
Certification of Iterative Predictions in Bayesian Neural Networks

Matthew Wicker*¹

Luca Laurenti*¹

Andrea Patane¹

Nicola Paoletti²

Alessandro Abate¹

Marta Kwiatkowska¹

¹Department of Computer Science, University of Oxford, Oxford, UK

²Department 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 evaluations and analyses of a system under aleatoric conditions [McAllister and Rasmussen, 2016], which can model partial knowledge of a system as well as its intrinsic stochasticity [Depeweg et al., 2016].

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 output at each step is a function of the model output 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, Vinogradskaya 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-

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 of more than 10 dimensions.

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., 2018, 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., 2020], 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.

2 BACKGROUND

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

Bayesian Neural Networks Let $f^w : \mathbb{R}^m \rightarrow \mathbb{R}^n$ 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, $p_w(w)$, with w being the vector of random variables associated to the weights vector. Given a dataset \mathcal{D} , a BNN posterior, $p_w(w|\mathcal{D})$, is inferred approximately by means of Bayes' rule [Neal, 2012]. Unfortunately, $p_w(w|\mathcal{D})$ is analytically intractable. Thus, various techniques have been developed to approximate $p_w(w|\mathcal{D})$, 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^w , we consider its associated dynamical system described by the following discrete-time stochastic control process:

$$\mathbf{x}_k = f^w(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{v}_k, \quad \mathbf{u}_k = \pi_k(\mathbf{x}_k), \quad (1)$$

$$k \in \mathbb{N}_{>0}, \mathbf{x}_k, \mathbf{v}_k \in \mathbb{R}^n, \mathbf{u}_k \in \mathcal{U} \subseteq \mathbb{R}^c,$$

where \mathbf{v}_k is a random variable modelling an additive noise term with stationary zero mean Gaussian distribution $\mathcal{N}(\bar{x}|0, \sigma^2 \cdot I)$. 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) $\pi : \mathbb{R}^n \times \mathbb{N} \rightarrow \mathcal{U}$.¹

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

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^w , and is controlled by π . In this setting, f^w defines the transition probabilities of the system and $p_w(w|\mathcal{D})$ is employed to estimate the posterior predictive $p(\bar{x}|(x, u), \mathcal{D})$ 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:

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

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 4 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 $p_w(w|\mathcal{D})$. Hence, the particular $p(\bar{x}|(x, u), \mathcal{D})$ depends on the particular approximate inference method employed to estimate the posterior distribution. As such, the bounding results that we derive are to be understood to be valid only for each specifically trained BNN.

3 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:

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

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 $X_0, X_k \subseteq \mathbb{R}^n$:

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

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 $G \subseteq \mathbb{R}^m$, a finite-time horizon $[0, N] \subseteq \mathbb{N}$, and a safe set $S \subseteq \mathbb{R}^m$ such that $G \cap S = \emptyset$, compute for any given $x_0 \in G \cup S$

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

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 $P_{reach}(G, S, x_0, [0, N]|\pi)$ is above a given threshold δ .

Problem 2 (Strategy Synthesis for Probab. Reach-Avoid)

For a given tolerance $0 < \delta < 1$ and $x_0 \in G \cup S$, find a strategy $\pi : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^c$ such that

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

Outline of the Approach In Section 4 we show how $P_{reach}(G, S, x, [k, N]|\pi)$ 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 4.1). 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.

4 PROBABILISTIC REACH-AVOID

In this section we show how $P_{reach}(G, S, x, [k, N]|\pi)$ can be formulated as the solution of a backward iterative procedure, 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 $V_k^\pi : \mathbb{R}^n \rightarrow [0, 1]$, recursively defined as

$$\begin{aligned} V_N^\pi(x) &= \mathbf{1}_G(x), \\ V_k^\pi(x) &= \mathbf{1}_G(x) + \mathbf{1}_S(x) \int V_{k+1}^\pi(\bar{x}) p(\bar{x} | (x, \pi_k(x)), \mathcal{D}) d\bar{x}. \end{aligned} \quad (4)$$

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

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

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

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 itself analytically intractable. A computational scheme for its lower bounding is derived in the following section.

4.1 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 it is a pessimistic estimation, a lower bound on reach-avoid can thus be used to provide probabilistic certification of a strategy controlling the BNN dynamical system.

Our computational approach is illustrated in Figure 1. Let $Q = \{q_1, \dots, q_{n_q}\}$ be a partition of $S \cup G$ in n_q regions. We denote with $z : \mathbb{R}^n \rightarrow Q$ 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 $K_k^\pi : Q \rightarrow [0, 1]$ such that for all $x \in G \cup S$ 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 $K_N^\pi(q) = \mathbf{1}_G(q)$, with $\mathbf{1}_G(q)$ being the indicator function (that is, 1 if $q \subseteq G$ and 0 otherwise). Then, for each $k < N$, we first discretize the set of possible probabilities in n_p sub-intervals $0 = v_0 \leq v_1 \leq \dots \leq v_{n_p} = 1$. Hence, for any $q \in Q$ and probability interval $[v_i, v_{i+1}]$, we compute a lower bound $R(q, k, \pi, i)$ 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

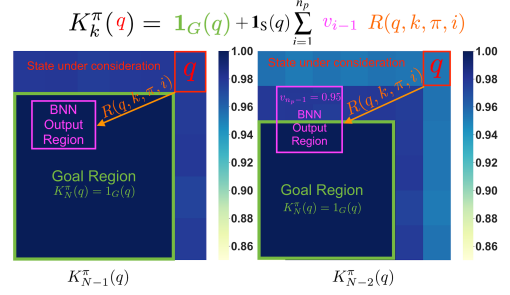


Figure 1: Examples of functions K_k^π , which are lower bounds of V_k^π for any $0 \leq k \leq N$. On the left, we consider the first step of our backward algorithm, where we compute $K_{N-1}^\pi(q)$ by computing the probability that $\mathbf{x}_N \in G$ given that $\mathbf{x}_{N-1} \in q$. On the right, 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).

region. The resulting values are used to build K_k^π (as we will detail in Eqn (6)). For a given $q \in S$, $K_k^\pi(q)$ 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 $\epsilon = \sqrt{2\sigma^2} \text{erf}^{-1}(\eta)$, 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 $K_k^\pi : Q \rightarrow [0, 1]$ defined recursively as follows:

$$K_N^\pi(q) = \mathbf{1}_G(q), \quad (5)$$

$$K_k^\pi(q) = \mathbf{1}_G(q) + \mathbf{1}_S(q) \sum_{i=1}^{n_p} v_{i-1} R(q, k, \pi, i), \quad (6)$$

where

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

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

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

²The thresholds are such that it holds that $Pr(|\mathbf{v}_k^{(i)}| \leq \epsilon) = \eta$. In the experiments of Section 6 we select $\eta = 0.99$.

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 $p_w(w|\mathcal{D})$ 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 $k \in \{0, \dots, N\}$ 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:

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

Finally, \bar{H} is built as a disjoint union of boxes \hat{H} that satisfy 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 $\hat{\Pi} = [\pi^L, \pi^U]$ 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 $\hat{\Pi}$ [Gowal et al., 2018, Gehr et al., 2018]. The results from Wicker et al. [2020] can then be used to propagate q , $\hat{\Pi}$ 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:

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

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 (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 in 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 $p_w(w|\mathcal{D})$ over the union of the accepted sets of weights.

5 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^* 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*

³In the case of Gaussian VI, in fact, the integral of Equation (7) can be computed in terms of the *erf* function, while Monte Carlo or numerical integration techniques can be used in general.

Algorithm 1 Probabilistic Reach-Avoid for BNNs

Input: BNN model f^w , safe region S , goal region G , discretization Q of $S \cup G$, 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

```

1: For all  $0 \leq k \leq N$  set  $K_k^\pi(q) = 1$  iff  $q \subseteq G$  and 0 otherwise
2: for  $k \leftarrow N$  to 1 do
3:   for  $q \in Q \setminus G$  do
4:      $v_1 \leftarrow \max_{x \in [q - \rho_x, q + \rho_x]} K_{k+1}^\pi(z(x))$ 
5:      $\hat{H} \leftarrow \emptyset$  { $\hat{H}$  is the set of safe weights}
6:     for desired number of samples,  $n_s$  do
7:        $w' \sim P(w|\mathcal{D})$ 
8:        $\hat{H} \leftarrow [w' - \rho_w, w' + \rho_w]$ 
9:       # Propagation according to (Eqn (9))
10:       $\bar{X} := [f_{q,\epsilon,k}^L, f_{q,\epsilon,k}^U] \leftarrow \text{Prop.}(q, \pi, \hat{H}, \gamma)$ 
11:      if  $K_{k+1}^\pi(\bar{X}) \geq v_1$  then
12:         $\hat{H} \leftarrow \hat{H} \cup \bar{H}$ 
13:      end if
14:    end for
15:     $K_k^\pi(q) = v_1 \cdot \eta^n \int_{\hat{H}} p_w(w|\mathcal{D}) dw$  (Eqn (6))
16:  end for
17: end for

```

if, for all $x \in G \cup S$, 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 $0 = v_0 < \dots < v_{n_p} = 1$, define the functions $K_k^* : \mathbb{R}^n \rightarrow [0, 1]$ recursively as follows

$$K_k^*(q) = \sup_{u \in \mathcal{U}} (\mathbf{1}_G(q) + \mathbf{1}_S(q) \sum_{i=1}^{n_p} v_i R(q, k, u, i)),$$

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 $K_0^{\pi^*}(z(x)) \leq P_{reach}(G, S, [0, N], x|\pi^*)$.

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 $p_w(w|\mathcal{D})$ 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 4. 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.

6 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 [Miller 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 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

⁴Link to code: github.com/matthewwicker/BNNIterativePrediction.

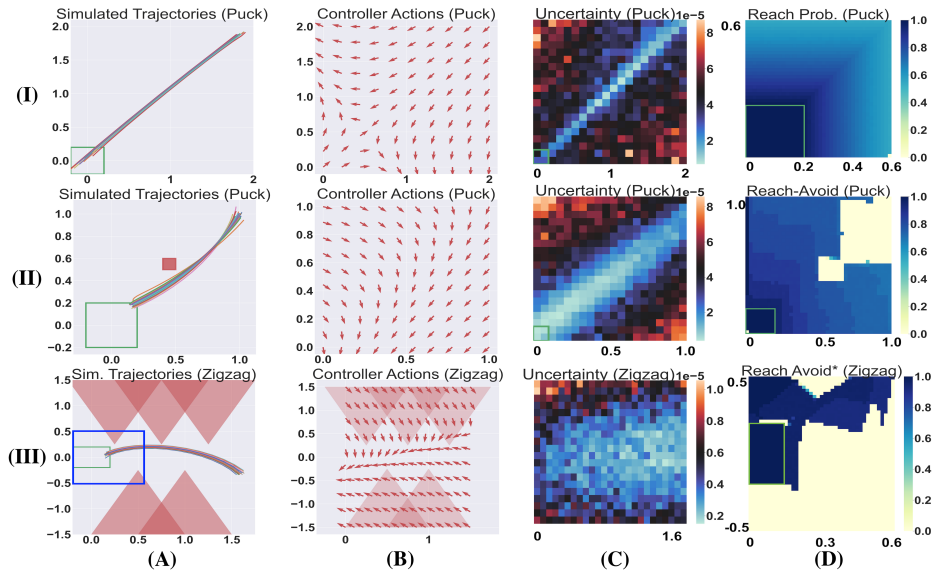


Figure 2: Top Row (I): the Simple Navigation environment with a 2D Puck agent. **Middle Row (II):** the obstacle environment with a 2D Puck agent. **Bottom Row (III):** the Zigzag environment with 2D Kinematic Car agent. We note that the blue rectangle in column (A) corresponds to area verified in column (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.

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 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 5, one is able to obtain substantially tighter bounds.

Effect of Parameter Bounding 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.

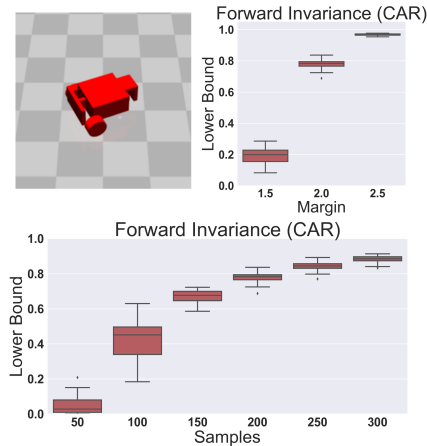


Figure 3: Top Left: Visualization of the car in the Mujoco simulator. **Top Right:** Increasing the weight margin has a positive effect on the bound. **Bottom:** Increasing the number of samples considered has a considerably positive effect on the bound.

Synthesis of Certified Strategies In Table 1, we compare the lower bound obtained from Algorithm 1 with an empirical estimate obtained by simulating 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

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.

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 8, 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.

7 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*, 2008.
- Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd ICML*, pages 1–8, 2006.
- Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *CDC*, 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 *ICML 2018*, pages 274–283. PMLR, 2018.
- C.G. Atkeson and J.C. Santamaria. A comparison of direct and model-based reinforcement learning. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 3557–3564 vol.4, 1997. doi: 10.1109/ROBOT.1997.606886.
- 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 *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 *AISTATS*, pages 3372–3382. PMLR, 2020.
- Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.
- Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.
- Ginevra Carbone, Matthew Wicker, Luca Laurenti, Andrea Patane, Luca Bortolussi, and Guido Sanguinetti. Robustness of bayesian neural networks to gradient-based attacks. *NeurIPS*, 2020.
- Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, Nicola Paoletti, Andrea Patane, and Matthew Wicker. Statistical guarantees for the robustness of Bayesian neural networks. In *IJCAI 2019*, pages 5693–5700. AAAI Press, 2019a.
- Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, and Andrea Patane. Robustness guarantees for Bayesian inference with Gaussian processes. In *AAAI*, 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 *HSCC*, 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 *NeurIPS*, pages 4754–4765, 2018.
- Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *In Proceedings of the ICML*, 2011.
- Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udfluft. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *arXiv preprint arXiv:1605.07127*, 2016.
- Chuchu Fan, Umang Mathur, Sayan Mitra, and Mahesh Viswanathan. Controller synthesis made real: reach-avoid specifications and linear dynamics. In *CAV 2018*, pages 347–366. Springer, 2018.
- Timon Gehr, Matthew Mirman, Dana Drachler-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 Quinero Candela, and Roderick Murray-Smith. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In *NeurIPS*, 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. *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, 2020.
- 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 *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 *CAV*, 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 *ICML*, pages 2611–2620. PMLR, 2018.
- Bettina Könighofer, Roderick Bloem, Sebastian Junges, Nils Jansen, and Alex Serban. Safe reinforcement learning using probabilistic shields. In *CONCUR 2020*. Schloss Dagstuhl-Leibniz-Zentrum für 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. *arXiv preprint arXiv:1810.01279*, 2018.
- Rowan McAllister and Carl Edward Rasmussen. Data-efficient reinforcement learning in continuous-state POMDPs. *arXiv preprint arXiv:1602.02523*, 2016.
- Rhiannon Michelmore, Matthew Wicker, Luca Laurenti, Luca Cardelli, Yarin Gal, and Marta Kwiatkowska. Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. In *ICRA 2020*, pages 7344–7350. IEEE, 2020.
- Kristina Miller, Chuchu Fan, and Sayan Mitra. Fast and guaranteed safe controller synthesis for nonlinear vehicle models. 2020. doi: 10.2514/6.2021-0996.
- 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 *ICML*, 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 *IFAAMS 2019*, 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 *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. Esmail 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 *HSCC 2019*, pages 147–156, 2019.
- Julia Vinogradskaya, Bastian Bischoff, Duy Nguyen-Tuong, Anne Romer, Henner Schmidt, and Jan Peters. Stability of controllers for gaussian process forward models. In *ICML 2016*, pages 545–554. PMLR, 2016.
- Matthew Wicker, Luca Laurenti, Andrea Patane, and Marta Kwiatkowska. Probabilistic safety for Bayesian neural networks. *UAI*, 2020.
- Matthew Wicker, Luca Laurenti, Andrea Patane, Zhoutong Chen, Zheng Zhang, and Marta Kwiatkowska. Bayesian inference with certifiable adversarial robustness. *AIS-TATS*, 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.

A MOTIVATION FOR BNNS IN CONTROL

Here we provide further details on the motivation to use Bayesian models in a model-based control or reinforcement learning scenario. Choosing an appropriate model when capturing unknown dynamics of a system under consideration is of critical importance to the success of the ultimate algorithm. In particular, it is known that the introduction of slight biases can greatly affect the learning of a good control policy [Atkeson and Santamaria, 1997, Abbeel et al., 2006]. Model bias can lead to over-confident predictions in the early stages of learning which can in turn lead to unsafe exploration and to the degradation of the learned control policy [Abbeel et al., 2006]. Moreover, at deployment time, being able to reason about both out-of-distribution scenarios as well as the uncertainties about one's beliefs regarding the underlying dynamics can enable more safe actions in principle [Michelmor et al., 2020]. This, incorporating a model which is inherently capable of reasoning about uncertainty and which can provide the modeller with critical feedback about model choice is intuitively desirable.

Bayesian neural networks represent a potentially powerful model for uncertainty-aware model-based reinforcement learning Chua et al. [2018]. While deterministic neural networks enable greater scalability than Bayesian neural networks, they fail to reason about uncertainty and can be a great source of model bias. Similarly, while GPs tend to be more successful in terms of calibrated uncertainty, they fail to scale to high-dimensional, large-data regime required by many real world problems [Deisenroth and Rasmussen, 2011]. Bayesian Neural Networks combine the uncertainty benefits of Gaussian processes with the scalability of neural networks. In addition, their uncertainty has been shown to make them more resistant to small changes in their inputs [Carbone et al., 2020] as well as more sample-efficient during model-based learning Chua et al. [2018].

B AGENT DESCRIPTIONS AND DYNAMICS

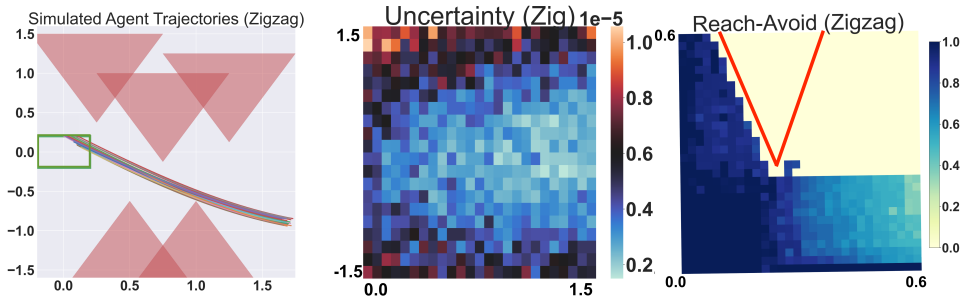


Figure 4: Analysis of the harder variant of Zigzag. **Left:** Layout of the state-space and 25 simulations from the BNN control loop demonstrates that we learn to solve the problem. **Center:** The uncertainty lines up well with what is considered in the main text and displays that the model is uncertain in states it is unable to visit. **Right:** Here we see that unlike what is presented in the main text, the controller has a bias toward navigating the agent upward and thus it is easier to verify the region below the goal.

2D Kinematic Car Both the 2D kinematic car dynamics and the Zigzag environment given in the first row of Figure 2 are benchmarks from [Fan et al., 2018]. The agent dynamics model is a planar version of a single rear wheel kinematic vehicle and has three state space variables: two for its position in the plane and one for the rotation status of the wheel. The controller chooses how to change the angle of the wheel as well as the magnitude of its movement vector. We only use the 2D kinematic car in the Zigzag environment where the agent starts below a trench created by a set of five equilateral triangle obstacles. We further consider a ‘harder’ variant of this problem with a more challenging placement of the triangles. For this environment, we use the negative l_1 distance from your current position as the reward function, meaning the agent’s action is rewarded proportional to its improvement to the goal region. In the standard instance, no information about the obstacles is given and thus must be learned via trial and error at train time. In the harder instance the reward function is modified to compute the l_2 distance of the agent to the point of each triangle and the reward is penalized according to this.

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

2D Puck The Puck environment is derived from a classical control problem of controlling a vehicle from an initial condition to a goal state or way point [Astrom and Murray, 2008]. This scenario is slightly more challenging than Zigzag not only due to the increase state-space dimension but also due to the introduction of momentum and reduced control. The state space of the agent is a four vector containing the position in the plane as well as a vector representing the current velocity. The control signal is a two vector representing a change in the velocity (i.e. an acceleration vector). We study this agent in both the ‘Simple Navigation’ and ‘Obstacle Avoidance’ scenarios as visualized in Figure 2 rows (I) and (II), respectively. For the former, an l_2 reward signal is used, and for the latter an l_2 reward penalized by the l_2 distance to the obstacle is used.

In Table 1 we report that our synthesis algorithm not only improves the certification but also the empirical performance of the controller. In Figure 8, we give a visual explanation for how this is the case. The original controller learned by the PE-TS algorithm did not, under certain conditions, apply enough velocity in the negative x direction thus not only making certification difficult, but also adversely affecting the performance. We see that our synthesis algorithm corrects for this and enables us to not only improve performance, as reported in Table 1, but also improve the certification.

The dynamics of the puck can be given as a the following system of equations where η determines friction, m determines the mass of the puck, and h determines the size of the time discretization.

$$\dot{q} = Aq + Bc$$

$$A = \begin{bmatrix} 1 & 0 & h & 0 \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 - h\eta/m & 0 \\ 0 & 0 & 0 & 1 - h\eta/m \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ h/m & 0 \\ 0 & h/m \end{bmatrix}$$

3D Hovercraft The dynamics of the 3D hovercraft are similar to those of the 2D kinematic car, save for the fact that it exists in 3D and must also learn a policy which maintains altitude. This vehicle is taken from [Miller et al., 2020] and amended with a gravity term that makes learning slightly harder, but more realistic. We consider the task of learning to navigate to the goal in the presence of an infinitely tall triangular obstacle placed directly in the path of the agent.

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & 0 \\ \sin(\theta) & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

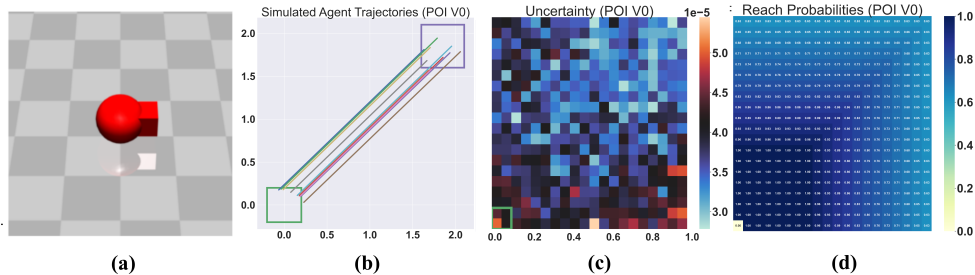


Figure 5: Far Left: A render of the Ball robot in the Mujoco simulator. Center Left: 25 simulations of the Ball robot control loop. Center Right: The uncertainty of the BNN for the Ball robot does not show an interpretable pattern indicating that it may not be very well calibrated. Right: Despite the higher dimensionality of the Ball robot problem we are still able to compute good lower-bounds for a portion of the state-space.

Ball Robot For the 3D Ball Robot environment we consider a simple locomotive task inside of the OpenAI gym RL suite [Brockman et al., 2016]. The agent observes a set of noisy sensor outputs involving the center of mass of the robot, the

status of its wheel, velocity, and the location of its sensor (red cube seen in [TODO]). The challenge of this task is not only the dimensional of the robots state-space but also noise injected into the observations. Due to this, the BNN dynamics model takes longer to converge and it makes it challenging for the PE-TS algorithm to quickly identify a strong strategy. Despite this, after 20 episodes of learning, the BNN fits the dynamics of the system well and is able to reliably navigate to the goal. The dynamics of this agent are wholly defined by the Mujoco physics simulator and thus are too complex to be listed in closed form here.

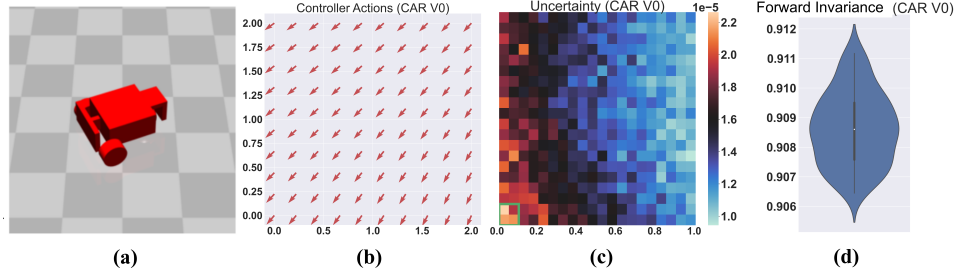


Figure 6: Far Left: A render of the Car robot in the Mujoco simulator. Center Left: A visualization of the control policy learned by the car robot. Center Right: The uncertainty of the BNN for the Car robot does not show an interpretable pattern indicating that it may not be very well calibrated. Right: Similarly to what is shown in Figure 3 we are able to obtain high probability certificates for the one-step forward invariance property.

Car Robot The Car Robot poses a locomotion problem in which the controller must navigate a vehicle with two independently-driven parallel wheels and a free rolling rear wheel into a goal region. Just as with the ball robot, the Car is not fixed to the 2D-plane, and observations are taken from noisy sensors about the state of the robot in space. This problem is significantly more challenging than the previous robots as moving forward and turning require coordination of the independently actuated wheels. Due to the large dimensionality of the state space of this robot, reach properties cannot be easily handled due to the explosion of the state space discretization needed. Instead, we compute a probabilistic lower bound on the one step forward invariance property presented in [Ames et al., 2014]. Similar to the Ball robot, the dynamics of the physics simulator are too complex to be listed in closed form here.

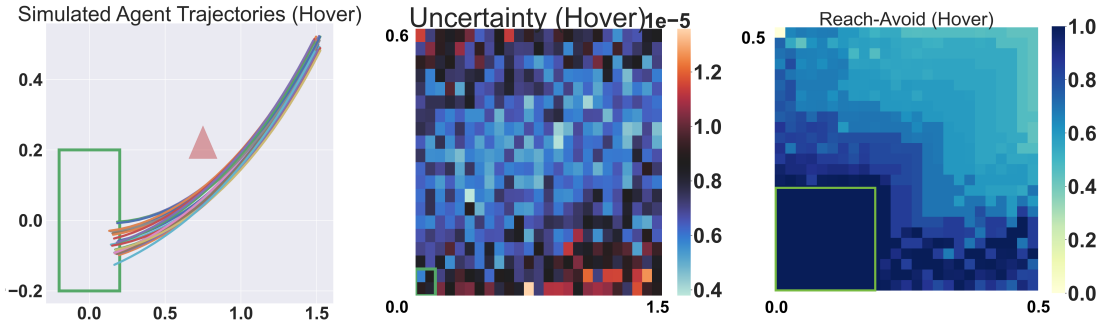


Figure 7: Left: Simulated trajectories from the hovercraft control loop demonstrates that it successfully solves the task well and navigates around the obstacle. Center: The uncertainty of this BNN is interpretable but may not be considered well-calibrated. Right: The reach-avoid property shows that we are able to verify a large portion of the state-space, here we are considering negative altitude as the avoid region of the state-space.

C LEARNING PARAMETERS

In this section, we use Table 2 to report the parameters of our learning set up. We report the state-space and control dimensions (n and m respectively) as well as the architectures used for both the dynamics model (BNN) and controller (DNN). We give the number of episodes of PE-TS that are used in these scenarios as well as the number of samples and time horizon considered during the trajectory sampling stage of PE-TS. During the trajectory sampling stage we have a set of 3-10 different discrete control signals which can be picked for each control dimension (see dynamics above). In order to pick the best action for the current time step, one simple randomly samples a series of actions over a finite time horizon and

uses the sum of discounted rewards in order to pick the most promising action sequence. From this, only the first action is taken and then the process is repeated. For more details see Chua et al. [2018] and for the exact sampling parameters for our settings we reference Table 2.

All BNN posteriors are approximately inferred with the VOGN algorithm Khan et al. [2018].

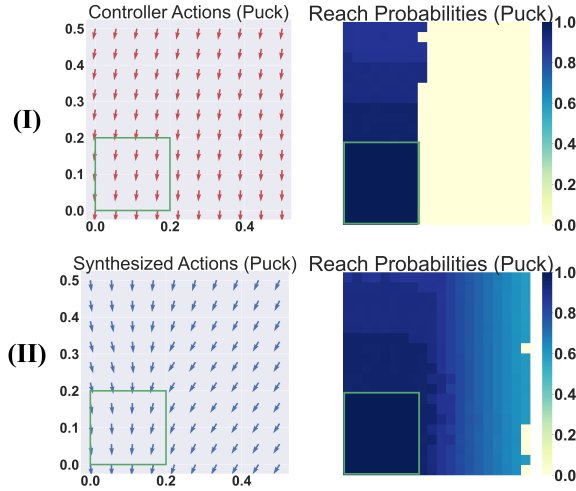


Figure 8: Left Column: Actions given by DNN controller trained with PE-TS (I) do not properly account for movement in the negative x direction compared to synthesized actions (II). **Right Column:** Verification of learned controller (I) has worse certified bounds compared to that of synthesized control actions (II).

Benchmark:	n, m	BNN Arch.	Control Arch.	Property	Episodes	Action Samples	Time Horizon
Zigzag	3,2	FCN-1-64	FCN-1-32	Reach-Avoid	8	4096	5
Zigzag-Hard	3,2	FCN-1-64	FCN-1-64	Reach-Avoid	12	8000	3
PointMass (Simple)	4,2	FCN-1-64	FCN-1-64	Reach	10	2048	9
PointMass (Obstacle)	4,2	FCN-1-64	FCN-1-32	Reach-Avoid	10	7500	10
Hovercraft	5,3	FCN 1-64	FCN 1-32	Reach-Avoid	15	10000	5
Ball Robot	8,2	FCN-1-128	FCN-1-64	Reach	20	10000	10
Car Robot	23,2	FCN-1-128	FCN-1-64	Forward Invariance	25	5000	5

Table 2: Here we report the parameters of our learning environments. Episodes refers to the number of learning episodes performed before the policy was successful at solving the task. Action Samples and Time Horizon refer to the trajectory sampling required by PE-TS.

D LIMITATIONS OF THE APPROACH

The limitations of our approach come from a few places: approximation of Bayesian inference, approximation in certification, and the reliance on discretization of the state space.

Approximate Inference In both theory and practice, an overly approximate inference method can hamper our framework. While it makes perfect sense to verify with respect to ones posterior beliefs about an underlying system’s unknown dynamics, if those posterior beliefs are flawed (e.g. due to overly approximate inference) then the probabilities of safety may not match the true probability of safety due to a gap between posterior belief and reality. We do note, however, that recent works in approximate inference have shown that with a larger computational budget that BNNs inferred (even with MCMC) can reliably scale to inputs with tens of thousands of dimensions.

Reliance on Discretization While modern certification and bound propagation techniques for NNs and BNNs have shown remarkable scalability, being able to scale to images with thousands of input dimensions, we remark that formal reach-avoid certification of an input space can only be done by fully considering the potential interdependence of different dimensions of the state space. As such, while single-step certification may be able to scale to large inputs, our method incurs a complexity

exponential in the number of state space variables that one discretizes over. Finally we note that, as in our experiments, one can choose to only discretize safety critical variables while also providing certification for single-step constraints over other variables.

E PROOFS

Proof of Proposition 1 In what follows, we omit π (which is given and held constant) from the probabilities for a more compact notation. The proof is by induction. The base case is $k = N$, for which we have

$$V_N^\pi(x) = \mathbf{1}_G(x) = P_{reach}(G, S, x, [N, N]),$$

which holds trivially. Under the assumption that, for any given $k \in [0, N - 1]$, it holds that

$$V_{k+1}^\pi(x) = P_{reach}(G, S, x, [k + 1, N]), \quad (10)$$

we show the induction step for time step k . In particular,

$$\begin{aligned} P_{reach}(G, S, x, [k, N] | \pi) &= \\ Pr(\mathbf{x}_k \in G | \mathbf{x}_k = x) &+ \sum_{j=k+1}^N Pr(\mathbf{x}_j \in G \wedge \forall j' \in [k, j], \mathbf{x}_{j'} \in S | \mathbf{x}_k = x) = \\ \mathbf{1}_G(x) + \mathbf{1}_S(x) &\sum_{j=k+1}^N Pr(\mathbf{x}_j \in G \wedge \forall j' \in [k, j], \mathbf{x}_{j'} \in S | \mathbf{x}_k = x) \end{aligned}$$

Now in order to conclude the proof we want to show that

$$\begin{aligned} \sum_{j=k+1}^N Pr(\mathbf{x}_j \in G \wedge \forall j' \in [k, j] + 1, \mathbf{x}_{j'} \in S | \mathbf{x}_k = x) = \\ \int V_{k+1}^\pi(\bar{x}) p(\bar{x} | (x, \pi_k(x)), \mathcal{D}) d\bar{x}. \end{aligned}$$

This can be done as follow

$$\begin{aligned}
& \sum_{j=k+1}^N Pr(\mathbf{x}_j \in \mathbf{G} \wedge \forall j' \in [k+1, j), \mathbf{x}_{j'} \in \mathbf{S} | \mathbf{x}_k = x) = \\
& Pr(\mathbf{x}_{k+1} \in \mathbf{G} | \mathbf{x}_k = x) + \\
& \sum_{j=k+2}^N Pr(\mathbf{x}_j \in \mathbf{G} \wedge \forall j' \in [k+1, j), \mathbf{x}_{j'} \in \mathbf{S} | \mathbf{x}_k = x) = \\
& \int_{\mathbf{G}} p(\bar{x} | (x, \pi_k(x)), \mathcal{D}) d\bar{x} + \\
& \sum_{j=k+2}^N \int_{\mathbf{S}} Pr(\mathbf{x}_j \in \mathbf{G} \wedge \forall j' \in [k+2, j), \mathbf{x}_{j'} \in \mathbf{S} \wedge \mathbf{x}_{k+1} = \bar{x} | \mathbf{x}_k = x) d\bar{x} = \\
& \int_{\mathbf{G}} p(\bar{x} | (x, \pi_k(x)), \mathcal{D}) d\bar{x} + \\
& \sum_{j=k+2}^N \int_{\mathbf{S}} Pr(\mathbf{x}_j \in \mathbf{G} \wedge \forall j' \in [k+2, j), \mathbf{x}_{j'} \in \mathbf{S} | \mathbf{x}_{k+1} = \bar{x}) p(\bar{x} | (x, \pi_k(x)), \mathcal{D}) d\bar{x} = \\
& \int (\mathbf{1}_{\mathbf{G}}(\bar{x}) + \\
& \mathbf{1}_{\mathbf{S}}(\bar{x}) \sum_{j=k+2}^N Pr(\mathbf{x}_j \in \mathbf{G} \wedge \forall j' \in [k+2, j), \mathbf{x}_{j'} \in \mathbf{S} | \mathbf{x}_{k+1} = \bar{x})) p(\bar{x} | (x, \pi_k(x)), \mathcal{D}) d\bar{x} = \\
& \int V_{k+1}^{\pi}(\bar{x}) p(\bar{x} | (x, \pi_k(x)), \mathcal{D}) d\bar{x}
\end{aligned}$$

where the third step holds by application of Bayes rule over multiple events.

Proof of Theorem 1 The proof is by induction. The base case is $k = N$, for which we have

$$\inf_{x \in q} V_N^{\pi}(x) = \inf_{x \in q} \mathbf{1}_{\mathbf{G}}(x) = \mathbf{1}_{\mathbf{G}}(q) = K_N^{\pi}(q).$$

Next, under the assumption that for any $k \in \{0, N-1\}$ it holds that

$$\inf_{x \in q} V_{k+1}^{\pi}(x) \geq K_{k+1}^{\pi}(q),$$

we can work on the induction step: in order to derive it, it is enough to show that for any $\epsilon > 0$

$$\begin{aligned}
& \int V_{k+1}^{\pi}(\bar{x}) p(\bar{x} | (x, \pi_k(x)), \mathcal{D}) d\bar{x} \geq \\
& F([- \epsilon, \epsilon] | \sigma^2)^n \sum_{i=1}^{n_p} \int_{H_{k,i}^{q,\pi}} v_{i-1} p_{\mathbf{w}}(w | \mathcal{D}) dw,
\end{aligned}$$

where $F([- \epsilon, \epsilon] | \sigma^2) = \text{erf}(\frac{\epsilon}{\sqrt{2\sigma^2}})$ is the cumulative function distribution for a normal random variable with zero mean and variance σ^2 being within $[- \epsilon, \epsilon]$. This can be argued by rewriting the first term in parameter space (recall that the stochastic

kernel T is induced by $p_{\mathbf{w}}(w|\mathcal{D})$) and providing a lower bound, as follows:

$$\int V_{k+1}^{\pi}(\bar{x})p(\bar{x} | (x, \pi_k(x)), \mathcal{D})d\bar{x} =$$

(By definition of predictive distribution)

$$\int \left(\int V_{k+1}^{\pi}(\bar{x})p(\bar{x}|(x, u), w)d\bar{x} \right)p_{\mathbf{w}}(w|\mathcal{D})dw \geq$$

(By V_{k+1}^k being non negative everywhere and by the Gaussian likelihood)

$$\int \left(\int_{f^w(x, \pi(x, k)) + \epsilon}^{f^w(x, \pi(x, k)) - \epsilon} V_{k+1}^{\pi}(\bar{x})\mathcal{N}(\bar{x}|f^w(x, \pi(x, k)), \sigma^2 \cdot I)d\bar{x} \right)p_{\mathbf{w}}(w|\mathcal{D})dw \geq$$

(By standard inequalities of integrals)

$$\int \inf_{\bar{\gamma} \in [-\epsilon, \epsilon]} V_{k+1}^{\pi}(f^w(x, \pi(x, k)) + \bar{\gamma}) \left(\int_{[-\epsilon, \epsilon]^n} \mathcal{N}(\gamma|0, \sigma^2)d\gamma \right)^n p_{\mathbf{w}}(w|\mathcal{D})dw \geq$$

(By the assumptions that for $i \neq j$ $H_{k,i}^{q,\pi}$ and $H_{k,j}^{q,\pi}$ are non-overlapping)

$$\left(\int_{[-\epsilon, \epsilon]} \mathcal{N}(\gamma|0, \sigma^2)d\gamma \right)^n \sum_{i=1}^{n_p} \int_{H_{k,i}^{q,\pi, \epsilon}} \inf_{\bar{\gamma} \in [-\epsilon, \epsilon]} V_{k+1}^{\pi}(f^w(x, \pi(x, k)) + \bar{\gamma})p_{\mathbf{w}}(w|\mathcal{D})dw,$$

(By the fact that $v_i \leq \inf_{x \in q} V_{k+1}^{\pi}(f^w(x, \pi(x, k)) + \bar{\gamma})$)

$$\left(\int_{[-\epsilon, \epsilon]} \mathcal{N}(\gamma|0, \sigma^2)d\gamma \right)^n \sum_{i=1}^{n_p} v_i \int_{H_{k,i}^{q,\pi, \epsilon}} p_{\mathbf{w}}(w|\mathcal{D})dw,$$

where the last step concludes the proof because, by the induction hypothesis, we know that for $q' \subseteq \mathbb{R}^n$

$$\inf_{\bar{x} \in q'} V_{k+1}^{\pi}(\bar{x}) \geq K_{k+1}^{\pi}(q')$$

and by the construction of sets $H_{k,i}^{q,\pi}$ for each of its weights $K_{k+1}^{\pi}(f^w(\bar{x}, \pi(x, k)))$ is lower bounded by v_{i-1} .