Certification of Iterative Predictions in Bayesian Neural Networks: Supplementary Material

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

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 ones beliefs regarding the underlying dynamics can enable more safe actions in principle [Michelmore 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

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.

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

Supplement for the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021).

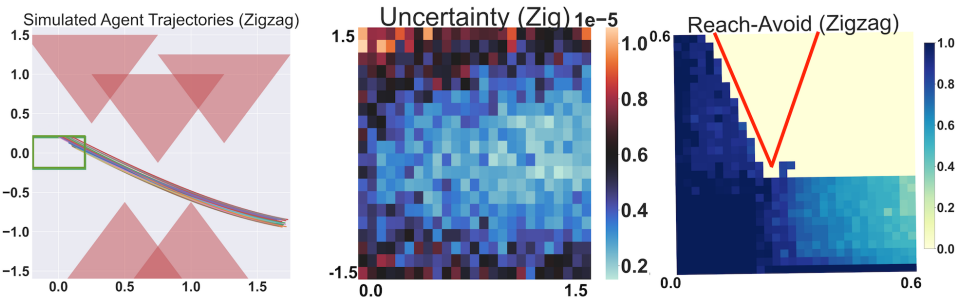

Figure 1: Analysis of the harder variant of Zigzag. **Left**: Layout of the state-space and 25 simulations from the BNN control loop demonstrates that we learn to solve the problem. **Center**: The uncertainty lines up well with what is considered in the main text and displays that the model is uncertain 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 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 5, 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 [Fan 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.

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

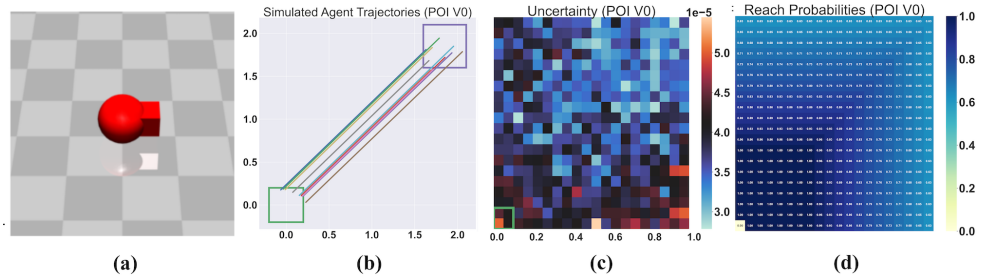

Figure 2: Far Left: A render of the Ball robot in the Mujoco simulator. **Center Left**: 25 simulations of the Ball roboto control loop. **Center Right**: The uncertainty of the BNN for the Ball robot does not show an interpretable pattern indicating that it may not be very well calibrated. **Right**: 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.

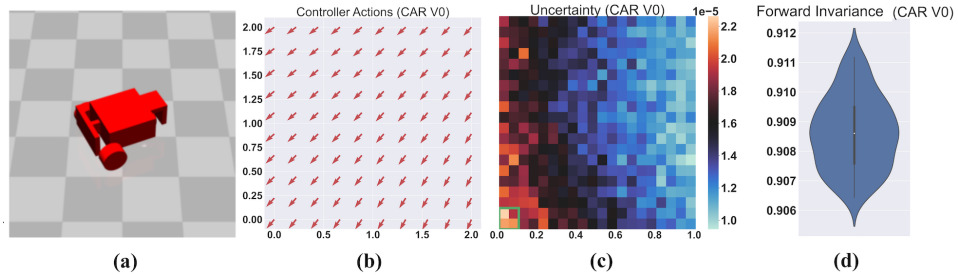

Figure 3: Far Left: A render of the Car robot in the Mujoco simulator. **Center Left**: A visualization of the control policy learned by the car robot. **Center Right**: The uncertainty of the BNN for the Car robot does not show an interpretable pattern indicating that it may not be very well calibrated. **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.

C LEARNING PARAMETERS

In this section, we use Table 1 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

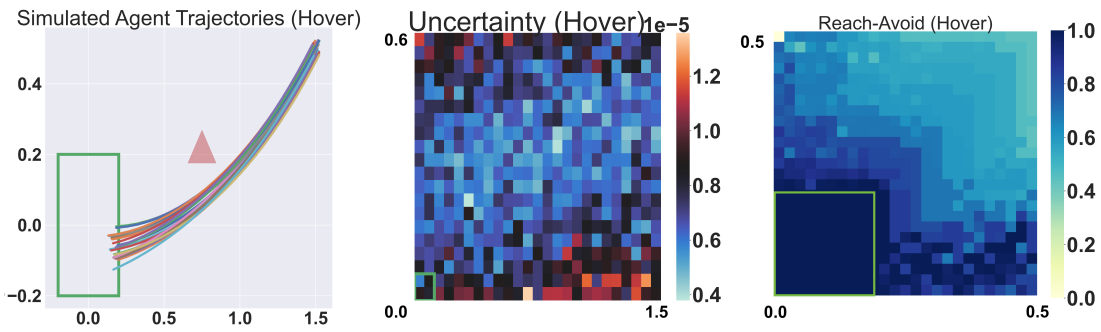

Figure 4: Left: Simulated trajectories from the hovercraft control loop demonstrates that it successfully solves the task well and navigates around the obstacle. **Center**: The uncertainty of this BNN is interpretable but may not be considered well-calibrated. **Right**: The reach-avoid property shows 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.

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 1.

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

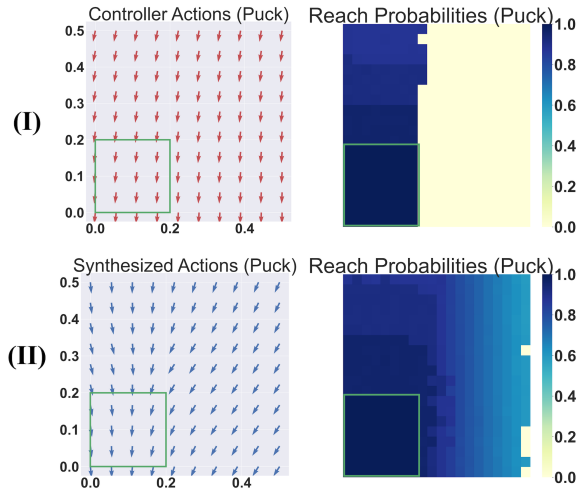

Figure 5: Left Column: Actions given by DNN controller trained with PE-TS (I) do not properly account for movement in the negative x direction compared to synthesized actions (II). **Right Column**: Verification of learned controller (I) has worse certified bounds compared to that of synthesized control actions (II).

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.

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 1: Here we report the parameters of our learning environments. Episodes referes 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.

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 chose 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

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

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

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

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

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

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

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

This can be done as follow

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

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

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

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 $\epsilon > 0$

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

where $F([-\epsilon, \epsilon] | \sigma^2) = \mathrm{erf}(\frac{\epsilon}{\sqrt{2\sigma^2}})$ 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:

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

(By definition of predictive distribution)

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

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

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

(By standard inequalities of integrals)

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

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

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

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

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

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

$$\inf_{\bar{x}\in q'} V^{\pi}_{k+1}(\bar{x}) \geq K^{\pi}_{k+1}(q')$$

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

References

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 *53rd IEEE Conference on Decision and Control*, pages 6271–6278. IEEE, 2014.

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

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.

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. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15602–15613. Curran Associates, Inc., 2020.

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.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *In Proceedings of the International Conference on Machine Learning*, 2011.

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.

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.

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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7344–7350. IEEE, 2020.