Certification of Iterative Predictions in Bayesian Neural Networks

Matthew Wicker[1] **Luca Laurenti**[1] **Andrea Patane**[1] **Nicola Paoletti**[2] **Alessandro Abate**[1] **Marta Kwiatkowska**[1]

[1]Department of Computer Science, University of Oxford, Oxford, UK
[2]Department of Computer Science , Royal Holloway University of London, London, UK

Abstract

We consider the problem of computing reach-avoid probabilities for iterative predictions made with Bayesian neural network (BNN) models. Specifically, we leverage bound propagation techniques and backward recursion to compute lower bounds for the probability that trajectories of the BNN model reach a given set of states while avoiding a set of unsafe states. We use the lower bounds in the context of control and reinforcement learning to provide safety certification for given control policies, as well as to synthesize control policies that improve the certification bounds. On a set of benchmarks, we demonstrate that our framework can be employed to certify policies over BNNs predictions for problems of more than 10 dimensions, and to effectively synthesize policies that significantly increase the lower bound on the satisfaction probability.

1 INTRODUCTION

While retaining the main advantages intrinsic to deep learning, *Bayesian neural networks* (BNNs) reason about uncertainty in a principled and probabilistic manner, making them a particularly appealing model class for tackling *safety-critical* scenarios. In principle, their predictive uncertainty can be propagated through the decision pipeline to enable formal evaluation and analysis of a system under unknown conditions [McAllister and Rasmussen, 2017], which can model partial knowledge of the system as well as its intrinsic stochasticity [Depeweg et al., 2017].

In scenarios such as sequential planning, time-series forecasting/control and model-based reinforcement learning, to evaluate a model w.r.t. a control policy (or strategy) one often needs to be able to make several predictions correlated across time [Liang, 2005]. While multiple models can be learned for each time step, a common setting is for these predictions to be made iteratively by the same machine learning model [Huang and Rosendo, 2020], where the predicted model state at each step is a function of the model configuration at the previous step and possibly an additional control input. We refer to this setting as *iterative predictions*. The challenge with BNN models is that they output probability distributions, posing the problem of successive predictions over a stochastic input. Even when the BNN posterior weights are estimated using analytical approximations, its deep and non-linear nature makes iterative predictions with BNNs an analytically intractable problem [Neal, 2012]. To the best of our knowledge, computing formal bounds on the probability of BNN-based iterative predictions remains an open problem. Such bounds would enable one to provide safety guarantees over a given (or learned) control policy, which is a necessary precondition before deploying the policy in a real-world environment [Polymenakos et al., 2020, Vinogradska et al., 2016].

In this paper, we develop a method for the computation of probabilistic guarantees for iterative predictions with BNNs over *reach-avoid* specifications. A reach-avoid specification, also known as constrained reachability [Soudjani and Abate, 2013], requires that the trajectories of a dynamical system reach a goal region over a given (finite) time horizon, whilst avoiding a given set of unsafe states. Probabilistic reach-avoid is a key property for formal analysis of stochastic processes [Abate et al., 2008], underpinning richer temporal logic specifications [Baier et al., 2008, Mnih et al., 2016, Cauchi et al., 2019]. Even though the exact computation of reach-avoid probabilities for iterative prediction with BNNs is analytically intractable, we show how to derive a guaranteed lower bound by solving a backward iterative problem obtained via a discretisation of the state space. In particular, starting from the final time step, we back-propagate lower bounds to reach-avoid probabilities through previous time steps and for each discretised portion of the state-space, beginning from the goal region. The propagation of bounds through consecutive time steps leverages bound propaga-

Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021), PMLR 161:1713–1723.

tion techniques for BNNs [Wicker et al., 2020]. By further combining these with bound propagation techniques for (non-Bayesian) neural networks (NNs) [Gowal et al., 2018, Gehr et al., 2018], we then discuss how the resulting lower bound can be employed to provide certificates for NN policies learned over the BNN dynamical system. Finally, we demonstrate how our bound can be used to tackle the synthesis problem, where given an initial policy we seek to maximise the lower bound associated to a given reach-avoid specification.

In a set of case studies, we confirm the scalability of our methodology. We begin by considering four planar control problems involving obstacle layouts of varying complexity. We then study the scalability of our framework on two locomotion problems from the Mujoco robotic physics simulator [Ray et al., 2019]. With our approach, we can derive probabilistic reach-avoid certifications for planar control tasks, including a 25-dimensional car agent. Finally, we demonstrate how controllers can be successfully improved by using our synthesis algorithm. In summary, this paper makes the following contributions:

- We show how probabilistic reach-avoid for iterative prediction with BNNs can be formulated as the solution of a backward computation problem, and design an algorithm for the lower bounding of the latter.
- We discuss how our lower-bound can be used for policy certification and, further, for synthesising NN control policies via dynamic programming.
- We demonstrate the applicability of our methodology on a set of case studies with more than 10 dimensions.

2 RELATED WORK

Certification of machine learning models is a rapidly growing area [Gehr et al., 2018, Katz et al., 2017, Gowal et al., 2018]. While most of these methods have been designed for deterministic NNs, recently safety analysis of Bayesian machine learning models has been studied both for Gaussian processes (GPs) [Grosse et al., 2017, Cardelli et al., 2019b, Blaas et al., 2020] and BNNs [Athalye et al., 2018, Cardelli et al., 2019a, Wicker et al., 2020], including methods for adversarial training [Liu et al., 2019, Wicker et al., 2021]. The above works, however, focus exclusively on the input-output behaviour of the models, that is, can only reason about static properties. Conversely, the problem we tackle in this work has additional complexity, as we aim to formally reason about iterative predictions, i.e., trajectory-level behaviour of a BNN interacting in closed-loop with a controller. Iterative predictions have been widely studied for Gaussian processes [Girard et al., 2003] and safety guarantees have been proposed in this setting in the context of model-based RL with GPs [Jackson et al., 2020, Polymenakos et al., 2019, Berkenkamp et al., 2016]. However, all these works are specific to GPs and cannot be extended to BNNs, whose posterior predictive distribution is intractable and non-Gaussian even for the more commonly employed approximate Bayesian inference methods [Neal, 2012].

Various recent works consider verification or synthesis of RL schemes against reachability specifications [Sun et al., 2019, Könighofer et al., 2020, Bacci and Parker, 2020]. None of these approaches, however, support both continuous state-action spaces and probabilistic models, as in this work. Continuous action spaces are supported in [Hasanbeig et al., 2019], where the authors provide RL schemes for the synthesis of policies maximising given temporal requirements. However, the guarantees resulting from these model-free algorithms are asymptotic, and thus of different nature than those in this work. The work of Haesaert et al. [2017] integrates Bayesian inference and formal verification over control models, additionally proposing strategy synthesis approaches for active learning [Haesaert et al., 2016, Wijesuriya and Abate, 2019]. In contrast to our paper these works do not support unknown noisy models learned via BNNs.

3 BACKGROUND

In this section we briefly review BNNs and modeling of discrete-time dynamical systems with BNNs.

Bayesian Neural Networks Let $f^w : \mathbb{R}^m \to \mathbb{R}^n$ be a feed-forward NN architecture, where $w \in \mathbb{R}^{n_w}$ is the vector containing all the weights and biases of the network. BNNs extend NNs by having a prior distribution placed over the network parameters, $p_{\mathbf{w}}(w)$, with $\mathbf{w}$ being the vector of random variables associated to the weights vector. Given a dataset $\mathcal{D}$, a BNN posterior, $p_{\mathbf{w}}(w|\mathcal{D})$, is inferred approximately by means of Bayes' rule [Neal, 2012]. Unfortunately, $p_{\mathbf{w}}(w|\mathcal{D})$ is analytically intractable. Thus, various techniques have been developed to approximate $p_{\mathbf{w}}(w|\mathcal{D})$, including Hamiltonian Monte Carlo (HMC) [Neal, 2012] and Variational Inference (VI) [Blundell et al., 2015]. While we conduct experiments on VI, the techniques we describe are general and can be employed to HMC methods, e.g., by using the approach of Wicker et al. [2021].

Iterative Predictions of BNNs Given a trained BNN, $f^{\mathbf{w}}$, we consider its associated dynamical system described by the following discrete-time stochastic control process:

$$\mathbf{x}_k = f^{\mathbf{w}}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{v}_k, \quad \mathbf{u}_k = \pi_k(\mathbf{x}_k), \quad (1)$$
$$k \in \mathbb{N}_{>0},\ \mathbf{x}_k, \mathbf{v}_k \in \mathbb{R}^n,\ \mathbf{u}_k \in \mathcal{U} \subseteq \mathbb{R}^c,$$

where $\mathbf{v}_k$ is a random variable modelling an additive noise term with stationary, zero-mean Gaussian distribution $\mathcal{N}(\bar{x}|0, \sigma^2 \cdot I)$, where I is the identity matrix. The vector $\mathbf{x}_k$ is the model state at time k; $\mathbf{u}_k$ represents the control input applied at time k, selected from an admissible, compact

set $\mathcal{U} \subset \mathbb{R}^c$ by a (deterministic) feedback Markov strategy (policy) $\pi : \mathbb{R}^n \times \mathbb{N} \to \mathcal{U}$.[1] Intuitively, the model in Eqn (1) represents a noisy controlled discrete-time stochastic process whose time evolution is given by iterative predictions of the BNN $f^{\mathbf{w}}$, and is controlled by π. In this setting, $f^{\mathbf{w}}$ defines the transition probabilities of the system and $p_{\mathbf{w}}(w|\mathcal{D})$ is employed to estimate the posterior predictive distribution $p(\bar{x}|(x,u),\mathcal{D})$ that describes the probability density of the system at the next time step being $\bar{x}$, given that the current state and action are (x,u), and it is defined as:

$$p(\bar{x}|(x,u),\mathcal{D}) = \int_{\mathbb{R}^{n_w}} \mathcal{N}(\bar{x}|f^w(x,u),\sigma^2 \cdot I) p_{\mathbf{w}}(w|\mathcal{D}) dw,$$

where $\mathcal{N}(\bar{x}|f^w(x,u),\sigma^2 \cdot I)$ is the Gaussian likelihood induced by $\mathbf{v}_k$ and centered at the NN output [Neal, 2012]. Observe that the posterior predictive distribution induces a probability density function over the state space. In iterative prediction settings this implies that at each step the state vector $\mathbf{x}_k$ fed into the BNN is a random variable. Hence, a principled propagation of the BNN uncertainty through consecutive time steps poses the problem of predictions over stochastic inputs. In Section 5 we will tackle this for the particular case of reach-avoid properties, by designing a backward computation scheme that starts its calculations from the goal region. We remark that $p(\bar{x}|(x,u),\mathcal{D})$ is defined by marginalizing over $p_{\mathbf{w}}(w|\mathcal{D})$. Hence, the particular $p(\bar{x}|(x,u),\mathcal{D})$ depends on the specific approximate inference method employed to estimate the posterior distribution. As such, the bounding results that we derive are valid only for a specifically trained BNN.

4 PROBLEM FORMULATION

For an action $u \in \mathbb{R}^c$, a subset of states $X \subseteq \mathbb{R}^m$ and a starting state $x \in \mathbb{R}^m$, we call $T(X|x,u)$ the *stochastic kernel* associated to the dynamical system. $T(X|x,u)$ describes the one-step transition probability of the model of Eqn. (1) and is defined by integrating the predictive posterior distribution with input (x,u) over X:

$$T(X|x,u) = \int_X p(\bar{x}|(x,u),\mathcal{D}) d\bar{x}.$$

Note that the integral is defined here over the state space ($\mathbb{R}^n$). In what follows, it will be convenient at times to work in the parameter space of the BNN instead. To do so, we can re-write the stochastic kernel by applying Fubini's theorem to switch the integration order, thus obtaining:

$$T(X|x,u) = \int_{\mathbb{R}^{n_w}} \left[\int_X \mathcal{N}(\bar{x}|f^w(x,u),\sigma^2 \cdot I) d\bar{x} \right] p_{\mathbf{w}}(w|\mathcal{D}) dw.$$

[1] For our settings, optimal strategies are time-dependent and Markov [Bertsekas and Shreve, 2004].

From the definition of T it follows that, under a strategy π and for a given initial condition x_0, $\mathbf{x}_k$ is a Markov process with a well defined probability measure $\Pr$ uniquely generated by the stochastic kernel T [Bertsekas and Shreve, 2004, Proposition 7.45] and such that for $X_0, X_k \subseteq \mathbb{R}^n$:

$$\Pr[\mathbf{x}_0 \in X_0] = \mathbf{1}_{X_0}(x_0),$$
$$\Pr[\mathbf{x}_k \in X_k | \mathbf{x}_{k-1} = x, \pi] = T(X_k|x, \pi_{k-1}(x)).$$

The definition of $\Pr$ allows one to make probabilistic statements over the model in Eqn (1). In Problem 1 we consider probabilistic reach-avoid, that is the probability that a trajectory of $\mathbf{x}_k$ reaches a goal region within the state space, whilst always avoiding a given set of (bad) states.

Problem 1 (Computation of Probab. Reach-Avoid) *Given a strategy π, a goal region $\mathrm{G} \subseteq \mathbb{R}^m$, a finite-time horizon $[0,N] \subseteq \mathbb{N}$, and a safe set $\mathrm{S} \subseteq \mathbb{R}^m$ such that $\mathrm{G} \cap \mathrm{S} = \emptyset$, compute for any given $x_0 \in \mathrm{G} \cup \mathrm{S}$*

$$P_{reach}(\mathrm{G},\mathrm{S},x_0,[0,N]|\pi) = Pr\big[\exists k \in [0,N], \mathbf{x}_k \in \mathrm{G} \wedge \forall 0 \leq k' < k, \mathbf{x}_{k'} \in \mathrm{S} \mid \mathbf{x}_0 = x_0, \pi\big]. \tag{2}$$

Note that, in Problem 1, the strategy π is given, and the goal is to quantify the probability with which the trajectories of $\mathbf{x}_k$ satisfy the given specification.

In Problem 2 below we generalise the previous problem and seek to synthesise a controller π that guarantees that $P_{reach}(\mathrm{G},\mathrm{S},x_0,[0.N]|\pi)$ is above a given threshold δ.

Problem 2 (Strategy Synthesis for Probab. Reach-Avoid) *For a given tolerance $0 < \delta < 1$ and $x_0 \in \mathrm{G} \cup \mathrm{S}$, find a strategy $\pi : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^c$ such that*

$$P_{reach}(\mathrm{G},\mathrm{S},x_0,[0,N] \mid \pi) > 1 - \delta. \tag{3}$$

Outline of the Approach In Section 5 we show how $P_{reach}(\mathrm{G},\mathrm{S},x,[k,N]|\pi)$ can be formulated as the solution of a backward iterative computational procedure, where the uncertainty of the BNN is propagated backward over time starting from the goal region. We will show that such a formulation of P_{reach} has two main advantages. Firstly, it allows us to define techniques for certification of BNNs to compute a sound lower bound on P_{reach}, thus guaranteeing that the process $\mathbf{x}_k$ satisfies the specification with a given probability (Section 5.1). Secondly, relying on the differentiability of the resulting lower bound, it allows one to synthesize control strategies to improve the lower bound on the reach-avoid probability.

5 PROBABILISTIC REACH-AVOID

In this section we show how $P_{reach}(\mathrm{G},\mathrm{S},x,[k,N]|\pi)$ can be formulated as the solution of a backward iterative proce-

dure, which will allow us to compute a lower bound on its value.

Given a time $0 \leq k < N$ and strategy π, consider the value functions $V_k^\pi : \mathbb{R}^n \to [0, 1]$, recursively defined as

$$V_N^\pi(x) = \mathbf{1}_{\mathrm{G}}(x),$$
$$V_k^\pi(x) = \mathbf{1}_{\mathrm{G}}(x) + \mathbf{1}_{\mathrm{S}}(x) \int V_{k+1}^\pi(\bar{x}) p\big(\bar{x} | (x, \pi_k(x)), \mathcal{D}\big) d\bar{x}. \tag{4}$$

Intuitively, V_k^π is computed backwards starting from the goal region G at $k = N$, where it is initialised at value 1. The computation then proceeds backwards for each state x by combining the current values with the transition probabilities coming from Eqn (1). The following proposition, proved in the Supplementary Material, guarantees that $V_0^\pi(x)$ is indeed a characterisation of $P_{reach}(\mathrm{G}, \mathrm{S}, x, [0, N] | \pi)$.

Proposition 1 *For $0 \leq k \leq N$ and $x_0 \in \mathrm{G} \cup \mathrm{S}$, it holds that*

$$P_{reach}(\mathrm{G}, \mathrm{S}, x_0, [k, N] | \pi) = V_k^\pi(x).$$

The backward recursion in Eqn (4) does not generally admit a solution in closed-form, as it would require integrating over the BNN posterior predictive distribution, which is in general analytically intractable. A computational scheme for lower bounding this quantity is derived in the following section.

5.1 LOWER BOUND ON P_{reach}

We develop a computational approach based on the discretisation of the state space, and on the backward formulation of Eqn (4), for calculating a lower bound for P_{reach}. As this represents a conservative estimation, the lower bound on reach-avoid can thus be used to provide probabilistic certification of a strategy controlling the BNN dynamical system.

The proposed computational approach is illustrated in Figure 1. Let $Q = \{q_1, ..., q_{n_q}\}$ be a partition of $\mathrm{S} \cup \mathrm{G}$ in n_q regions. We denote with $z : \mathbb{R}^n \to Q$ the function that associates to a state in $\mathbb{R}^n$ the corresponding partitioned state in Q. For each $0 \leq k \leq N$ we iteratively build a set of functions $K_k^\pi : Q \to [0, 1]$ such that for all $x \in \mathrm{G} \cup \mathrm{S}$ we have that $K_k^\pi(z(x)) \leq V_k^\pi(x)$. Intuitively, K_k^π provides a discretised lower bound for the value functions on the computation of P_{reach}.

The functions K_k^π are obtained by propagating backward the BNN predictions from time $k = N$, where we set $K_N^\pi(q) = \mathbf{1}_{\mathrm{G}}(q)$, with $\mathbf{1}_{\mathrm{G}}(q)$ being the indicator function (that is, 1 if $q \subseteq \mathrm{G}$ and 0 otherwise). Then, for each $k < N$, we first discretize the set of possible probabilities in n_p sub-intervals $0 = v_0 \leq v_1 \leq ... \leq v_{n_p} = 1$. Hence, for any $q \in Q$ and probability interval $[v_i, v_{i+1}]$, we compute a lower bound

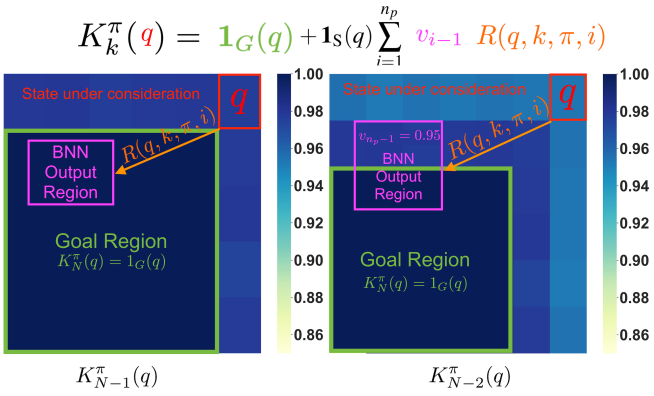

Figure 1: Examples of functions K_k^π, which are lower bounds of V_k^π for any $0 \leq k \leq N$. On the left, we consider the first step of our backward algorithm, where we compute $K_{N-1}^\pi(q)$ by computing the probability that $\mathbf{x}_N \in \mathrm{G}$ given that $\mathbf{x}_{N-1} \in q$. On the right, we consider the subsequent step. We outline the state we want to verify in red and the goal region in green. With the orange arrow we represent the 0.95 transition probability of the BNN dynamical model, and in pink we represent the worst-case probabilities spanned by the BNN output. On top, we show where each of these key terms comes into play in Eqn (6).

$R(q, k, \pi, i)$ on the probability that, starting from any state in q at time k, we reach in the next step a region that has probability $\in [v_i, v_{i+1}]$ of safely converging to the goal region. The resulting values are used to build K_k^π (as we will detail in Eqn (6)). For a given $q \subset \mathrm{S}$, $K_k^\pi(q)$ is obtained as the sum over i of $R(q, k, \pi, i)$ multiplied by v_{i-1}, i.e., the lower value that K_{k+1}^π obtains in all the states of the $i - th$ region. Note that the discretisation of the probability values does not have to be uniform, but can be adaptive for each $q \in Q$. A heuristic for picking the value of thresholds v_i will be given in Algorithm 1. In what follows, we formalise the intuition behind this computational procedure.

Lower Bounding of the Value Functions For a given strategy π, consider a constant $\eta \in (0, 1)$ and $\epsilon = \sqrt{2\sigma^2}\mathrm{erf}^{-1}(\eta)$, which are used to bound the value of the noise, $\mathbf{v}_k$, at any given time.[2] Then, for $0 \leq k < N$, consider the functions $K_k^\pi : Q \to [0, 1]$ defined recursively as follows:

$$K_N^\pi(q) = \mathbf{1}_{\mathrm{G}}(q), \tag{5}$$
$$K_k^\pi(q) = \mathbf{1}_{\mathrm{G}}(q) + \mathbf{1}_{\mathrm{S}}(q) \sum_{i=1}^{n_p} v_{i-1} R(q, k, \pi, i), \tag{6}$$

[2]The thresholds are such that it holds that $Pr(|\mathbf{v}_k^{(i)}| \leq \epsilon) = \eta$. In the experiments of Section 7 we select $\eta = 0.99$.

where

$$R(q,k,\pi,i) = \eta^n \int_{H_{k,i}^{q,\pi,\epsilon}} p_{\mathbf{w}}(w|\mathcal{D})dw, \tag{7}$$

$$H_{k,i}^{q,\pi,\epsilon} = \{w \in \mathbb{R}^{n_w} \,|\, \forall x \in q, \forall \gamma \in [-\epsilon,\epsilon]^n, \text{ it holds that:}$$
$$v_{i-1} \leq K_{k+1}^{\pi}(q') \leq v_i, \text{ with } q' = z(f^w(x,\pi_k(x)) + \gamma)\}.$$

The key component for combining the above computations together is $R(q,k,\pi,i)$, which bounds the probability that, starting from q at time k, we have that $\mathbf{x}_{k+1}$ will be in a region q' such that $K_{k+1}^{\pi}(q') \in [v_i, v_{i+1}]$. In fact, $H_{k,i}^{q,\pi,\epsilon}$ defines the weights for which that is true, so that integration of the posterior $p_{\mathbf{w}}(w|\mathcal{D})$ over the $H_{k,i}^{q,\pi,\epsilon}$ will return the probability mass for the BNN dynamical system transitioning from q to q' with probability in $[v_i, v_{i+1}]$. The computation of Eqn (6) then reduces to computing the set of weights $H_{k,i}^{q,\pi,\epsilon}$, which we call the *projecting weight set*. A method to compute a safe under-approximation $\bar{H} \subseteq H_{k,i}^{q,\pi,\epsilon}$ is discussed below. Before describing that, we analyze the correctness of the above recursion.

Theorem 1 *Given $x \in \mathbb{R}^n$, for any $k \in \{0,...,N\}$ and $q = z(x)$, assume that $H_{k,i}^{q,\pi,\epsilon} \cap H_{k,j}^{q,\pi,\epsilon} = \emptyset$ for $i \neq j$. Then:*

$$\inf_{x \in q} V_k^{\pi}(x) \geq K_k^{\pi}(q).$$

A proof of Theorem 1 is given in the Supplementary Material. Note that the assumption on the null intersection between different projecting weight sets required in Theorem 1 can always be enforced by taking their intersection and complement.

Computation of Projecting Weight Sets Theorem 1 allows us to compute a safe lower bound to Problem 1, by relying on an abstraction of the state space, that is, through the computation of $K_0^{\pi}(q)$. This can be evaluated once the projecting set of weight values $H_{k,i}^{q,\pi,\epsilon}$ associated to $[v_{i-1}, v_i]$ is known.[3] Unfortunately, direct computation of $H_{k,i}^{q,\pi,\epsilon}$ is intractable. Nevertheless, a method for its lower bounding was developed by Wicker et al. [2020] in the context of adversarial perturbations for one-step BNN predictions, and can be directly adapted to our settings.

The idea is that a safe approximation $\bar{H} \subseteq H_{k,i}^{q,\pi,\epsilon}$ is built by sampling weight boxes of the shape $\hat{H} = [w^L, w^U]$, according to the posterior, and checking whether:

$$v_{i-1} \leq K_{k+1}^{\pi}(z(f^w(x,\pi_k(x)) + \gamma)) \leq v_i,$$
$$\forall x \in q,\ \forall w \in \hat{H},\ \forall \gamma \in [-\epsilon,\epsilon]^n. \tag{8}$$

Finally, $\bar{H}$ is built as a disjoint union of boxes $\hat{H}$ satisfying the above condition. In order to apply this method to our setting, we propagate the abstract state q through the policy function $\pi_k(x)$, so as to obtain a bounding box $\widehat{\Pi} = [\pi^L, \pi^U]$ such that $\pi^L \leq \pi_k(x) \leq \pi^U$ for all $x \in q$. In the experiments we focus on the case in which $\pi_k(x)$ is given by a NN controller, so that methods for bound propagation of NNs can be used for the computation of $\widehat{\Pi}$ [Gowal et al., 2018, Gehr et al., 2018]. The results from Wicker et al. [2020] can then be used to propagate q, $\widehat{\Pi}$ and $\hat{H}$ through the BNN, that is, to compute values $f_{q,\epsilon,k}^L$ and $f_{q,\epsilon,k}^U$ such that, for all $x \in q, \gamma \in [-\epsilon,\epsilon]^n, w \in \hat{H}$, it holds that:

$$f_{q,\epsilon,k}^L \leq f^w(x,\pi_k(x)) + \gamma \leq f_{q,\epsilon,k}^U. \tag{9}$$

Furthermore, $f_{q,\epsilon,k}^L$ and $f_{q,\epsilon,k}^U$ are differentiable w.r.t. to the input vector. Finally, the two bounding values can be used to check whether or not the condition in Eqn (8) is satisfied, by simply checking whether $[f_{q,\epsilon,k}^L, f_{q,\epsilon,k}^U]$ propagated through K_{k+1}^{π} is within $[v_i, v_{i+1}]$. Now that we have the necessary ingredients, in the following we describe our algorithm for the lower bounding of P_{reach}.

Probabilistic Reach-Avoid Algorithm In Algorithm 1 we summarize our approach for computing a lower bound for Problem 1. For simplicity of presentation, we consider the case $n_p = 2$, (i.e., we partition the range of probabilities in just two intervals $[0, v_1]$, $[v_1, 1]$ - the case $n_p > 2$ follows similarly). The algorithm proceeds by first initializing the reach-avoid probability for the partitioned states q inside the goal region G to 1, as per Eqn (5). Then, for each of the N time steps and for each one of the remaining partition states q, in line 4 we set the threshold probability v_1 equal to the maximum value that K^{π} attains at the next time step over the states in the neighbourhood of q (which we capture with a hyper-parameter $\rho_x > 0$). We found this heuristic for the choice of v_1 to work well in practice (notice that the obtained bound is formal irrespective of the choice of v_1, and different choices could potentially be explored). We then proceed in the computation of Eqn (6). This computation is performed in lines 5–14. First, we initialise to the null set the current under-approximation of the projecting weight set, $\bar{H}$. We then sample n_s weights boxes $\hat{H}$ by sampling weights from the posterior, and expanding them with a margin ρ_w heuristically selected (lines 6-8). Then, for each of these sets we first propagate the state q, policy function, and weight set $\bar{H}$ to build a box $\bar{X}$ according to Eqn (9) (line 9), which is then accepted or rejected based on the value that K^{π} at the next time step attains in states in $\bar{X}$ (lines 10-12). $K_{N-i}^{\pi}(q)$ is then computed in line 14 by integrating $p_{\mathbf{w}}(w|\mathcal{D})$ over the union of the accepted sets of weights.

6 STRATEGY SYNTHESIS

We now focus on the synthesis problem. More specifically, instead of bounding the reach-avoid probability for a given strategy π, we are interested in synthesising such a strategy. In particular, we do this by finding the strategy π^*

[3] In the case of Gaussian VI the integral of Equation (7) can be computed in terms of the *erf* function, whereas more generally Monte Carlo or numerical integration techniques can be used.

Algorithm 1 Probabilistic Reach-Avoid for BNNs

Input: BNN model $f^{\mathbf{w}}$, safe region S, goal region G, discretization Q of $\mathrm{S} \cup \mathrm{G}$, time horizon N, neural controller π, number of BNN samples n_s, weight margin ρ_w, state space margin ρ_x
Output: Lower bound on K^{π}

For all $0 \leq k \leq N$ set $K_k^{\pi}(q) = 1$ iff $q \subseteq \mathrm{G}$ and 0 otherwise
for $k \leftarrow N$ to 1 **do**
  for $q \in Q \setminus \mathrm{G}$ **do**
    $v_1 \leftarrow \max_{x \in [q-\rho_x, q+\rho_x]} K_{k+1}^{\pi}(z(x))$
    $\bar{H} \leftarrow \emptyset$ {$\bar{H}$ is the set of safe weights}
    for desired number of samples, n_s **do**
      $w' \sim P(w|\mathcal{D})$
      $\hat{H} \leftarrow [w' - \rho_w, w' + \rho_w]$
      # Propagation according to (Eqn (9))
      $\bar{X} := [f_{q,\epsilon,k}^{L}, f_{q,\epsilon,k}^{U}] \leftarrow$ Prop.$(q, \pi, \hat{H}, \gamma)$
      if $K_{k+1}^{\pi}(\bar{X}) \geq v_1$ **then**
        $\bar{H} \leftarrow \bar{H} \bigcup \hat{H}$
      end if
    end for
    $K_k^{\pi}(q) = v_1 \cdot \eta^n \int_{\bar{H}} p_{\mathbf{w}}(w|\mathcal{D}) dw$ (Eqn (6))
  end for
end for

that maximises the lower bound to P_{reach} that we developed in the previous section. Notice that, while no global optimality claim can be made about the strategy that we obtain, the maximisation of a lower bound guarantees that the true reach-avoid probability will still be greater than the improved bound obtained after the maximisation.

Definition 1 *A strategy π^* is called maximal certified (max-cert), w.r.t. to the discretised value function K^{π}, if and only if, for all $x \in \mathrm{G} \cup \mathrm{S}$, it satisfies*

$$K_0^{\pi^*}(z(x)) = \sup_{\pi} K_0^{\pi}(z(x)),$$

that is, the strategy π^* maximises the lower bound of P_{reach}.

It follows that, if $K_0^{\pi^*}(z(x)) > 1 - \delta$ for all $x \in G \cup S$, then the max-cert strategy π^* is a solution of Problem 2. Note that a max-cert strategy is guaranteed to exist when the set of admissible controls $\mathcal{U}$ is compact [Bertsekas and Shreve, 2004, Lemma 3.1] (as we assume in this work). In the next theorem we show that a max-cert strategy can be computed via dynamic programming with a backward recursion similar to that of Eqn (6).

Theorem 2 *For $0 \leq k < N$ and $0 = v_0 < ... < v_{n_p} = 1$, define the functions $K_k^* : \mathbb{R}^n \to [0, 1]$ recursively as follows*

$$K_k^*(q) = \sup_{u \in \mathcal{U}} \big(\mathbf{1}_{\mathrm{G}}(q) + \mathbf{1}_{\mathrm{S}}(q) \sum_{i=1}^{n_p} v_i R(q, k, u, i) \big),$$

where $R(q, k, u, i)$ and $H_{k,i}^{q,u,\epsilon}$ are defined as in Eqn (7). If π^* is s.t. $K_0^* = K_0^{\pi^*}$, then π^* is a max-cert strategy. Furthermore, for any x, it holds that $K_0^{\pi^*}(z(x)) \leq P_{reach}(\mathrm{G}, \mathrm{S}, [0, N], x|\pi^*)$.

A proof for Theorem 2 can be derived similarly as in [Abate et al., 2008, Theorem 2]. Theorem 2 allows one to recursively compute a max-cert strategy, by selecting at each time step the action that maximizes the function K. Note that the resulting π^* will generally depend on the time step k. We remark that Theorem 2 does not make any assumption on the form of π, so that any can be employed, as long as the set in which π varies is parametrised by a compact set. In the following we focus, in particular, on the case in which the set of allowed strategies $\mathcal{U}$ is parametrised by a NN controller π, which is of particular relevance for RL applications [Arulkumaran et al., 2017]. Namely, we show how a neural controller π that builds on the lower bound can be computed using standard training methods for NNs.

Training of Certified NN Strategies In Theorem 2, the only term that depends on the input π is $\sum_{i=1}^{n_p} v_i R(q, k, \pi, i)$. Hence, in order to synthesise a strategy one needs to find the neural controller input, over $\mathcal{U}$, that maximizes the integral of $p_{\mathbf{w}}(w|\mathcal{D})$ over the projecting weight sets $H_{k,i}^{q,\pi,\epsilon}$.

Let $\mathcal{L}_{reward}$ be a (differentiable) reward function for the control problem at hand (which we obtain at training time by employing standard model-based RL algorithms [Arulkumaran et al., 2017]). Our goal is to synthesize the parameters of π_k such that $\mathcal{L}_{reward}$ is maximised, while also maximising the lower-bound to P_{reach}. In order to do so, we proceed in a similar fashion to methods for adversarial training of NNs with bound propagation techniques [Gowal et al., 2018]. Consider $P_{reach}^{LB}(\pi)$ to be the lower bound to probabilistic reach-avoid that we have developed in Section 5. Interestingly, because of differentiability of $P_{reach}^{LB}(\pi)$ the policy parameters can be optimised using standard out-of-the-box gradient descent methods for NNs. We remark that, even though Theorem 2 guarantees existence of a max-cert strategy, performing gradient descent does not guarantee to find one. However, it does provide significant local improvements of the reach-avoid probability around the starting policy π, as we show in the next section.

7 EXPERIMENTS

In this section, we empirically study the effectiveness of our framework on several benchmarks of varying complexity. In particular, we consider three different environments (Simple Navigation, Obstacle Avoidance, and Zigzag) and 5 different agents (2D Kinematic Car [Fan et al., 2018], 2D Puck [Astrom and Murray, 2008], 3D Hovercraft [Fan et al., 2020], Ball Robot, and Car Robot [Ray et al., 2019]). In each setting, we apply Algorithm 1 to certify policies learned via existing model-based strategy synthesis algorithms [Chua et al., 2018], including an experiment to study the effect of the parameter choices in Algorithm 1. We then proceed to an investigation of our synthesis methodology and an

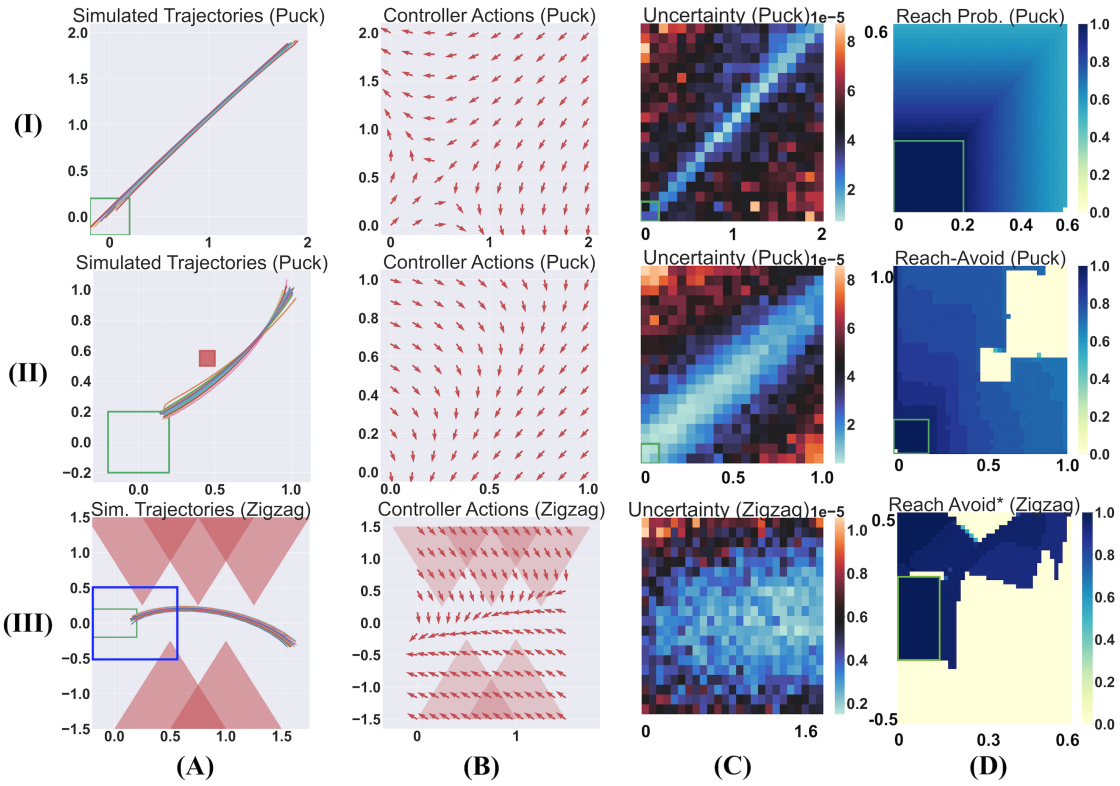

Figure 2: **Top Row (I)**: the Simple Navigation environment with a 2D Puck agent. **Middle Row (II)**: the obstacle environment with a 2D Puck agent. **Bottom Row (III)**: the Zigzag environment with 2D Kinematic Car agent. We note that the blue rectangle in column (A) corresponds to area verified in column (D). For each environment we analyse the main components of the system. **Column (A)**: a collection of 25 simulated runs using the learned policy indicates that the algorithm is successful in learning a policy to reach the goal. **Column (B)**: the per-point NN control actions show that the controller has learned a reasonable policy even outside of the explored region. **Column (C)**: Uncertainty quantification shows that where policy exploration has occurred the BNN is most certain. **Column (D)**: We are able to verify non-trivial probabilistic guarantees for each system.

evaluation of the tightness of our lower bounds against empirical probability estimates. Additional details for each of the benchmarks, environments, and agents can be found in the Supplementary Material along with a further discussion of motivation and limitations of our setting. [4]

Experimental Settings For BNN and neural policy training we utilize a standard model-based control loop. Specifically, we learn both model and policy concurrently in an episodic learning framework, whereby we operate in the environment with our policy (following the PE-TS algorithm [Chua et al., 2018]) and aggregate a dynamics dataset on which our BNN model is trained at the end of each episode. The trajectory sampling stage of PE-TS selects the action sequence which minimizes a cumulative discounted reward. We reward improvement of the agent's distance to the goal with a weighted l_p and, in the presence of obstacles, we add a penalty according to the distance to the obstacles. For all the experiments we initialize our BNN with a Gaussian prior over the parameters; approximate Bayesian inference is performed using Variational Online Gauss-Newton (VOGN) [Khan et al., 2018].

Simple Navigation The first environment we consider is a navigation task where an agent must navigate from any initial state to the origin. Albeit basic, this task becomes challenging with high dimensional agents and noisy sensors. In this scenario we have that the goal region G is a box centered at $(0.05, 0.05)$ (see Figure 2(I A)). For the Hovercraft agent the safe set is restricted to be all states with altitude within the interval (0.0, 0.5]. For the Puck, we encode a safe set that restricts the velocity of the object to be less than 1.0 at all times. For the Mujoco agents (Ball and Car Robots) we bound the change in velocity to be less than 0.25 and in these cases (where dimensionality is high) we do not discretize dimensions of the state space which are not safety-critical, e.g., the direction of the main sensor on the Car Robot. In Figure 2 row (I), we visualize the actions and simulated trajectories of the Puck agent. We note that the

[4] Link to code: *github.com/matthewwicker/BNNIterativePrediction*.

uncertainty of our BNN model is well calibrated, showing higher uncertainty in regions where less data are available. We observe that states with low uncertainty are those for which the lower bound of safely reaching the origin is higher and close to 1, even order of magnitude time steps away from the goal region (for the experiment we considered $N = 30$).

Obstacle Avoidance The obstacle avoidance task extends the simple navigation environment by adding an obstacle directly between the agent and the goal (see Figure 2(II A)). For the Hovercraft, which is not bound to the 2D plane, we assume the obstacle extends infinitely high. In Figure 2 row (II), we visualize the actions and simulated trajectories of the Puck agent. We observe in column (D) that, in this setting, the state-space portion directly behind the obstacle attains a reach-avoid probability of 0, even though sampled trajectories in Figure 2(II A) are able to safely reach the goal region. This is due to the conservatism of our approach that computes only a lower bound of P_{reach}.

Zigzag The Zigzag environment is taken from [Fan et al., 2018]. In this task, agents are placed in the fourth quadrant and are tasked with navigating through a series of equilateral triangles which impede the path to the goal region (see Figure 2(III A)). The goal region is a box centered at $(0, 0)$. In the Supplementary Material we also analyse a harder version of the Zigzag problem. In Figure 2 row (III), we visualize the actions and simulated trajectories of the Kinematic Car agent. The key observation here is that the large size of the obstacles (see column (D)) makes the verification much more challenging and Algorithm 1 produces overly conservative probabilities for a large portion of initial states. In what follows (see Table 1) we will show that, by modifying the training loss of the agent, as discussed in Section 6, one is able to obtain substantially tighter bounds.

Effect of Bound Parameters for the Car Robot Agent We analyse the effect of Algorithm 1 parameters on a *single-step* prediction on a 25D agent, the Mujoco Car, and the Simple Navigation environment. We just focus on a probabilistic version of the forward invariance property considered in Ames et al. [2014]: the policy is considered safe at a given time if the action taken does not move the agent away from the goal region at the next time step. In Figure 3, we show how increasing the number of samples from the posterior (parameter n_s in Algorithm 1), as well as increasing the size of the weight margin (parameter ρ_w in Algorithm 1), improves the resulting lower bound. Intuitively, by increasing the sample size and the weight margin we are able to build a larger under approximation of the projecting weight set. We should stress that, if the weight margin is too large, then this can be detrimental for performance due to increased approximation.

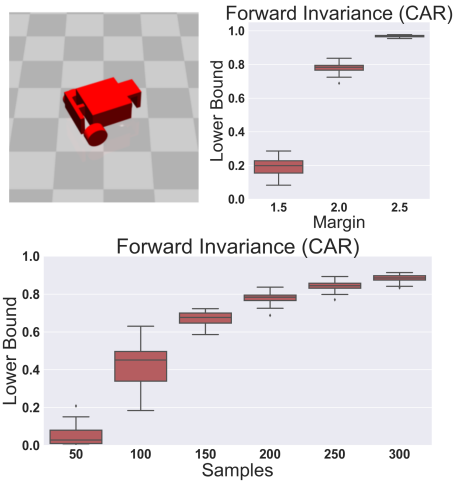

Figure 3: **Top Left**: Visualization of the car in the Mujoco simulator. **Top Right**: Increasing the weight margin has a positive effect on the bound. **Bottom**: Increasing the number of samples considered has a considerably positive effect on the bound.

Synthesis of Certified Strategies In Table 1, we compare the lower bound obtained from Algorithm 1 with an empirical estimate obtained by simulating $\mathbf{x}_k$ (Eqn 1) with a randomly picked initial state (we use 100 trajectory simulations to compute each empirical estimate). In this case we consider a subset of the verified states in Figure 2, which are close to the goal region and are in the first quadrant. For each of the tested agents (Puck, 2D Kinematic Car, and Hovercraft) our lower bound is, in the best case, within 0.22 of the empirical estimate and the tightness of the bound is greatly improved when employing Theorem 2 to synthesise strategies that maximize the lower bound given by Algorithm 1. In these examples we found that, for the Puck and Hovercraft, synthesised actions allowed us to get a certified safety within 0.03 of the statistically estimated bound. The improvement is expected because our synthesis approach aims to explicitly maximize the lower bound probability and is in line with what was observed for adversarial training of NNs with IBP [Gowal et al., 2018, Wicker et al., 2021]. Further benefits of synthesis can be observed in the Simple Navigation environment, where our approach for strategy synthesis not only improves the certification we provide, but also the empirical performance of the control policy. We further examine this in Figure 5, where we observe that with our synthesis algorithm we are able to correct for the erroneous behavior of the original PE-TS controller and certify that virtually all the states have a high probability of reaching the goal.

Env.	Agent	Emp.	Cert.	Emp. (S)	Cert. (S)
Simple	Puck	0.738	0.4444	0.986	0.9595
Zigzag	2D Car	1.00	0.7859	1.00	0.8550
Simple	Hover	1.00	0.6676	1.00	0.9706

Table 1: Lower bound obtained following Algorithm 1 compared to an empirical estimate. **Emp.** are the empirical estimates each computed over 100 trajectories simulations. **Cert.** is the average of the lower bound obtained considering only states in Q where the sampled trajectories start. **(S)** denotes bounds coming from the control actions synthesized according to our synthesis framework.

8 CONCLUSIONS

In this paper we considered iterative predictions with BNNs and studied the problem of computing the probability that a trajectory iteratively sampled from a BNN reaches safely a target goal region. We developed methods and algorithms to compute a lower bound of this reach-avoid probability and synthesize certified neural controllers, based on techniques from dynamic programming and non-convex optimization. In a set of experiments we show that our framework enables certification of strategies on BNN models and non-trivial, high-dimensional control tasks.

Acknowledgements

This project received funding from the ERC under the European Union's Horizon 2020 research and innovation programme (FUN2MODEL, grant agreement No. 834115).

References

Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008.

Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control*, pages 6271–6278. IEEE, 2014.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6): 26–38, 2017.

Karl J. Astrom and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, Princeton, NJ, USA, 2008.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, pages 274–283. PMLR, 2018.

Edoardo Bacci and David Parker. Probabilistic guarantees for safe deep reinforcement learning. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 231–248. Springer, 2020.

Christel Baier, Katoen, and Joost-Pieter. *Principles of Model Checking*. MIT Press, 2008.

Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with Gaussian processes. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 493–496, 2016.

Dimitir P Bertsekas and Steven Shreve. *Stochastic optimal control: the discrete-time case*. Athena Scientific, 2004.

Arno Blaas, Andrea Patane, Luca Laurenti, Luca Cardelli, Marta Kwiatkowska, and Stephen Roberts. Adversarial robustness guarantees for classification with Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 3372–3382. PMLR, 2020.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, Nicola Paoletti, Andrea Patane, and Matthew Wicker. Statistical guarantees for the robustness of bayesian neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5693–5700. International Joint Conferences on Artificial Intelligence Organization, 7 2019a. doi: 10.24963/ijcai.2019/789. URL `https://doi.org/10.24963/ijcai.2019/789`.

Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, and Andrea Patane. Robustness guarantees for Bayesian inference with Gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7759–7768, 2019b.

Nathalie Cauchi, Luca Laurenti, Morteza Lahijanian, Alessandro Abate, Marta Kwiatkowska, and Luca Cardelli. Efficiency through uncertainty: Scalable formal synthesis for stochastic hybrid systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 240–251, 2019.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

S Depeweg, JM Hernández-Lobato, F Doshi-Velez, and S Udluft. Learning and policy search in stochastic dynamical systems with bayesian neural networks. In *5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings*, 2017.

Chuchu Fan, Umang Mathur, Sayan Mitra, and Mahesh Viswanathan. Controller synthesis made real: reach-avoid specifications and linear dynamics. In *International Conference on Computer Aided Verification*, pages 347–366. Springer, 2018.

Chuchu Fan, Kristina Miller, and Sayan Mitra. Fast and guaranteed safe controller synthesis for nonlinear vehicle models. In *International Conference on Computer Aided Verification*, pages 629–652. Springer, 2020.

Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE S&P*, pages 3–18. IEEE, 2018.

Agathe Girard, Carl Edward Rasmussen, Joaquin Quinonero Candela, and Roderick Murray-Smith. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In *Advances in neural information processing systems*, pages 545–552, 2003.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *Neural Information Processing Systems (NeurIPS)*, 2018.

Kathrin Grosse, David Pfaff, Michael Thomas Smith, and Michael Backes. How wrong am i?-studying adversarial examples and their impact on uncertainty in gaussian process machine learning models. *arXiv preprint arXiv:1711.06598*, 2017.

S. Haesaert, P.M.J. V.d. Hof, and A. Abate. Experiment design for formal verification via stochastic optimal control. In *Proceedings of ECC*, pages 427–432, 2016.

S. Haesaert, P.M.J. V.d. Hof, and A. Abate. Data-driven and model-based verification via Bayesian identification and reachability analysis. *Automatica*, 79(5):115–126, 2017.

Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Certified reinforcement learning with logic guidance. *arXiv preprint arXiv:1902.00778*, 2019.

Jingyi Huang and Andre Rosendo. Deep vs. deep bayesian: Reinforcement learning on a multi-robot competitive experiment. *arXiv preprint arXiv:2007.10675*, 2020.

John Jackson, Luca Laurenti, Eric Frew, and Morteza Lahijanian. Safety verification of unknown dynamical systems via gaussian process regression. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 860–866. IEEE, 2020.

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

Mohammad Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable bayesian deep learning by weight-perturbation in adam. In *International Conference on Machine Learning*, pages 2611–2620. PMLR, 2018.

Bettina Könighofer, Roderick Bloem, Sebastian Junges, Nils Jansen, and Alex Serban. Safe reinforcement learning using probabilistic shields. In *International Conference on Concurrency Theory: 31st CONCUR 2020: Vienna, Austria (Virtual Conference)*. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2020.

Faming Liang. Bayesian neural networks for nonlinear time series forecasting. *Statistics and computing*, 15(1):13–29, 2005.

Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-bnn: Improved adversarial defense through robust Bayesian neural network. *7th International Conference on Learning Representations, ICLR 2019-Conference Track Proceedings*, 2019.

Rowan McAllister and Carl Edward Rasmussen. Data-efficient reinforcement learning in continuous state-action gaussian-pomdps. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Kyriakos Polymenakos, Alessandro Abate, and Stephen Roberts. Safe policy search using Gaussian process models. In *Proceedings of the 18th International Conference on Autonomous Agents and Multi Agent Systems*, pages 1565–1573. IFAAMS, 2019.

Kyriakos Polymenakos, Luca Laurenti, Andrea Patane, Jan-Peter Calliess, Luca Cardelli, Marta Kwiatkowska, Alessandro Abate, and Stephen Roberts. Safety guarantees for iterative predictions with Gaussian processes. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3187–3193. IEEE, 2020.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 2019.

S. Esmaeil Zadeh Soudjani and A. Abate. Probabilistic reach-avoid computation for partially-degenerate stochastic processes. *IEEE Transactions on Automatic Control*, 58(12):528–534, 2013.

Xiaowu Sun, Haitham Khedr, and Yasser Shoukry. Formal verification of neural network controlled autonomous systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 147–156, 2019.

Julia Vinogradska, Bastian Bischoff, Duy Nguyen-Tuong, Anne Romer, Henner Schmidt, and Jan Peters. Stability of controllers for gaussian process forward models. In *International Conference on Machine Learning*, pages 545–554. PMLR, 2016.

Matthew Wicker, Luca Laurenti, Andrea Patane, and Marta Kwiatkowska. Probabilistic safety for bayesian neural networks. In Jonas Peters and David Sontag, editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 1198–1207. PMLR, 03–06 Aug 2020.

Matthew Wicker, Luca Laurenti, Andrea Patane, Zhuotong Chen, Zheng Zhang, and Marta Kwiatkowska. Bayesian inference with certifiable adversarial robustness. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2431–2439. PMLR, 13–15 Apr 2021.

V. Wijesuriya and A. Abate. Bayes-adaptive planning for data-efficient verification of uncertain Markov decision processes. In *Proceedings of QEST, LNCS 11785*, pages 91–108, 2019.