 2023; Kotha *et al.*, 2024].

Problem Formulation. Given a policy network and an unsafe output predicate, enumeration-based verification computes the set of inputs that lead to unsafe actions. This can be formally defined as follows:

Definition 4 (ALLDNN-VERIFICATION FOR RL)**.**
Input: *A tuple* $\mathcal{T} = \langle f, \mathcal{X}, \mathcal{Y} \rangle$
Output: $\Gamma(\mathcal{T}) = \{x \in \mathcal{X} \mid f(x) \in \mathcal{Y}\}$

Exact enumeration methods seek to compute $\Gamma(\mathcal{T})$ (i.e., the set of states that violate a property) precisely, for example, as a union of polytopes whose volume can be estimated. Recalling our navigation example, $\Gamma(\mathcal{T})$ may be represented by the red region of states highlighted in Fig. 2 (b).

In detail, enumeration-based approaches typically rely on branch-and-bound procedures that recursively partition the input space or activation patterns [Bunel *et al.*, 2018] as illustrated in Fig. 4. While enabling fine-grained safety analysis [Matoba and Fleuret, 2020], their worst-case complexity grows exponentially with network depth and input dimension, rendering exact enumeration #P-hard [Marzari *et al.*, 2023] and motivating the need for novel perspectives.

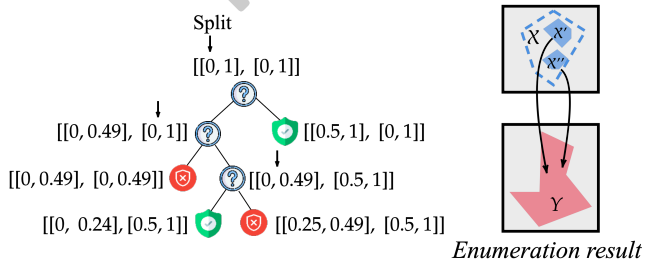

Figure 4: DNN preimage bound computation via BaB strategy. (Adapted from [Marzari *et al.*, 2025a].)

Enumeration Approaches. A first line of work proposes approximate preimage-bounding techniques based on linearization, which trade exactness for efficiency. Over-approximate methods, such as the backward analysis in [Kotha *et al.*, 2024], compute conservative outer bounds that contain all unsafe inputs, but still result in limited scalability for large RL policies. To enable scalable solutions, recent frameworks either combine over- and under-approximation analyses, such as PREMAP [Zhang *et al.*, 2025], or adopt a probabilistic perspective, as in CountinProVe, ϵ-ProVe, and RF-ProVe [Marzari *et al.*, 2023; Marzari *et al.*, 2024; Marzari *et al.*, 2025a]. These approaches improve the scalability of preimage analysis for large RL policies, enabling practical region-level behavioral analysis.

Position within the taxonomy. Knowing *where* unsafe decisions happen significantly pushes verification in the *RL readiness* direction of our taxonomy. For example, preimage information can be used to: (i) guide additional training or data collection in high-risk regions, (ii) synthesize recovery or shielding mechanisms that avoid unsafe states, and (iii) assess the severity of residual risk by estimating the size or probability mass of unsafe regions. These use cases go beyond what binary verification can provide and align enumeration-based methods with practical RL workflows.

4 Verification Tool Benchmarking & Selection

Verification of RL policies spans objectives ranging from local robustness under perception uncertainty to safety guarantees over closed-loop trajectories. Because these objectives

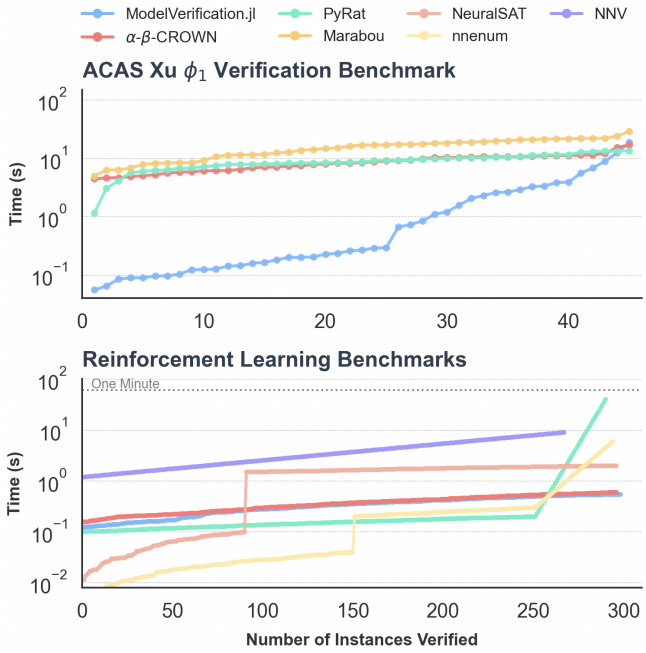

Figure 5: Runtime of RL verification tools as number of verified properties (instances, x-axis) and computation time (y-axis).

Method	Task	Coverage	Error (%)	Time
Exact	VCAS	100%	0%	6352.21s
PREMAP	VCAS	**90.8%**	0%	12.8s
ε-ProVe	VCAS	90.48%	0.02%	0.65s
RF-ProVe	VCAS	90.5%	0.06%	0.3s
PREMAP	Cartpole	75.5%	0%	32.37s
ε-ProVe	Cartpole	76.47%	0.27%	2s
RF-ProVe	Cartpole	**76.8%**	0.3%	4.5s
PREMAP	Lunarlander	75.1%	0%	85.42s
ε-ProVe	Lunarlander	76.51%	0.5%	12.2s
RF-ProVe	Lunarlander	**75.63%**	0.3%	59s
PREMAP	Dubin's rejoin	78.7%	0%	656.47s
ε-ProVe	Dubin's rejoin	85.02%	0.3%	260.23s
RF-ProVe	Dubin's rejoin	**90.08%**	0.3%	66s

Table 2: Preimage evaluation from [Marzari *et al.*, 2025a]. We show trade-offs between exactness, scalability, and computational efficiency.

differ in temporal scope, guarantee strength, and scalability, no single verification technique is universally applicable.

Fig. 5 and Table 2 complement our taxonomy with an empirical comparison of methods from (i) the international verification of neural networks competition (VNN-COMP) [Brix *et al.*, 2023] and (ii) preimage approximation approaches from [Marzari *et al.*, 2025a]. The benchmarks focus on open-loop sensor-noise robustness for discrete-action RL policies implemented as fully connected ReLU networks, trained on ACAS [Julian *et al.*, 2016], Dubin's rejoin, CartPole, and LunarLander from OpenAI Gym [Brockman *et al.*, 2016] tasks.

Peformance Analysis. Many methods perform similarly on ACAS Xu, but their scalability diverges on RL benchmarks. For step-wise verification, bound-propagation methods consistently outperform exact solvers as the number of instances grows, while relaxation-based approaches better handle the dimensionality and repetition of RL policy analysis. α, β-CROWN and ModelVerification.jl exhibit promising scaling across benchmarks, highlighting the effectiveness of tight linear relaxations and efficient branching heuristics.

For multi-step verification, the lack of standardized benchmarks remains an open challenge, reflecting fundamental difficulties in modeling dynamics, uncertainty, and temporal specifications rather than an omission of this survey. Nevertheless, scalability is the dominant bottleneck: reasoning over closed-loop trajectories rapidly expands the state space, limiting exact or worst-case guarantees. Control-theoretic methods based on Lyapunov or control barrier functions provide strong guarantees when applicable, but rely on additional modeling assumptions that constrain generality. Finally, enumeration-based methods provide the strongest guarantees by explicitly characterizing unsafe regions, but their cost grows rapidly with network depth and input dimension. Exact approaches such as [Matoba and Fleuret, 2020] do not scale to larger networks or complex RL specifications, often producing highly fragmented polytope representations that exhaust memory and hinder interpretability. Approximate methods [Zhang *et al.*, 2025], particularly probabilistic approaches such as ϵ-ProVe and RF-ProVe, substantially improve scalability by relaxing soundness in exchange for compact, informative representations better suited to downstream tasks such as safe recovery and explanation.

These results suggest that scalability, architectural coverage, and tooling maturity are tightly coupled dimensions rather than independent factors.

Verification Tool Selection. Tool selection depends on whether a method's assumptions and guarantees match the RL setting. In the following, we summarize practical guidelines for tool selection in RL: (i) *Local robustness or action-level safety*: Over-approximate reachability-based methods are often effective when certifying safety over neighborhoods of states, especially under epistemic uncertainty. (ii) *Falsification and debugging*: SAT-based solvers are well suited when concrete counterexamples are desired to diagnose policy failure modes. (iii) *High-dimensional spaces or large-scale evaluation*: Relaxation-based and probabilistic methods tend to scale better when verification must be performed repeatedly across many states [Brix *et al.*, 2023]. (iv) *Region-level risk analysis*: Enumeration-based tools are most appropriate when the goal is to localize and quantify unsafe regions of the state space rather than obtain a binary verdict. (v) *Long-horizon or deployment-oriented safety*: Multi-step or neuro-symbolic approaches are preferable when guarantees must hold over closed-loop trajectories.

Taken together, the taxonomy and empirical results provide practical guidance for selecting verification methods that are well matched to the structure, scale, and verification objectives of a given RL application, while making explicit the assumptions under which each method is most effective.

5 Open Challenges

Several issues remain in post-training RL verification: *history-dependent policies*, typically implemented via recurrent neural networks (RNNs), *multi-agent RL* (MARL), and orthogonal challenges spanning modern policy architectures and temporal scopes. These issues highlight a fundamental mismatch between neural network verification and RL systems, where correctness depends on policy-induced histories, interactions, and closed-loop dynamics.

Verifying Recurrent Policies. RNNs in partially observable tasks induce history-dependent behavior via latent hidden states, making safety and robustness depend on the set of feasible hidden states. Most existing methods are step-wise and memoryless, reducing RNN verification to bounded input perturbations over a fixed unrolling horizon; an abstraction that captures local robustness but fails to reflect uncertainty arising from the agent's internal memory.

Problem Formulation. Following our unified taxonomy, we reason over uncertainty in the policy's internal state (Def 5). Given an RNN-based policy f, a set of histories $\mathcal{H}$, a current observation o, and a postcondition $\mathcal{Y}$ encoding unsafe behavior as in step-wise SAT verification (Def. 1), the verification problem can be formulated to ask whether there exists a feasible hidden state $h \in \mathcal{H}^*$, corresponding to a realizable interaction history, such that $f(h, o) \in \mathcal{Y}$. This formulation inherently spans multi-step temporal reasoning.

Definition 5 (RNN-VERIFICATION for RL)**.**
Input*: A tuple* $\mathcal{T} = \langle f, \mathcal{H}, o, \mathcal{Y} \rangle$
Output*:* `violate` $\iff \exists h \in \mathcal{H}^* \mid f(h, o) \in \mathcal{Y}$

Prior work on this relies on discretizing RNNs into finite-state abstractions [Carr *et al.*, 2021] or on reachability analysis assuming known dynamics [Everett *et al.*, 2021]. These approaches have strong modeling assumptions and may suffer from poor scalability, as the abstract state space grows exponentially with the hidden-state dimension and planning horizon. A key challenge is that the feasible hidden-state set $\mathcal{H}^*$ is highly structured and policy-dependent. Standard reachability or interval-based abstractions typically over-approximate this set and may admit unrealizable hidden states, as they lack access to a *feasibility oracle* capable of distinguishing configurations induced by realizable interaction histories from infeasible ones. Hence, the abstract feasibility space may contain spurious counterexamples or yield overly conservative verification guarantees. Additionally, characterizing unsafe hidden states exactly is computationally intractable, subsuming #P-hard counting and preimage problems even in the feedforward case [Marzari *et al.*, 2023].

This exposes a mismatch, where behavior depends on policy-induced histories rather than single inputs. Open questions include: *how to define verification objectives faithful to history-dependent decision making in RL? How to characterize feasible histories without explicit enumeration or overly conservative over-approximation?*

Progress will require frameworks that reason over feasible histories, incorporate probabilistic guarantees when worst-case analysis is intractable, and better align formal methods with RL-specific uncertainty and execution semantics.

Verifying Multi-Agent Systems. In MARL, verification depends on the learning and execution paradigm (e.g., cooperative or adversarial settings, centralized or decentralized policies), and coordination assumptions. These challenges are also compounded by the exponential growth of the joint state-action space with the number of agents. Hence, we focus on the common fully cooperative setting with decentralized execution, typically realized via centralized training with decentralized execution [Papoudakis *et al.*, 2021]. A team of $\mathcal{N}$ agents coordinate to achieve a shared objective, where each agent i executes an RNN-based policy f_i based solely on its local observation o_i and hidden state $h_i \in \mathcal{H}_i$.

Problem Formulation. We formalize cooperative MARL verification (Def 6). Given agent-specific verification problems $\mathbf{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_\mathcal{N}\}$, with $\mathcal{T}_i = \langle f_i, \mathcal{H}_i, o_i, \mathcal{Y}\rangle$, a team-level violation occurs if at least one agent admits a feasible hidden state $h_i \in \mathcal{H}_i^*$ such that its local policy violates the cooperative behavioral property (encoded in $\mathcal{Y}$). This formulation operates over a multi-step horizon induced by decentralized interaction.

Definition 6 (RNN-VERIFICATION for cooperative MARL)**.**
Input*:* $\mathbf{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_\mathcal{N}\}$*, with* $\mathcal{T}_i = \langle f_i, \mathcal{H}_i, o_i, \mathcal{Y}\rangle$
Output*:* `violate` $\iff \exists i \in \{1, \ldots, \mathcal{N}\}$, $\exists h_i \in \mathcal{H}_i^* \mid f_i(h_i, o_i) \in \mathcal{Y}$

Extending this to adversarial or mixed settings introduces additional challenges, as verification must account for strategic opponents and worst-case behaviors, fundamentally altering semantics and tractability. Under decentralized execution, agent-wise verification can be performed independently, with system-level safety determined by the worst-performing agent, enabling parallel analysis without added conservatism. However, this decomposition raises open questions, such as: *when is agent-wise verification sufficient to guarantee team-level safety? How should verification account for implicit coordination encoded in agents' recurrent hidden states?*

Progress will require frameworks that reason about interaction-induced uncertainty, emergent behavior, and history-dependent coordination while balancing soundness and scalability.

Orthogonal Challenges. Beyond these settings, several orthogonal challenges further complicate RL verification across paradigms, objectives, and temporal horizons.

- *Environment uncertainty and closed-loop verification*. Most methods assume known or accurately approximated dynamics, whereas real-world RL systems operate under model mismatch, disturbances, and non-stationarity, raising questions about whether guarantees should apply to the policy, the closed-loop system, or uncertainty sets over environments, and how such guarantees can be provided tractably.
- *Long-horizon reasoning and rare events*. Many safety-critical failures arise from low-probability events over long horizons, which worst-case methods either miss or over-approximate excessively, making it difficult to capture risk accumulation, infinite-horizon behavior, and rare-event probabilities at scale.
- *Modern architectures* such as Transformers, the verification challenge is twofold: architectural, because attention and softmax violate piecewise-linear assumptions, and semantic, because these models reason over variable-length histories via attention rather than explicit recurrence.

As a result, existing abstractions struggle to scale and to faithfully capture temporal context, leaving this gap an open direction. These challenges highlight that RL verification is not merely a scaling problem but a conceptual one: correctness must be defined in terms of interactions, temporal dependence, and decentralized decisions.

6 Conclusion

This survey presented a unified view of post-training RL verification, organizing a fragmented literature along three axes: verification paradigm, objective, and temporal scope. Our taxonomy clarifies the assumptions, guarantees, and scalability trade-offs of existing methods, highlighting the tension between strong guarantees and computational tractability.

More broadly, our analysis shows that RL verification requires a conceptual shift beyond classical neural network verification, as correctness depends on closed-loop interaction, temporal dependence, and increasingly on history-dependent and decentralized decision making. Addressing these challenges (particularly for recurrent policies, multi-agent systems, long horizons, and modern architectures) will require verification frameworks that better align with RL execution semantics while balancing soundness and scalability.

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.

References

[Ames *et al.*, 2016] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62:3861–3876, 2016.

[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.

[Bak, 2021] Stanley Bak. nnenum: Verification of relu neural networks with optimized abstraction refinement. In *NASA Formal Methods: International Symposium*, 2021.

[Barrett and Tinelli, 2018] Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of model checking*. Springer, 2018.

[Brix *et al.*, 2023] Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T Johnson. Vnn-comp 2023: Summary and results. *arXiv preprint arXiv:2312.16760*, 2023.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[Bunel *et al.*, 2018] Rudy R Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and Pawan K Mudigonda. A unified view of piecewise linear neural network verification. *NeurIPS*, 31, 2018.

[Carr *et al.*, 2021] Steven Carr, Nils Jansen, and Ufuk Topcu. Verifiable rnn-based policies for pomdps under temporal logic constraints. In *IJCAI*, 2021.

[Cohen *et al.*, 2019] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

[Demarchi *et al.*, 2023] Stefano Demarchi, Dario Guidotti, Luca Pulina, and Armando Tacchella. Supporting standardization of neural networks verification with vnn-lib and coconet. In *Formal Methods for ML-Enabled Autonomous Systems*, 2023.

[Duong *et al.*, 2024] Hai Duong, ThanhVu Nguyen, and Matthew Dwyer. A dpll(t) framework for verifying deep neural networks. *arXiv preprint arXiv:2307.10266*, 2024.

[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.

[Gehr *et al.*, 2018] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, et al. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *IEEE SP*, pages 3–18, 2018.

[Hasanbeig *et al.*, 2019] Mohammadhosein Hasanbeig, Yiannis Kantaros, Alessandro Abate, et al. Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees. In *CDC*, 2019.

[Hu *et al.*, 2025] Hanjiang Hu, Yujie Yang, Tianhao Wei, and Changliu Liu. Verification of neural control barrier functions with symbolic derivative bounds propagation. In *Conference on Robot Learning*, pages 1797–1814, 2025.

[Julian *et al.*, 2016] Kyle D Julian, Jessica Lopez, Jeffrey S Brush, Michael P Owen, and Mykel J Kochenderfer. Policy compression for aircraft collision avoidance systems. In *IEEE/AIAA Digital Avionics Systems Conference*, pages 1–10, 2016.

[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 *CAV*, 2017.

[Kofnov *et al.*, 2025] Andrey Kofnov, Daniel Kapla, Ezio Bartocci, and Efstathia Bura. Exact upper and lower bounds for the output distribution of neural networks with random inputs. In *ICML*, 2025.

[Kotha *et al.*, 2024] Suhas Kotha, Christopher Brix, J Zico Kolter, et al. Provably bounding neural network preimages. *NeurIPS*, 36, 2024.

[Lemesle *et al.*, 2024] Augustin Lemesle, Julien Lehmann, and Tristan Le Gall. Neural network verification with pyrat. *arXiv preprint arXiv:2410.23903*, 2024.

[Lichtlé *et al.*, 2024] Nathan Lichtlé, Eugene Vinitsky, Matthew Nice, et al. From sim to real: A pipeline for training and deploying traffic smoothing cruise controllers. *IEEE Transactions on Robotics*, pages 4490–4505, 2024.

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

[Lomuscio and Maganti, 2017] Alessio Lomuscio and Lalit Maganti. An approach to reachability analysis for feed-forward relu neural networks. *arXiv preprint arXiv:1706.07351*, 2017.

[Lopez *et al.*, 2023] Diego Manzanas Lopez, Sung Woo Choi, Hoang-Dung Tran, and Taylor T Johnson. Nnv 2.0: the neural network verification tool. In *CAV*, pages 397–412. Springer, 2023.

[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, 2023.

[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.*, 2023] Luca Marzari, Davide Corsi, Ferdinando Cicalese, and Alessandro Farinelli. The #dnn-verification problem: Counting unsafe inputs for deep neural networks. In *IJCAI*, pages 217–224, 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, 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, 2025.

[Marzari *et al.*, 2025b] Luca Marzari, Ferdinando Cicalese, and Alessandro Farinelli. Probabilistically tightened linear relaxation-based perturbation analysis for neural network verification. *Journal of Artificial Intelligence Research*, 84, 2025.

[Marzari *et al.*, 2025c] 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.*, 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*, pages 9630–9637, 2025.

[Matoba and Fleuret, 2020] Kyle Matoba and François Fleuret. Exact preimages of neural network aircraft collision avoidance systems. In *Machine Learning for Engineering Modeling, Simulation, and Design*, 2020.

[Melcer *et al.*, 2022] Daniel Melcer, Christopher Amato, and Stavros Tripakis. Shield decentralization for safe multi-agent reinforcement learning. In *NeurIPS*, pages 13367–13379, 2022.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[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 *NeurIPS*, 2021.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. *Symposium on Foundations of Computer Science*, 1977.

[Sutton and Barto, 2018] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

[Szegedy *et al.*, 2013] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[Tjeng *et al.*, 2017] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *ICLR*, 2017.

[Valiant, 1984] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Wang *et al.*, 2018] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Formal security analysis of neural networks using symbolic intervals. In *USENIX Security Symposium*, pages 1599–1614, 2018.

[Wang *et al.*, 2021] Shiqi Wang, Huan Zhang, Kaidi Xu, et al. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *NeurIPS*, 34, 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 *CAV*, pages 395–408. Springer, 2025.

[Weng *et al.*, 2019] Lily Weng, Pin-Yu Chen, Lam Nguyen, et al. Proven: Verifying robustness of neural networks with a probabilistic approach. In *ICML*, 2019.

[Winston, 2004] Wayne L Winston. *Operations research: applications and algorithm*. Thomson Learning, 2004.

[Wu *et al.*, 2024] Haoze Wu, Omri Isac, Aleksandar Zeljić, et al. Marabou 2.0: a versatile formal analyzer of neural networks. In *CAV*, pages 249–264, 2024.

[Xu *et al.*, 2020] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33:1129–1141, 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 *ICLR*, 2021.

[Yang *et al.*, 2024] Lujie Yang, Hongkai Dai, Zhouxing Shi, Cho-Jui Hsieh, Russ Tedrake, and Huan Zhang. Lyapunov-stable neural control for state and output feedback: a novel formulation. In *ICML*, 2024.

[Zhang *et al.*, 2018] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *NeurIPS*, 31, 2018.

[Zhang *et al.*, 2025] Xiyue Zhang, Benjie Wang, Marta Kwiatkowska, and Huan Zhang. Premap: A unifying preimage approximation framework for neural networks. *JMLR*, 26(133):1–44, 2025.

[Zhao *et al.*, 2020] Hengjun Zhao, Xia Zeng, Taolue Chen, and Zhiming Liu. Synthesizing barrier certificates using neural networks. In *International Conference on Hybrid Systems: Computation and Control*, 2020.