Research Directions: Cyber-Physical Systems

www.cambridge.org/cbp

Results

Cite this article: Krish V, Mata A, Bak S, Hobbs K, and Rahmati A (2024). Provable observation noise robustness for neural network control systems. *Research Directions: Cyber-Physical Systems*. **2**, e1, 1–12. https://doi.org/10.1017/cbp.2023.5

Received: 30 June 2023
Revised: 3 November 2023
Accepted: 29 November 2023

Keywords:
neural network robustness; safety-critical control systems

Corresponding author:
Veena Krish; Email: kveena@cs.stonybrook.edu

Research Directions

CAMBRIDGE UNIVERSITY PRESS

Provable observation noise robustness for neural network control systems

Veena Krish[1], Andrew Mata[1], Stanley Bak[1], Kerianne Hobbs[2] and Amir Rahmati[1]

[1]Stony Brook University, Stony Brook NY, USA and [2]Air Force Research Lab, Wright-Patterson Air Force Base, Ohio, USA

Abstract

Neural networks are vulnerable to adversarial perturbations: slight changes to inputs that can result in unexpected outputs. In neural network control systems, these inputs are often noisy sensor readings. In such settings, natural sensor noise – or an adversary who can manipulate them – may cause the system to fail. In this paper, we introduce the first technique to provably compute the minimum magnitude of sensor noise that can cause a neural network control system to violate a safety property from a given initial state. Our algorithm constructs a tree of possible successors with increasing noise until a specification is violated. We build on open-loop neural network verification methods to determine the least amount of noise that could change actions at each step of a closed-loop execution. We prove that this method identifies the unsafe trajectory with the least noise that leads to a safety violation. We evaluate our method on four systems: the Cart Pole and LunarLander environments from OpenAI gym, an aircraft collision avoidance system based on a neural network compression of ACAS Xu, and the SafeRL Aircraft Rejoin scenario. Our analysis produces unsafe trajectories where deviations under 1% of the sensor noise range make the systems behave erroneously.

Introduction

Real-world systems are increasingly adopting machine learning (ML) methods to make control decisions. Neural networks, considered universal function approximators, are now widely used to represent control policies of autonomous systems (Miller, 1989; Morel et al., 2001; Palancar et al., 1998; Shin and Kim, 2004). However, neural networks are known to be vulnerable to *adversarial examples*: small perturbations of observations that can cause incorrect predictions. These vulnerabilities were originally identified in perception systems (Goodfellow et al., 2014) and later extended to other domains such as audio processing (Carlini and Wagner, 2018), natural language processing (Cheng et al., 2020), and reinforcement learning (RL) systems (Lin et al., 2017; Pinto et al., 2017). Given the rapid expansion of ML into real-world systems, the potential lack of robustness to adversarial examples is concerning when such networks are to be applied in safety-critical domains (Eykholt et al., 2018). Numerous methods have emerged to efficiently identify these perturbations, which can be synthesized without knowing details about the target system (Papernot et al., 2016), without the exact input example (Moosavi-Dezfooli et al., 2017), and in real-time (Gong et al., 2019). In response, provable defense mechanisms have been developed for perception systems based on *open-loop* neural network verification (Albarghouthi, 2021; Bak et al., 2021; Liu et al., 2019; Xiang et al., 2018). These methods examine a single execution of a neural network and can prove, in certain cases, that no images within some finite noise around a given input image are misclassified. However, for closed-loop systems where a network controller interacts with an environment repeatedly, no such methods exist to identify an unsafe trajectory with provably minimum noise.

In this work, we strive to find the minimum-noise sequence that can cause a closed-loop neural network control system to violate a safety specification. Our technique guarantees that all executions with observation noise below this threshold can never cause a safety violation. An example of such a trajectory in the context of an air-to-air collision avoidance system we analyze in our evaluation is shown in Figure 1. Due to erroneous position readings (shown in gray) caused by injected observation noise, the aircraft under control (starting at the top left) is maneuvered into a collision with an intruder aircraft at the bottom-right corner of the figure. Our approach builds upon open-loop neural network verification methods to determine guaranteed minimum-noise thresholds at each step, exploring and expanding a tree search of possible system states until an unsafe trajectory is identified. This approach is adaptable as it (1) supports any neural network verification tool that can return whether an output can be reached within a given noise region around an input, and (2) supports any norm for defining the *least magnitude* of noise, assuming that the underlying verification tool can support it as well. We also prove this produces the minimum-noise adversarial trajectory from a given start state. We demonstrate our approach on four systems of varying complexity and domains: (1) the Cart

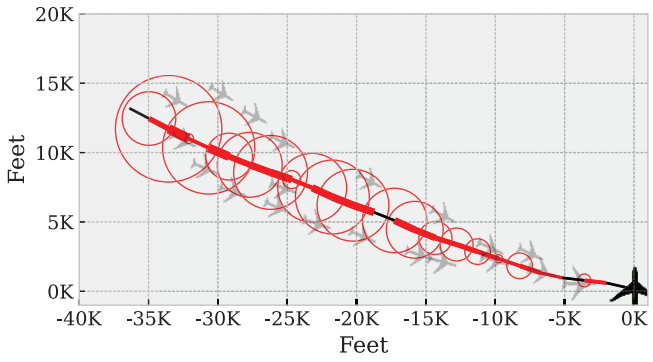

Figure 1. Example of a minimum-noise trajectory, starting within a safe region (far away from an oncoming aircraft) and ending at a near mid-air collision state at the bottom-right corner of the figure. States shown in gray correspond to the adversarial values that caused an incorrect control output. The segments of the trajectory shown in red also correspond to the timesteps that required observation noise to cause the intended turn advisory. The largest magnitude of noise, represented by the large circle around (−34K ft, 12K ft) is the least that is required to realize this particular collision.

Pole benchmark system used for reinforcement learning problems (Brockman et al., 2016), (2) the Lunar Lander system (Brockman et al., 2016), (3) an air-to-air collision avoidance system based on a proposed neural network compression of the Airborne Collision System X Unmanned (ACAS Xu) system (Owen et al., 2019), and (4) the SafeRL aircraft formation rejoin task (Ravaioli et al., 2022). Comparisons with existing work on best-effort adversarial attacks for RL systems show that our method finds trajectories that require less noise and fewer steps to succeed.

The main contributions of this paper are as follows:

- We introduce the first general-purpose approach to identify the largest amount of sensor noise that can be tolerated by a discrete action neural network controller before a safety property is violated.
- Our approach returns the exact sequence of adversarial perturbations that can guide the system from a given initial state to a final unsafe state with the least amount of noise.
- We evaluate our approach on four systems and show sequences of noisy sensed observations that can guide the systems to unsafe states. These sequences were found with sensor noise under 1% of the range of each variable across all possible observations. We compare these findings with recent methods for attacking reinforcement learning systems.

Background

Our work leverages advances in open-loop neural network verification, neural network control systems, and Bayesian optimization.

Open-loop neural network verification. Given a neural network $f_{NN} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, an input set $X \subseteq \mathbb{R}^n$, and an unsafe set of outputs $U \subseteq \mathbb{R}^m$, the *open-loop NN verification problem* is to prove that for all allowed inputs $x \in X$, the network output is never in the unsafe set, $f_{NN}(x) \notin U$.

A wide variety of algorithms (Albarghouthi, 2021; Liu et al., 2019; Xiang et al., 2018) and tools (Bak et al., 2021) have been recently developed to solve this problem. Our approach requires that the tool produce concrete counterexamples if verification fails: a specific $x \in X$ such that $f_{NN}(x) \in U$ (most tools support this). Recent Safe RL methods build on neural network verification to estimate unreliable starting positions or forecast regions of the state space where the network may act inconsistently. However, open-loop neural network verification is insufficient to provide provable safety for neural network control systems' robustness to noise – the focus of this work – as these systems repeatedly invoke the neural network as they interact with an environment. We are not solely interested in finding perturbations that produce unsafe actions due to a single network invocation; rather, we are interested in computing sequences of noise at each step that causes the system to eventually violate its specification.

Neural network control system. We consider networks with discrete action spaces so that the network output is mapped to one of a finite set of actions $\mathbb{A}$, given by a function, $g_{ACT} : \mathbb{R}^m \rightarrow \mathbb{A}$. A neural network control system (NNCS) interacts with an environment, modeled with a function $h_{ENV} : \mathbb{R}^n \times \mathbb{A} \rightarrow \mathbb{R}^n$. Given a state of the system $x_i \in \mathbb{R}^n$, the one-step semantics compute the state after one discrete time step as $x_{i+1} = h_{ENV}(x_i, g_{ACT}(f_{NN}(x_i)))$. We also define noisy one-step semantics where, in addition to x_i, we are given a sensor noise vector $\delta_i \in \mathbb{R}^n$ and compute the next state as $x_{i+1} = h_{ENV}(x_i, g_{ACT}(f_{NN}(x_i + \delta_i)))$. A neural network control system is also associated with either a specific initial state x_0 or a set of possible initial states $\mathcal{I} \subseteq \mathbb{R}^n$. Given an initial state $x_0 \in \mathcal{I}$, the one-step semantics can be repeated to compute the system state up to any number of steps, and specific systems include a *stopping condition* such as a maximum number of time steps or when the state enters some termination set.

In this work, we consider system trajectories subject to sensor noise. A *noisy trajectory*, or simply *trajectory* $\mathcal{T}$ of an NNCS is a finite sequence of states and noise vectors, written as $\mathcal{T} = x_0 \xrightarrow{\delta_0} x_1 \xrightarrow{\delta_1} \ldots \xrightarrow{\delta_{n-1}} x_n$, such that x_0 is an initial state, x_n meets the stopping condition, and each x_i in the sequence is related to the next state x_{i+1} according to the noisy one-step semantics with noise vector δ_i. When it is clear from context, we refer to just the sequence of states as the trajectory.

Bayesian optimization. Bayesian optimization (Frazier, 2018) is a well-studied global optimization algorithm for estimating the extrema of a black-box function. The most common approach uses function samples to fit a Gaussian process to the data and then performs a surrogate optimization by sampling over the Gaussian process. This is often much faster than sampling the original function. The original function is sampled at optimal points identified in the Gaussian process. The Gaussian process is then updated based on the new data (a posterior distribution is computed based on the prior distribution conditioned on the new information), and the optimization repeats.

A full review of Bayesian optimization is beyond the scope of this work (Rasmussen and Williams, 2006), but the following aspects motivated its use in our approach:

- Bayesian optimization is typically used when sampling the original function is expensive
- Each iteration requires inverting the covariance matrix in a Gaussian process; it becomes impractically slow when the number of function calls exceeds about a thousand
- Exploration and exploitation are controlled by defining an *acquisition function* that optimizes over a Gaussian process based on the predicted mean and covariance (uncertainty) at each point in the state space.

Finding minimum-noise adversarial trajectories

In this section, we define more formally the problem we are solving and describe our approach to computing minimum-noise adversarial example trajectories for neural network control systems.

https://doi.org/10.1017/cbp.2023.5 Published online by Cambridge University Press

Problem statement

For a given policy defined through g_{ACT} and environment defined through g_{ENV}, our goal is to determine a minimum-noise trajectory $\mathcal{T}$ that violates the specification. We focus on safety specifications with respect to a given set of unsafe states $U \subseteq \mathbb{R}^n$, although extensions to finite-length temporal logic properties are straightforward. A trajectory $\mathcal{T}$ is considered safe if all states $x_i \in \mathcal{T}$, $x_i \notin U$. The noise bound Δ on trajectory $\mathcal{T}$ is computed as the maximum-noise over all steps i, $\Delta = \max_i(||\delta_i||)$. Since $\mathcal{T}$ with noise Δ is the minimum-noise trajectory, any other trajectory $\mathcal{T}'$ with noise $\Delta' < \Delta$ must be safe.

In our evaluations, we represent the magnitude of sensor noise at each state as the L_∞ norm of observed variables after scaling each variable by the possible range of sensed values. However, this technique can be used with any norm that the underlying open-loop neural network verification tool supports. We consider two problems:

In **Problem 1**, we are given a single initial state $x_{init} \in \mathbb{R}^n$. We seek to find a minimal-noise trajectory with one terminus within a set of safe states and another terminus within a set of unsafe states. This problem can be considered either forward or backward in time: x_{init} can refer to a state at either the start or the end of a trajectory. If x_{init} represents the start of a trajectory within a safe region, we explore states of the environment forward in time (Mode 1) until the unsafe set is reached. Alternatively, we can seek a trajectory that ends at a given unsafe state, where x_{init} represents the end of a trajectory. Given a state transition model that can simulate backward in time Mode 2, we search over possible trajectories until we find one that would have started in a safe region. In both these cases, we can produce the provable minimum-noise trajectory, given a noise tolerance and maximum trajectory length.

In Problem 2, we consider an outer optimization over these found trajectories. We consider a set of initial states $I \subseteq \mathbb{R}^n$, where the initial state of each trajectory $x_{init} \in \mathcal{I}$. In this case, we use Bayesian optimization to sample over $\mathcal{I}$ and compute the minimum-noise trajectory as an instance of Problem 1, optimizing to find the global minimum. Although Bayesian optimization is sample-based (so we no longer have a guarantee the returned trajectory is globally the minimum), finding the global minimum is likely not possible without stronger assumptions on the environment h_{ENV}. For example, if h_{ENV} is given as a white-box symbolic model with differential equations, it may be possible to extend our approach with reachability analysis methods (Althoff et al., 2021) to find the global minimum. In this work, we assume h_{ENV} is given as a black-box function.

This definition of minimum noise and adversarial attack is a natural extension of the case of adversarial examples in perception systems. It provides information on robustness due to both naturally occurring sensor noise as well as against attackers that can alter the environment observations (for example, through bounded GPS spoofing (Nassi et al., 2021)).

Problem 1: Minimum noise from a given state

We first describe Problem 1 in depth for the forward mode, and we follow with details specific to the backward mode.

Mode 1: From an initial state, forward in time

Algorithm 1 details the procedure for determining the minimum observation noise bound and corresponding minimum-noise trajectory that violates the specification. We characterize the search by constructing a tree representing all possible trajectories that begin at a given initial state. Each node in the tree represents a possible state of the system at a given time, and the edges of the tree correspond to discrete actions.

Algorithm 1. Exact Search Algorithm

```
successor_noise_map ▷ map of (unexplored states, action) to computed noise δ, global variable

function MinErrSearch(state_start)
    root ← state_start
    exploring ← root
    while not met stopping condition do
        ComputeMinNoiseSuccessors(exploring)
        exploring, δ ← GetNextStateToExplore()
    end while

    ▷ Return the trajectory from tree root to the last explored node and corresponding δ
    return δ, 𝒯: root → exploring
end function

function ComputeMinNoiseSuccessors(state)
    a ← g_ACT(f_NN(state))                          ▷ no-noise action
    successor_noise_map[state, a] ← 0
    for a_i ∈ remaining actions do
        ▷ Get noise that would have predicted a_i
        δ = NN_VERIFY_SEARCH(state, g_ACT(f_NN), a_i)
    successor_noise_map[state, a] ← δ
    end for
end function

function GetNextStateToExplore()
    (state _next, a), δ ← arg min_δ (successor_noise_map)
    del successor_noise_map[state _next, a]
    exploring.child _edge=a ← state _next            ▷ expand tree
    exploring ← exploring.child
    exploring, δ
end function
```

Algorithm 1 describes three functions: (i) the outer-loop method that starts and terminates the search (MinErrSearch), (ii) the computation of noisy states using an open-loop neural network verification solver (ComputeMinNoiseSuccessor), and (iii) a method for determining the next node to explore (GetNextStateToExplore). We keep track of noisy states and corresponding trajectories $\mathcal{T}$ by representing each tree node as an object that stores both the original and noisy states. The initial state ($state_{start}$) is the root of the tree. For simplicity of the presentation of Algorithm 1, we omit the explicit storage and tracking of the noisy states. Nodes are explored in order of the increasing observation noise. The search concludes when a leaf node has been found such that its trajectory (the path from the root) violates the specification. In effect, we continue building out a trajectory until the noise needed to expand it further exceeds that of any other possible trajectory, and the search concludes when a path has been found to violate a safety property. The computeMinNoiseSuccessors function computes the smallest deviation to a state that would cause an incorrect target action to be predicted. This leverages open-loop neural network verification methods, which return a noisy state (the counterexample) for a fixed amount of noise. The NN_VERIFY_SEARCH function computes the smallest noise

https://doi.org/10.1017/cbp.2023.5 Published online by Cambridge University Press

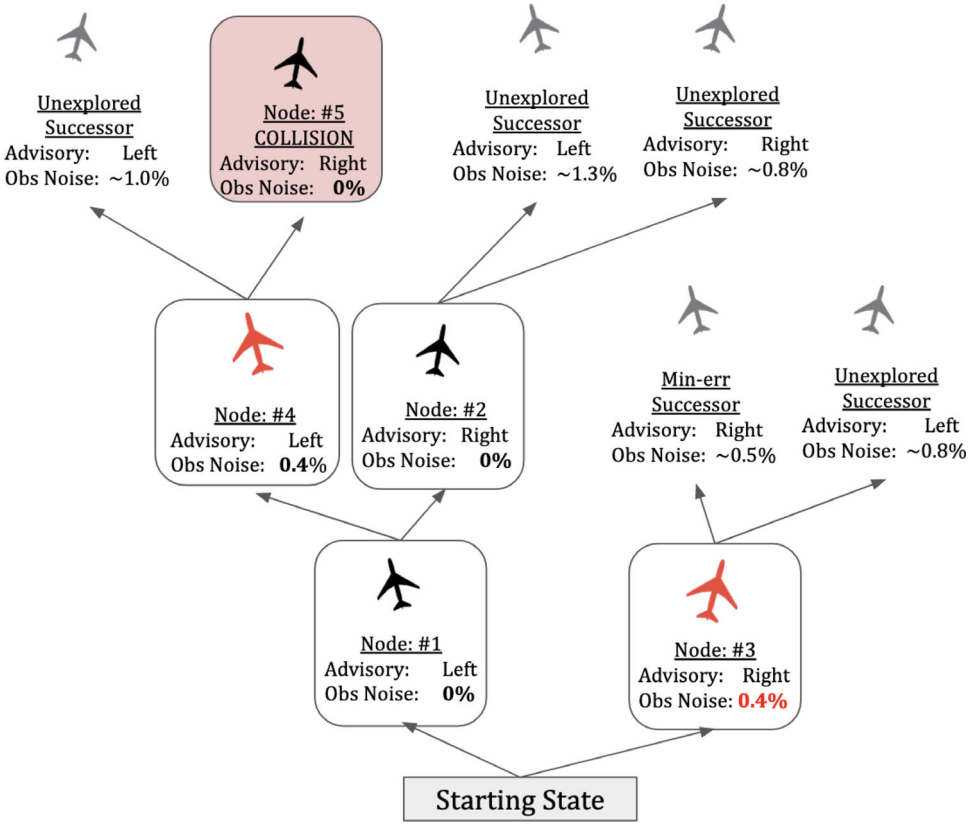

Figure 2. Illustration of a tree generated by Algorithm 1.

threshold that would cause a change in action. This is done through multiple calls to an open-loop neural network verification algorithm (Brix et al., 2023), where the exact noise threshold needed to cause a change in action is computed using a binary search. The minimum trajectory noise therefore depends on the tolerance of the binary search. Some nodes are reachable with no noise (these represent the natural behavior of the system); however, with no upper bound on added noise, all paths could be made feasible.

Algorithm 1 is guaranteed to return the trajectory associated with the least required noise, given a bound on the length of the simulation. Additionally, terminating the search at any time will yield a lower bound on the noise required to generate an unsafe trajectory; this is a certification that the system is robust to perturbations under this bound.

All the zero-noise paths are initially explored before we explore any node that requires some non-zero noise. While simulating a well-trained agent forward in time, the expansion of zero-noise paths will not terminate unless provided a maximum simulation duration (equivalently: maximum depth of the tree). Importantly, limiting the tree depth guarantees that the search will conclude and that an unsafe trajectory will be found, assuming one exists. Once all safe paths have been exhausted, the search will consider higher magnitudes of noise; given sufficient noise and a finite trajectory length, finding a path that meets the unsafe specification will always be possible.

A simplified (two control actions) illustration of the tree-building process for an aircraft collision avoidance problem is presented in Figure 2. The search is performed forward in time. The root of the tree represents a specific collision state, and the node numbers correspond to the exploration order determined by Algorithm 1. Initially, the zero-noise successors in nodes #1 and #2 are explored. As no other 0-noise successors exist, the global noise threshold is raised, and ComputeMinNoiseSuccessors identifies the smallest magnitude of sensor noise that would reveal each potential child node. Next, node #3 is explored, since the 0.4% noise needed to actualize it is the least among potential successors revealed in the previous step. The process continues until a valid start state is found, for which we used the condition that two clear-of-conflict commands would be issued in a row. If we assume that Node #5 is a safe state, then the trajectory of Nodes #5, #4, #1 and the Collision State is the minimum-noise trajectory that would terminate in the given collision. No trajectory without at least 0.4% observation noise would begin in a safe state and terminate in that collision.

Mode 2: From a final state, backward in time

The same search tree can be constructed from an initial state x_{init} that represents a given last, unsafe, state in a trajectory. In this case, we construct a tree of state predecessors backward in time, until one is found that exists in a set of safe states. The structure of Algorithm 1 remains the same; whereas we previously enumerated potential state successors, we now enumerate potential state *predecessors*. Lines 16 and 18 in the function ComputeMinNoiseSuccessors instead refer to the actions that would have caused the agent to end in the given state. For both modes, we begin with x_{init}: one state in a given set, and search for the other end of its trajectory in the opposite set.

Theorem 1. *Algorithm 1 finds the trajectory that reaches the given collision state with the minimum sensor noise.*

Proof. Let $\mathcal{T}$ be the unsafe trajectory returned by the search algorithm, given an unsafe end state $\mathcal{S}$. Let $err_{\mathcal{T}}$ refer to the noise associated with $\mathcal{T}$, or the maximum observation noise across all nodes in $\mathcal{T}$, and let $n_{\mathcal{T}}$ refer to the node at which $err_{\mathcal{T}}$ is observed. We proceed by contradiction. Assume the returned trajectory $\mathcal{T}$ is

https://doi.org/10.1017/cbp.2023.5 Published online by Cambridge University Press

Table 1. Descriptions of environments used for evaluation

	ACAS Xu	Dubins Rejoin	Cart Pole	Lunar Lander
Description	Guide ownship to avoid collision with intruder	Flight formation: guide wingman to radius around lead	Balance pole on cart by controlling cart position	Land the craft within the landing pad by controlling left and right engines
Simulation direction	Backward	Forward	Forward	Forward
Actions (Num commands)	Turn left or right (5)	Control rudder and throttle actuators (16)	Move left or right (2)	Fire left, main, or right engines (or no-op) (4)
Observations and Ranges	$\rho \in [0, 60760]$ $\theta \in [-\pi, \pi]$ $\psi \in [-\pi, \pi]$ $v_{own} = 800$ $v_{int} = 5000$	$pos_{lead-wing} \in [0, 1000]$ $pos_{rejoin-wing} \in [0, 1000]$ $v_{wingman} = 300$ $v_{lead} = 300$	$x_{cart} \in [-2.4, 2.4]$ $v_{cart} \in [-5, 5]$ $\theta_{pole} \in [-0.419, 0.419]$ $\omega_{pole} \in [-5, 5]$	$\in [-1, 1]$, $y \in [-5, 5]$ $v_x \in [-5, 5]$, $v_y \in [-5, 5]$ $\theta_{lander} \in [-\pi, \pi]$ $\omega_{lander} \in [-5, 5]$
Safety Violation	$\rho < 500$	$\|\|pos_{lead-wing}\|\|_2 < 150$	$(\|\theta_{pole}\| > 12°) \vee (\|x_{cart}\| > 2.4)$	$((y == 0) \wedge (\|x\| > 0.5)) \vee (\|x\| > 1)$

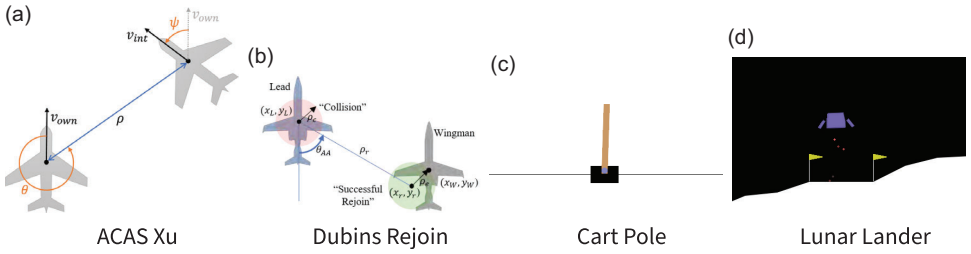

Figure 3. Environments. (a): ACAS Xu from Owen et al. (2019), (b): Dubins Rejoin from Ravaioli et al. (2022), (c): Cart Pole from Brockman et al. (2016), (d): Lunar Lander from Brockman et al. (2016).

not the minimum-noise trajectory so that there exists another trajectory $\mathcal{T}_2$ that leads to $\mathcal{S}$ with noisy $err_{\mathcal{T}_2}$, where $err_{\mathcal{T}_2} < err_{\mathcal{T}}$. Let $err_{\mathcal{T}_2}$ be associated with node $n_{\mathcal{T}_2}$. Every node corresponding to trajectory $\mathcal{T}_2$ has an observation noise less than or equal to $err_{\mathcal{T}_2}$. Since $err_{\mathcal{T}_2} < err_{\mathcal{T}}$, every node corresponding to $\mathcal{T}_2$ has an observation noise strictly less than $err_{\mathcal{T}}$. However, GetNextNodeToExplore returns the node with the least-noise child at every iteration of the search so that all nodes in $\mathcal{T}_2$ must have been explored before node $n_{\mathcal{T}}$ is explored. This then is a contradiction, because if $\mathcal{T}_2$ was an unsafe trajectory returned by GetNextNodeToExplore, then the algorithm would have terminated and MinErrSearch would have returned $\mathcal{T}_2$.

Problem 2: Minimum noise from an initial set

The MinErrSearch algorithm yields a trajectory that begins from a given initial state x_0. To determine the initial state associated with the global minimum-noise trajectory, we wrap calls to the function in an outer Bayesian optimization loop. The constructed Gaussian Process returns minimum-noise predictions in the form of a univariate normal distribution. We used the probability of expected improvement as the *acquisition function*, the normal CDF of the z-score for the minimum found value, with an added ε constant to add an exploration factor $\Phi\left(\frac{y' - \mu_k(x) - \varepsilon)}{\sigma_k(x)}\right)$ (Brochu et al., 2010). This function favors more of an exploitation strategy if ε is 0; we set ε empirically.

Evaluation

Environments

We evaluate our approach on four systems presented in Figure 3: two benchmarks from the Open AI reinforcement learning Gym and two aerospace tasks.

These environments were chosen to demonstrate the flexibility of the tool: evaluations include runs that were performed both forward and backward on systems of varied complexity and with varied input and output dimensions. The Dubins Rejoin system was originally designed for continuous output control: we trained a discretized version for use with our tool. The ACAS Xu networks were not trained via reinforcement learning; the networks were trained to mimic a lookup table (Julian et al., 2016). The OpenAI benchmarks were trained with the Stable Baselines framework (default parameters from the Stable Baselines Zoo) (Raffin et al., 2021) and the Rejoin network with the authors' implementation. More details about the set of starting states, STL safety violations, and other training information are included in Table 1. For the aircraft collision and Lunar Lander environments, we only explore noise in the position measurements (x, y, and headings); velocities are fixed throughout all simulations. The Dubins Rejoin state observations are represented in relative terms among the lead, rejoin region, and wingman.

Neural network verification

We use the neural network verification tool *nnenum* to compute the closest counterexample for a given observation and target output command. The distance from the original observation and the counterexample corresponds to the least amount of noise ρ that would cause the system to behave erroneously.

The *nnenum* tool uses linear star sets (Duggirala and Viswanathan, 2016) to compute possible outputs of a neural network given constructed sets of inputs. As inputs are propagated through a network, sets are split whenever multiple output predictions (advisories) are possible. When verification is impossible, the tool yields a counterexample (x where $x \in \mathbb{X}, f_{NN}(x) = y$ and $y \in \mathbb{U}$). Our approach relies on these counterexamples to calculate unsafe trajectories.

https://doi.org/10.1017/cbp.2023.5 Published online by Cambridge University Press

Table 2. Global minimum-noise trajectories from random and uniform samples

Environment	Min Err	Initial state	Uncertainty in units ('-' if variable fixed)
ACAS-XU	0.6%	[500, $\frac{5\pi}{6}$, $\frac{2\pi}{8}$]	[384.97 ft, 0.0398°, 0.0397°]
Dubins Rejoin	7.2 %	[500.00 ft, 0.00 ft, 26.53 ft, 160.72 ft, ... 300.00 ft/s, 0.00 ft/s, 58.53 ft/s, −294.24 ft/s]	[58.85 ft, 71.55 ft, 71.55 ft, 71.55 ft, -, -, -, -]
CartPole	1.6%	[−0.0137, 0.026, −0.0474, −0.005]	[0.16 units, 0.0896 units/s, 0.1338 rad, 0.16 rad/s]
LunarLander	0.1%	[0.45, 0.5, 0, 0, 0, 0, 0, 0]	[0.01, 0.01, -, -, 0.01, -, -, -]

The tool takes in an observation, a target output command, and a threshold for noise (box constraints) around the observation to search within. Because the exact noise threshold for each case is unknown to us in advance, we make multiple calls to *nnenum* within a binary search to pinpoint the lowest noise threshold that would yield a successful counterexample. This noise threshold corresponds to the least amount of noise ρ, and the counterexample corresponds to the adversarial observation of the noisy trajectory. We used a binary search tolerance of $1e^{-4}$ (0.01%) of the maximum range for all experiments.

In principle, any similar verification tool could be used for our approach (Bak et al., 2021). Tools that do not return counterexamples could still be used within our method to compute minimum magnitudes of tolerated noise; however, counterexamples are needed for returning the exact sequence of noisy observations.

Simulation direction and trajectory length

Our algorithm for Problem 1 was introduced for both forward and backward simulations in time. For efficiency, the direction chosen should depend on the relative sizes of the safe and unsafe states. For example, the set of possible initial states for the ACAS Xu system is much larger than the set of collision states, so we choose to perform Bayesian optimization over the unsafe collision states and explore trajectories backward in time. One advantage of conducting the search beginning with terminal unsafe states, is that a maximum simulation length is not required. With sufficient noise, the search is guaranteed to reach a safe state, yielding a trajectory that begins safely yet terminates in the supplied unsafe state. In contrast, if simulating forward from a safe state using a well-trained network, the system might never reach an unsafe state when following a 0-noise path along the tree. For this reason, we acknowledge limitations around scalability and, in practice, limit the length of the trajectories explored. This constraint guarantees that we will explore unsafe trajectories within a reasonable time, similar to limiting the length of the simulation while training the agent.

The Cart Pole and Lunar Lander evaluations were performed forward in time, and the trajectory lengths were capped at 15 timesteps. While the noise threshold returned by our approach for these systems serves as an upper bound on the true minimum, it still offers certainty that the system is provably safe within a given time horizon. Moreover, existing techniques for identifying stealthy adversarial perturbations similarly focus on a specific, vulnerable time horizon. Comparisons with these state-of-art techniques are included in the Evaluation section.

Results

Table 2 presents the least-noisy trajectory observed from uniform samples of initial states. Results are filtered to include only initial states that would have been safe without interference. Multiple minimal-noise trajectories were generated from an independent MinErrSearch (Problem 1). Each row describes the unsafe trajectory associated with the least magnitude of noise across all trials. The last two columns contextualize this noise in terms of the initial state and maximum deviation in real units.

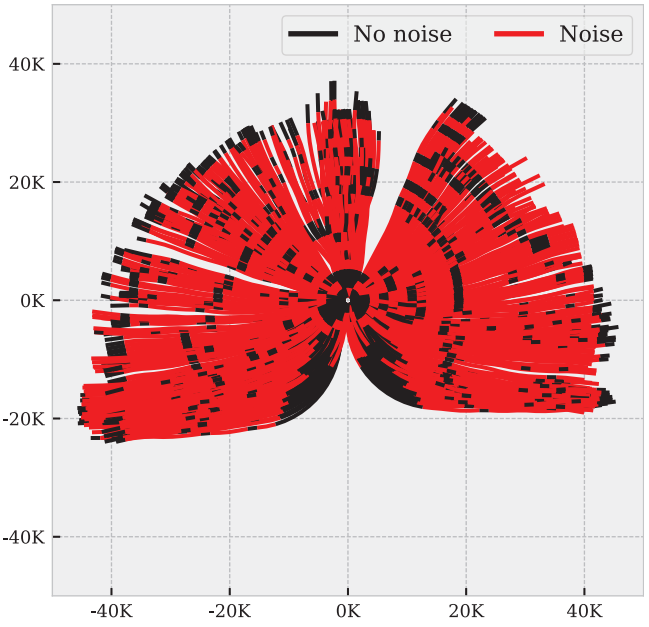

Figure 4. Minimum-noise trajectories leading up to each collision point around the intruder aircraft (minimum for each position across all headings).

ACAS Xu

Figure 4 presents unsafe trajectories found by our approach on the ACAS Xu system. Steps along each trajectory that could only be realized with sensor noise are shown in red, whereas steps without noise are shown in black. Gaps in this figure indicate regions of starting states that are safe from future collision with under 1% noise. For instance, no trajectory that terminates in a collision that we sampled begins in a tail-chasing scenario (region at the bottom half of the plot).

Interestingly, these trajectories are not symmetric around the intruder: this is consistent with observations that the response of the aircraft at single time steps is not entirely symmetric (Katz et al., 2017a).

The noise associated with each trajectory is displayed in Figure 5 along with its ending position and heading at collision. Darker cells in the heatmap correspond to smaller minimum-noise trajectories and represent more vulnerable portions of the collision

https://doi.org/10.1017/cbp.2023.5 Published online by Cambridge University Press

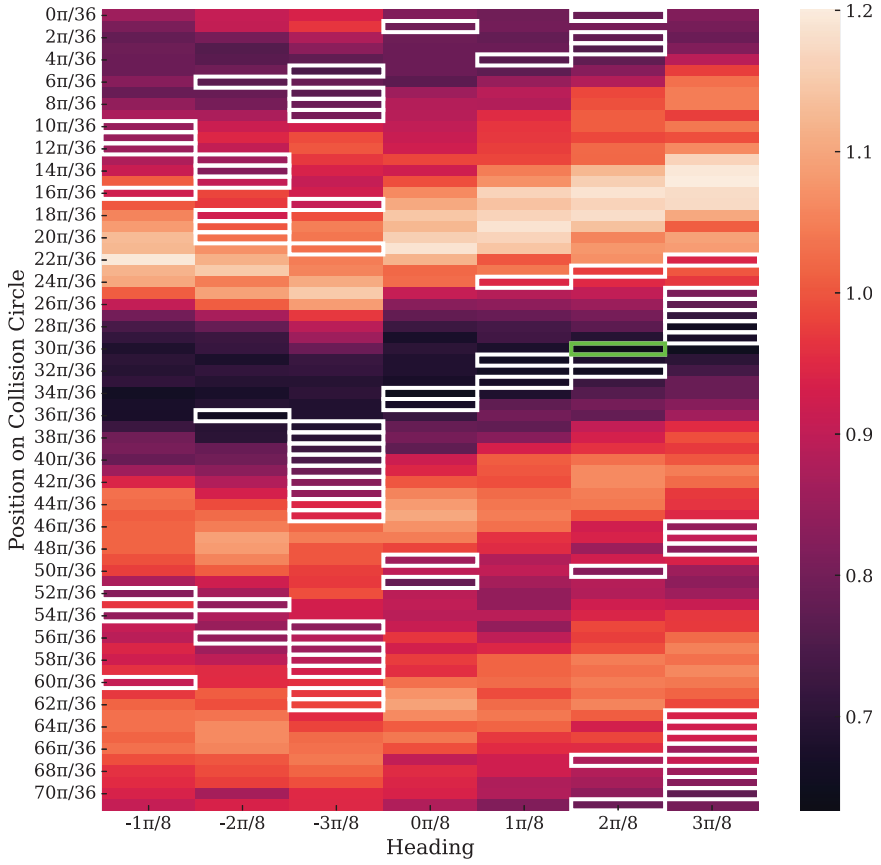

Figure 5. Noise required to end in collisions specified by position and heading. Each cell represents the least magnitude of noise required to form a trajectory that ends at the corresponding collision position/heading. The least value per row is highlighted in white; the least noise overall is in green.

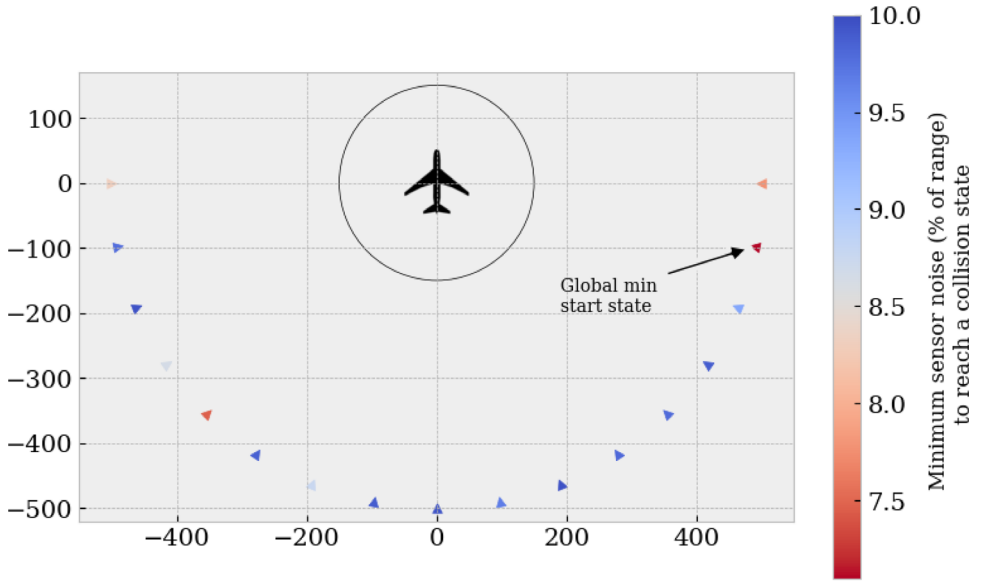

Figure 6. Least noise required to end in a collision starting from uniform initial states around the lead. The least across all runs is indicated by the arrow. The initial states plotted are within the rejoin region; the collision radius is shown by the inner circle.

surface. Cells with a white border emphasize the least-noise heading per row. The green cell indicates the smallest magnitude of noise, which quantifies the system's robustness: no smaller amount of noise will lead the aircraft into collision.

Dubins Rejoin

Results from 16 runs of the Dubins Rejoin environment are presented in Figure 6. We sampled safe starting points along the bottom half of a lead radius of 500 ft (π to 2π rad): properly trained rejoin networks should control the wingman to stay within the rejoin region and avoid flying closer to the lead. The inner circle in Figure 6 represents the collision radius of 150 ft. The color of each point represents the least amount of noise required to force the lead to cross the collision radius. The least noise observed across the start space was 7.1% of the input ranges, which translates to under 100 ft of uncertainty in the sensed positions (smaller than the length of the aircraft). The trajectories on either side of $\frac{3\pi}{2}$ rad are

https://doi.org/10.1017/cbp.2023.5 Published online by Cambridge University Press

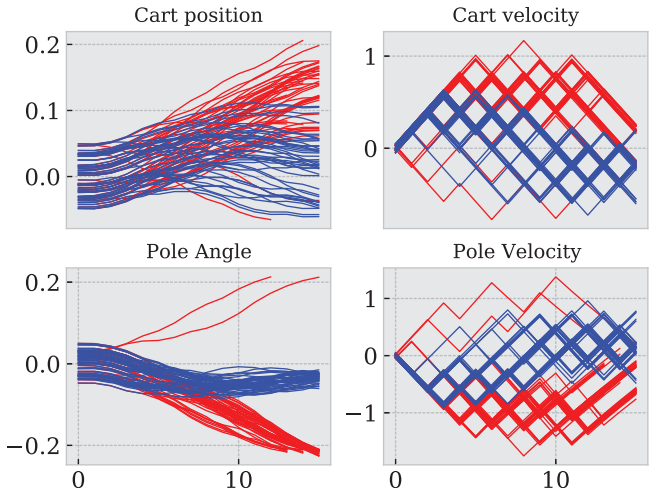

Figure 7. Cart Pole clean and adversarial states over min-noise trajectories. Blue lines represent the original (no-noise) trajectories from each initial state, and red lines represent the corresponding adversarial trajectories (The X-axis is time). The unsafe paths tend to force the cart right and pole counter-clockwise, which suggests uneven vulnerabilities in the control network. The unsafe paths additionally resulted in higher magnitude cart and pole velocities.

not completely symmetrical, similar to those from the ACAS Xu results. Again, similarly, the states closer to a tail-chasing collision are the most robust. This is likely because these states represent situations where the wingman can avoid collision without turning sharply.

Relevance to continuous-control systems: The Dubins Rejoin system can be implemented for both continuous or discrete action spaces. We investigated the extent to which findings from our approach can generalize from the discretized version to the original continuous-control version by applying the perturbed set of noisy observations to the original system. While we observed that the minimum separation distance between the two aircraft did approach the collision radius, none of the trials ended in collision. This is likely due to the size of the bins and the resulting difference in network controllers: we discretized the action space into 4 points for both the rudder and throttle actuators so that each node in the search tree had a maximum of 16 children. Smaller bins would likely result in similar controllers but with runtime tradeoffs.

Cart Pole

We evaluate on the benchmark Cart Pole problem to illustrate yet another use of our approach: understanding how clean and adversarial trajectories tend to differ. Figure 7 shows differences between clean and adversarial trajectories originating from each initial safe state; trajectories in red are those that would lead to a crash from the initial point with the least amount of sensor noise. The least-noise trajectories tended to push the cart right (higher position values in the "Cart position" subplot), causing the pole to tip counter-clockwise (lower angle values in the "Pole Angle" subplot). This illustrates a bias in the trained control network: components involved in controlling the cart right are more vulnerable to small amounts of noise than components involved in the left control. Moreover, we see that adversarial trajectories tend to have higher linear and angle velocities across all steps.

Lunar Lander

The Lunar Lander experiments were initialized from two separate regions near each landing pad pole, and trajectories were capped at a maximum depth of 30 steps from the start. Our method revealed many starting states that ended in a crash naturally; this approach provides a quantifiable metric for assessing the quality of a network. However, our approach also revealed the existence of trajectories that would have been safe without noise. These trajectories reveal vulnerabilities in a network that would have otherwise handled the case appropriately. Similar to the cartpole setting, some adversarial trajectories were also associated with higher linear and angular velocities: the landers make contact with the moon faster than in the clean settings. We reserve plots similar to Figures 5 and 7 for the appendix.

Properties of unsafe trajectories

Positions and frequency of required noisy readings

Intuitively, as an agent approaches an unsafe state, the control network should be more robust to sensor noise than it might have been farther away. However, we observed that the simulation step associated with maximum sensor noise over unsafe trajectories was not always close to the near-collision state. The ACAS Xu and Cart Pole systems did satisfy expectations: the largest noise along a trajectory tended to be closer to the collision.

However, the Dubins Rejoin and Lunar Lander distributions are less intuitive: the most critical points for noise injection were often at the start of a trajectory.

We can further analyze whether noisy observations were required across most trajectory steps or if the safety violation resulted from just a few noisy readings. While the unsafe trajectories generally require small amounts of noise ($<1\%$ of the range of each input), a large proportion of steps in each trajectory required some noise to keep the agents along an unsafe path. For ACAS Xu and Dubins Rejoin, most trajectories needed over 75% of steps to have noise for the system to reach an unsafe state. On the other hand, most trajectories from the Cart Pole and Lunar Lander environments only required noise under half the time. Nevertheless, no unsafe trajectories were realized across all experiments and environments with noise at just 1 timestep.

Overall, we find that minimum-noise unsafe trajectories are generally the result of consistent noisy readings, mostly during simulation steps closer to an unsafe state. Additional plots to support these claims are included in the appendix.

Ablation experiments: Per-variable noise

We can use our method to further investigate how the control networks are vulnerable to noise along specific input dimensions. We performed ablation experiments with the ACAS Xu system as a case study on the three inputs: ρ, θ, ψ; full details are included in Table 3.

We observe that variable θ is most sensitive – an unsafe trajectory was found with just 4.96 of noise in θ (1.3785% of its range). Fixing θ yields unsafe trajectories with much higher noise. A similar analysis could be performed on other systems to reveal vulnerabilities in certain input dimensions. These results could help prioritize how to safeguard against sensor spoofing attacks.

Effectiveness of Bayesian optimization

For environments whose initial safe states span a large range, we show that the Bayesian optimization proposed in Problem 2 adequately estimates the most vulnerable starting state. We present a case study of this optimization for the ACAS Xu system due to its large initial state space (all points on a circle at 500 ft around the intruder aircraft). Random or uniform sampling for Cart Pole and

https://doi.org/10.1017/cbp.2023.5 Published online by Cambridge University Press

Table 3. Results from a set of ablation (masking) experiments for ACAS Xu. As expected, the noise required for the least-noise unsafe trajectories decreases as more variables are allowed uncertainty. Moreover, we observe that the variable θ is most vulnerable to noise: it consistently takes a smaller magnitude of noise across all variables when θ is included. Of all experiments when just one variable is allowed uncertainty, the one where θ is noisy requires the least noise (note that all values are percentages of the allowed range for each variable, normalized to $[-1.0,\ 1.0]$)

ID	ρ	θ	ψ
A	0.6336	0.6336	0.6328
B	>10	✗	✗
C	✗	1.3785	✗
D	✗	✗	2.2520
E	1.2952	1.2886	✗
F	✗	0.8474	0.8374
G	2.1413	✗	2.1413

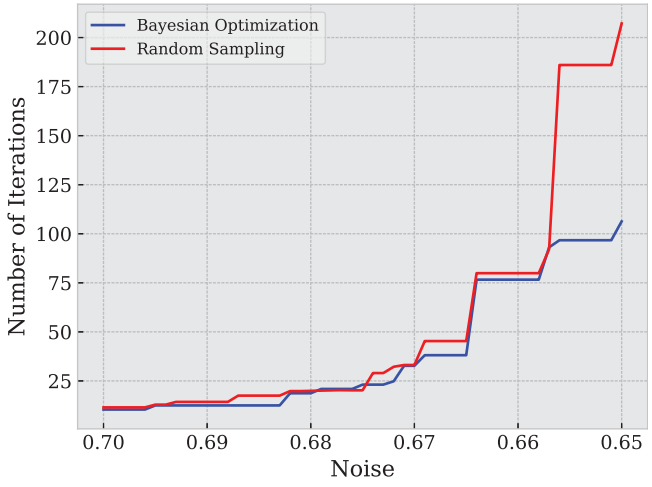

Figure 8. Comparison of Bayesian Optimization and Random Sampling for uniform grid points for ACAS Xu. MinErrSearch iterations are run until a noise limit is reached. Both techniques perform similarly until the global minimum is approached, beyond which Bayesian methods perform significantly better.

Lunar Lander is sufficient as their initial starting regions are small. The ACAS Xu system is similar to Dubins Rejoin; we focus this work on just the former to illustrate its use and avoid redundancy.

We compared the least noise found via random sampling with that found via Bayesian Optimization over the start-space domain for the same number of calls to MinErrSearch. Figure 8 shows how the number of calls needed scales when approaching the global minimum of 0.63%. Random sampling needs an average of more than 200 iterations, while Bayesian Optimization only needs ∼110 to approach the global minimum.

The exact global minimum was not found consistently with Bayesian Optimization because of the location of its corresponding state: the Matern kernel function makes assumptions of smoothness for the objective function it mimics. We confirmed that the location of the exact global minimum is located in a region with discontinuous values, making exploitation later in the search ineffective as the valley is overlooked. Instead, Bayesian optimization identified the state with the second smallest uniformly-sampled point (0.65%), which is a useful upper bound on the robustness of the system. While we demonstrated this outer-loop technique on just the ACAS Xu system, we expect similar results for other systems.

Comparisons with reinforcement learning adversarial attack methods

While our approach is the first to guarantee the smallest magnitude of noise that can cause a system to fail, we note that a parallel body of work addresses adversarial testing of agents trained via Reinforcement Learning. These works tend to focus on developing attack strategies that empirically cause the system to fail, given various limitations, by relying on training models to forecast future states and leverage probabilistic machine learning approaches that generate counterexamples at each time step.

Early works by Lin *et al.* (2017) and Sun *et al.* (2020) focused on developing attack strategies that minimize the number of perturbed time steps throughout a trajectory. Lin *et al.*'s *"Strategic Attack"* proposes using a heuristic to inform when to attack. Their timing heuristic is based on the preference of an agent taking one action over another, and the attack uses a Projected Gradient Descent method (Madry et al., 2017) to perturb the observations when the metric exceeds a tuned hyperparameter. Sun *et al.*'s *"Critical Point Attack"* (2020) expands on this by using a planning model to seek the most critical set of consecutive timesteps to attack. More recently, Zhang *et al.* (2021) developed an attack method that learns an adversarial agent that perturbs state observations to minimize the reward of its environment. They frame the adversarial agent as a Markov Decision Process (MDP) that uses the negative of the reward defined by the nominal training process to train a new neural network that approximates this adversary. The new network generates the sequence of adversarial perturbations for a supplied trajectory length and noise cap; we refer to this strategy as the *"MDP Attack."*

Although the intended uses of our approach and the mentioned attack strategies differ, we present these comparisons to highlight how our certified approach can not only identify more optimal attacks but also serve as a baseline for evaluating the optimality of such attack strategies. We focus on the three aforementioned strategies because they have publicly released their code and cover common strategies (thresholded heuristic, planning method, and training an adversarial agent). We used public-source implementations of *Strategic Attack*, *Critical Point Attack*, *MDP Attack* on GitHub.[1,2] Figures 9 and 10 present our results on the CartPole system.

First, we compare our approach against these recent methods to examine whether other methods identify the certified, smallest amount of noise found by our approach to cause a system failure. Empirical methods are not guaranteed to succeed; therefore, we report the attack success as a rate over 50 trials for which we previously found a minimal-error trajectory of 1.6%. Figure 9 presents these results for the CartPole system. While our technique is able to find the minimum-noise trajectory at 1.6% (represented by a straight line), the Strategic and Critical Point attacks only succeed when the provided magnitude of noise exceeds 10%. The MDP Attack, however, comes close: the smallest noise percentage seen to generate a successful attack is 1.65% (notably not seen at 1.6%). However, guaranteed failure (attack success at 100%) was only observed when the noise tolerance reached 2.5%.

[1]https://github.com/davide97l/rl-policies-attacks-defenses
[2]https://github.com/huanzhang12/ATLA_robust_RL

https://doi.org/10.1017/cbp.2023.5 Published online by Cambridge University Press

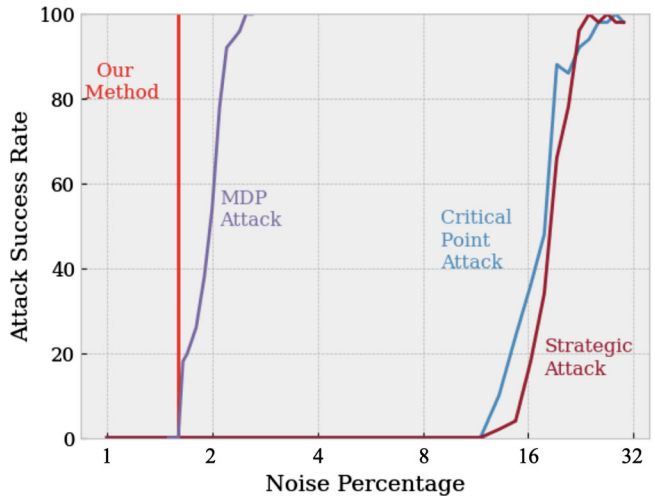

Figure 9. Attack strength v success rate, over 50 trials. The MDP (Zhang et al., 2021), Critical Point (Sun et al., 2020), and Strategic (**lin_tactics_2019**) Attacks require a set noise percentage; as that noise increases, we observe the chance of success increasing. The first percentage at which we observe a successful attack is 1.65%, 13.2%, and 14.7%, respectively, compared with 1.6% from our method for a guaranteed 100% success, noise needs to reach 2.5%, 18.5%, and 23.9% respectively. Increasing the allowed noise percentage does not change the minimum-error trajectories we obtain with our method.

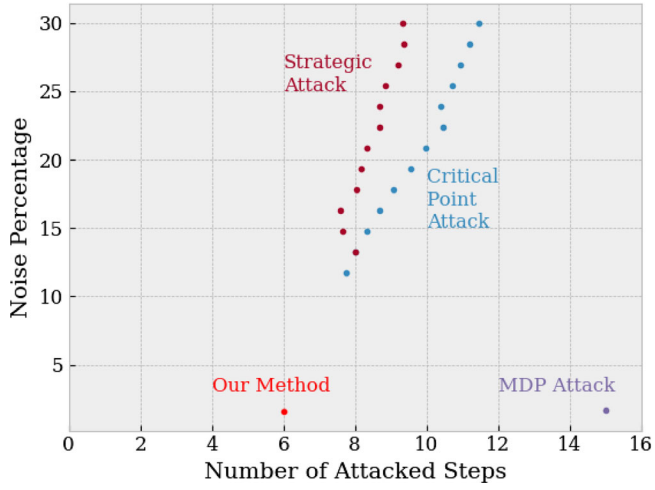

Figure 10. Comparison among the noise percentage associated with the number of attacked steps. The simpler Strategic and Critical Point strategies generally required about 8 attacked steps. The MDP Attack found successful unsafe trajectories at 1.65%, close to the 1.6% minimal noise found by our method. However, the strategy attacks all steps within the specified trajectory length, whereas we identified trajectories with as few as 6 attacked steps.

Next, as many reinforcement learning attack methods seek to reduce the number of attacked timesteps and maintain attack success, we compare the number of attacked steps and required noise percentage across attack strategies. Figure 10 presents the relationship between attacked steps and noise: intuitively, as the supplied noise cap increases, the number of successfully attacked steps could increase (in order to maximize overall attack success). The Strategic and Critical Point attacks show this relationship; they were able to generate unsafe trajectories with about 8 attacked steps. The MDP Attack, similar to our own, requires planning (and potentially attacking) all steps of a trajectory of a supplied length; we limited the Cart Pole trajectories to 15 steps for comparison with our method.

Overall, the gradient-based strategies failed to find attacks as stealthy as our method and MDP. The MDP approach empirically neared the stealthiest possible attack, which was demonstrated by a comparison with our certified approach. Going forward, we believe our method can continue to be used to evaluate the robustness of RL systems and adversarial strategies, especially as this approach can evaluate networks not trained via reinforcement learning.

Related work

Existing research on the safety of neural network controllers in cyber-physical systems generally lies in three categories: (1) on the demonstration of sensor spoofing or feasible perturbations in specific environments, (2) on anomaly detection algorithms to filter noisy observations, and (3) on identifying optimal adversarial perturbations or unsafe initial conditions for an agent. Our work is most closely related to the last area of research, with a focus on formal verification.

Neural network verification seeks to prove the properties of neural networks by relating sets of inputs to all possible sets of outputs. The formal verification problem has been shown to be NP-Complete (Katz et al., 2017b), so contemporary work looks at efficient implementations for practical scalability of verification tools to larger networks (Bak et al., 2021; Liu et al., 2019; Xiang et al., 2018).

Existing work on adversarial trajectories has focused on strategies to perturb a sequence of sensed states that empirically result in unsafe outcomes. Huang *et al.* (2017) were the first to show that neural network policies can be susceptible to adversarial attacks. The authors examined the feasibility of gradient-based attacks on policies and the transferability of examples across policies. While they showed that an adversary was able to effectively decrease accumulated rewards during an episode by attacking every frame, they did not aim to target specific actions or undesired states strategically. At the same time, Kos and Song (2017) evaluated adversarial examples against random noise and the effectiveness of re-training control policies. Lin *et al.* (2017) improved on the Huang *et al.* attack with methods for selectively modifying important frames and probabilistically guiding the agent to a target state. Moreover, Sun *et al.* (2020) looked to prediction models to determine the least number of timesteps to attack. A more recent approach by Zhang *et al.* (2021) adopts a MDP to learn the optimal adversarial attack for RL agents. This process is still empirical in nature, as it relies on the accuracy of the trained MDP model.

In contrast to these works, each trajectory that our approach generates ends in an unsafe state. Fundamentally, our approach is the first to provide certified guarantees. The data that support the findings of this study are available on request from the corresponding author.

Conclusion

Neural networks are being increasingly considered in a wide variety of systems to handle control decisions. However, the susceptibility of these networks to observation noise limits their adoption in safety-critical systems. Whereas methods like adversarial training can increase empirical robustness against such attacks, they do not offer formal guarantees.

In this paper, we proposed a method to find the minimum-noise failing trajectory for a neural network control system. Our approach is restricted to networks with discrete action spaces, and since the approach essentially constructs a tree of possible

https://doi.org/10.1017/cbp.2023.5 Published online by Cambridge University Press

trajectories with increasing noise values, attack generation can only succeed when the number of such paths is not too large. When these restrictions are met, however, our method is able to provide a strong formal guarantee that no trajectory of smaller noise is possible from the given start state. We have applied the method to four systems in order to demonstrate its practical applicability, and were able to find situations where noise under 1% of an input range can cause these systems to fail. Our results also lay the groundwork for the design of more certifiably-robust systems. For instance, minimum-error unsafe trajectories might be used to retrain networks, after which analysis could be repeated to increase provable noise robustness.

Further, the masking experiments where we explored only attacking specific sensors revealed that some inputs could tolerate more noise than others. This knowledge could be used to prioritize deployment of safeguards against spoofing attacks or to determine where to run anomaly detection systems.

Acknowledgements. This material is based upon work supported by the Air Force Office of Scientific Research and the Office of Naval Research under award numbers FA9550-19-1-0288, FA9550-21-1-0121, FA9550-23-1-0066 and N00014-22-1-2156, as well as the Air Force Research Laboratory Innovation Pipeline Fund. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the United States government, the Department of Defense, the United States Air Force or the United States Navy.

Competing interests. Multiple authors of this work have been employed by the Air Force Research Lab.

Supplementary material. To view supplementary material for this article, please visit https://doi.org/10.1017/cbp.2023.5

Connections references

Paoletti N and Woodcock J (2023) How to ensure safety of learning-enabled cyber-physical systems? *Research Directions: Cyber-Physical Systems*, 1–4. https://doi.org/10.1017/cbp.2023.2

References

Albarghouthi Aws (2021) *Introduction to neural network verification*. Available at http://verifieddeeplearning.com.verifieddeeplearning.com. arXiv: 2109.10317 [cs.LG].

Althoff M, Frehse G and Girard A (2021) Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems* **4**, 369–395.

Bak S, Liu C and Johnson T (2021) *The second international Verification of Neural Networks Competition (VNN-COMP 2021): Summary and results*. arXiv: 2109.00498, [cs] (August). Available at http://arxiv.org/abs/2109.00498 (accessed 28 October 2021).

Brix C, Müller MN, Bak S, Johnson TT and Liu C (2023) First three years of the international Verification of Neural Networks Competition (VNN-COMP). Available at https://arxiv.org/abs/2301.05815.

Brochu E, Cora VM and de Freitas N (2010) *A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning*. arXiv: 1012.2599 [cs.LG].

Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J and Zaremba W (2016) *Openai gym*. eprint: arXiv:1606.01540.

Carlini N and Wagner D (2018) Audio adversarial examples: targeted attacks on speech-to-text. In *2018* IEEE *Security and Privacy Workshops (SPW)*. IEEE, pp. 1–7.

Cheng M, Yi J, Chen P-Y, Zhang H and Hsieh C-J (2020) Seq2sick: evaluating the robustness of sequence-to-sequence models with adversarial examples. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 3601–3608.

Duggirala PS and Viswanathan M (2016) Parsimonious, simulation based verification of linear systems. In *International conference on computer aided verification*. Springer, pp. 477–494.

Eykholt K, Evtimov I, Fernandes E, Li B, Rahmati A, Xiao C, Prakash A, Kohno T and Song D (2018) Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1625–1634.

Frazier PI (2018) *A tutorial on Bayesian optimization*. Available at https://arxiv.org/abs/1807.02811.

Gong Y, Li B, Poellabauer C and Shi Y (2019) *Real-time adversarial attacks*. ArXiv: 1905.13399, [cs, eess] (June). Available at http://arxiv.org/abs/1905.13399 (accessed 26 October 2021).

Goodfellow IJ, Shlens J and Szegedy C (2014) Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

Huang S, Papernot N, Goodfellow I, Duan Y and Abbeel P (2017) Adversarial attacks on neural network policies. arXiv preprint arXiv:1702.02284.

Julian KD, Lopez J, Brush JS, Owen MP and Kochenderfer MJ (2016) Policy compression for aircraft collision avoidance systems. In *2016 IEEE/AIAA 35th digital avionics systems conference (DASC)*. https://doi.org/10.1109/DASC.2016.7778091.

Katz G, Barrett C, Dill D, Julian K and Kochenderfer M (2017a) *Reluplex: an efficient SMT solver for verifying deep neural networks*. ArXiv: 1702.01135, [cs] (May). Available at http://arxiv.org/abs/1702.01135 (accessed 30 October 2021).

Katz G, Barrett C, Dill DL, Julian K and Kochenderfer MJ (2017b) Reluplex: an efficient SMT solver for verifying deep neural networks. In *International conference on computer aided verification*. Springer.

Kos J and Song D (2017) Delving into adversarial attacks on deep policies. arXiv preprint arXiv:1705.06452.

Lin Y-C, Hong Z-W, Liao Y-H, Shih M-L, Liu M-Y and Sun M (2017) Tactics of adversarial attack on deep reinforcement learning agents. In *Proceedings of the 26th international joint conference on artificial intelligence*, 3756–3762.

Liu C, Arnon T, Lazarus C, Strong C, Barrett C and Kochenderfer MJ (2019) Algorithms for verifying deep neural networks. arXiv preprint arXiv:1903.06758.

Madry A, Makelov A, Schmidt L, Tsipras D and Vladu A (2017) Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083.

Miller WT (1989) Real-time application of neural networks for sensor-based control of robots with vision. *IEEE Transactions on Systems, Man, and Cybernetics* **19**, 825–831.

Moosavi-Dezfooli S-M, Fawzi A, Fawzi O and Frossard P (2017) Universal adversarial perturbations. ArXiv: 1610.08401, [cs, stat] (March). Available at http://arxiv.org/abs/1610.08401 (accessed 26 October 2021).

Morel N, Bauer M, El-Khoury M and Krauss J (2001) Neurobat, a predictive and adaptive heating control system using artificial neural networks. *International Journal of Solar Energy*, **21**(2–3), 161–201.

Nassi B, Bitton R, Masuoka R, Shabtai A, and Elovici Y. 2021. Sok: security and privacy in the age of commercial drones. In *2021 IEEE symposium on security and privacy (SP)*. IEEE, pp. 1434–1451.

Owen MP, Panken A, Moss R, Alvarez L and Leeper C (2019) ACAS Xu: integrated collision avoidance and detect and avoid capability for UAS. In *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*. https://doi.org/10.1109/DASC43569.2019.9081758.

Palancar MC, Aragon JM and Torrecilla JS (1998) Ph-control system based on artificial neural networks. *Industrial & Engineering Chemistry Research* **37**, 2729–2740.

Papernot N, McDaniel P and Goodfellow I (2016) *Transferability in machine learning: from phenomena to black-box attacks using adversarial samples*. arXiv: 1605.07277, [cs] (May). Available at http://arxiv.org/abs/1605.07277 (accessed 26 October 2021).

Pinto L, Davidson J, Sukthankar R and Gupta A (2017) Robust adversarial reinforcement learning. In *International conference on machine learning*. PMLR, pp. 2817–2826.

Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M and Dormann N (2021) Stable-baselines3: Reliable reinforcement learning implementations. *Journal*

of Machine Learning Research **22**, 1–8. http://jmlr.org/papers/v22/20-1364.html.

Rasmussen CE and Williams CKI (2006) *Gaussian processes for machine learning*. MIT Press.

Ravaioli U, Cunningham J, McCarroll J, Gangal V, Dunlap K and Hobbs K (2022) Safe reinforcement learning benchmark environments for aerospace control systems. In *IEEE aerospace conference*.

Shin D-H and Kim Y (2004) Reconfigurable flight control system design using adaptive neural networks. *IEEE Transactions on Control Systems Technology* **12**, 87–100.

Sun J, Zhang T, Xie X, Ma L, Zheng Y, Chen K and Liu Y (2020) Stealthy and efficient adversarial attacks against deep reinforcement learning. arXiv: 2005.07099, [cs] (May). Available at http://arxiv.org/abs/2005.07099 (accessed 28 October 2021).

Xiang W, Musau P, Wild AA, Lopez DM, Hamilton N, Yang X, Rosenfeld J and Johnson TT (2018) Verification for machine learning, autonomy, and neural networks survey. arXiv preprint arXiv:1810.01989.

Zhang H, Chen H, Boning D and Hsieh C-J (2021) Robust reinforcement learning on state observations with learned optimal adversary. arXiv preprint arXiv:2101.08452.

https://doi.org/10.1017/cbp.2023.5 Published online by Cambridge University Press