Deep Actor-Critics with Tight Risk Certificates

Bahareh Tasdighi † **Manuel Haussmann** † **Yi-Shan Wu** †
Andres R. Masegosa ‡ **Melih Kandemir** †

† Department of Mathematics and Computer Science, University of Southern Denmark, Denmark
{tasdighi,haussmann,yswu,kandemir}@imada.sdu.dk
‡ Department of Computer Science, Aalborg University, Denmark
arma@cs.aau.dk

Abstract

After a period of research, deep actor-critic algorithms have reached a level where they influence our everyday lives. They serve as the driving force behind the continual improvement of large language models through user-collected feedback. However, their deployment in physical systems is not yet widely adopted, mainly because no validation scheme quantifies their risk of malfunction. We demonstrate that it is possible to develop tight risk certificates for deep actor-critic algorithms that predict generalization performance from validation-time observations. Our key insight centers on the effectiveness of minimal evaluation data. Surprisingly, a small feasible set of evaluation roll-outs collected from a pretrained policy suffices to produce accurate risk certificates when combined with a simple adaptation of PAC-Bayes theory. Specifically, we adopt a recently introduced recursive PAC-Bayes approach, which splits validation data into portions and recursively builds PAC-Bayes bounds on the excess loss of each portion's predictor, using the predictor from the previous portion as a data-informed prior. Our empirical results across multiple locomotion tasks and policy expertise levels demonstrate risk certificates that are tight enough to be considered for practical use.

1 Introduction

Reinforcement Learning (RL) is transforming emerging AI technologies. Large language models incorporate human feedback via RL, thereby continually improving their accuracy [Christiano et al., 2017, Ziegler et al., 2019, DeepSeek-AI et al., 2025]. Generative AI is increasingly being integrated into agentic workflows to automate complex decision-making tasks. RL has also shown great promise in the control of physical robotic systems. Recent deep actor-critic algorithms learned to make a legged robot walk after only 20 minutes of outdoor training in an online mode [Kostrikov et al., 2023]. Model-based extensions of actor-critic pipelines can also achieve sample-efficient visual-control tasks in diverse settings [Hafner et al., 2025, Zhang et al., 2023]. Despite the exciting results observed in experimental conditions, RL is used far less than classical approaches in physical robot control. This opportunity has largely been missed mainly because deep RL algorithms are overly sensitive to initial conditions and can change behavior drastically during training. Embodied intelligent systems have a high risk of causing harm when their generalization performance differs significantly from their observed validation performance. Predictable generalization performance is even more critical when these systems update their behavior based on interactions with humans.

Full paper available at https://arxiv.org/pdf/2505.19682.

18th European Workshop on Reinforcement Learning (EWRL 2025).

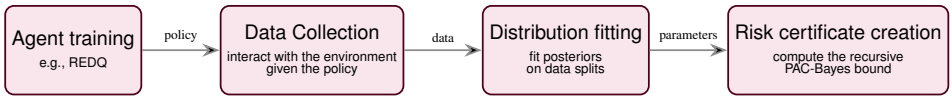


Figure 1: Our four steps to generate tight risk certificates for deep actor-critic algorithms.

There has been an effort to use learning-theoretic approaches to train high-capacity predictors with risk certificates, i.e., bounds that guarantee a predictor's generalization performance. Typically, this performance is estimated from observed validation results, which may be misleading. *Probably Approximately Correct Bayesian (PAC-Bayes) theory* [McAllester, 1999, Alquier et al., 2024] provides risk certificates for stochastic predictors, relative to a prior distribution over the hypothesis space. In this framework, the computationally prohibitive capacity term is reduced to a Kullback-Leibler divergence between the posterior and the prior, enabling the incorporation of domain knowledge into the analysis. Since we often deal with stochastic policies, relying on PAC-Bayes is a natural choice.

PAC-Bayes is the first and remains the most promising method for providing meaningful risk certificates to deep neural networks [Dziugaite and Roy, 2017, Pérez-Ortiz et al., 2021, Lotfi et al., 2022]. Further studies have improved the tightness, i.e., precision, of these certificates through the following techniques: (i) pretraining probabilistic neural nets on held-out data and using them as *data-informed priors* [Ambroladze et al., 2006, Dziugaite et al., 2021]; (ii) using pretrained networks as first-step predictors and developing PAC-Bayes guarantees on the residual of their predictions, termed the *excess loss*; and (iii) recursively repeating the first two steps on multiple data splits, a recent method known as the *Recursive PAC-Bayes* [Wu et al., 2024]. The scope of these exciting developments has thus far been limited to simple classification tasks with feedforward neural networks. Their application to deep actor-critic algorithms remains open, primarily because the mainstream PAC-Bayes bounds assume i.i.d. datasets, whereas RL assumes a controlled Markov chain.

We present a simple recipe for providing risk certificates for deep model-free actor-critic architectures. We find that, contrary to what one might expect, the three modern PAC-Bayesian learning tools mentioned above can successfully handle the high variance of Monte Carlo samples collected by running a pretrained policy network for multiple episodes in evaluation mode. Our approach proposes self-certified training of probabilistic neural networks on different splits of an i.i.d. data set containing return realizations of the policy, computed by first-visit Monte Carlo and post-processed through a simple thinning approach. We recursively build a PAC-Bayes bound on the excess losses of these networks, following a new adaptation of the recipe introduced by Wu et al. [2024]. Figure 1 illustrates our risk-certificate generation workflow. Our results highlight that the risk certificates get significantly tighter as the recursion depth increases. The final bounds are tight enough for practical use. Furthermore, the tightness of the risk certificates is proportional to the policy's level of expertise.

2 Background

2.1 The state of the art of model-free deep actor-critic learning

Consider a set of states $\mathcal{S}$ an agent may be in and an action space $\mathcal{A}$ from which the agent can choose actions to interact with its environment. Denote by $\Delta(\mathcal{S})$ and $\Delta(\mathcal{A})$ the sets of probability distributions defined on $\mathcal{S}$ and $\mathcal{A}$, respectively. We define a Markov Decision Process (MDP) [Puterman, 2014] as the tuple $M = \langle \mathcal{S}, \mathcal{A}, r, P, P_0, \gamma \rangle$, where $r : \mathcal{S} \times \mathcal{A} \to [0, R]$ is a bounded reward function, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state-transition kernel conditioned on a state-action pair; specifically $P(s'|s, a)$ is the probability distribution of the next state $s' \in \mathcal{S}$ given the current state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$. We denote the initial-state distribution by $P_0 \in \Delta(\mathcal{S})$, the discount factor by $\gamma \in (0, 1)$, and let $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ be a policy. The goal of RL is to learn a policy that maximizes the expected discounted return, $\pi_* := \arg\max_{\pi \in \Pi} \mathbb{E}_{\tau_\pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$. The expectation is taken with respect to the trajectory $\tau_\pi := (s_0, a_0, s_1, a_1, s_2, a_2, \ldots)$ of states and actions generated when a policy π chosen from a feasible set Π is executed. We refer to π_* as the optimal policy. The exact Bellman operator for a policy π is defined as

$$T_\pi Q(s, a) := r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[Q(s', \pi(s'))\right] \tag{1}$$

for some function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The unique fixed point of this operator is the true action-value function Q_π, which maps a state-action pair (s, a) to the expected discounted

sum of rewards the policy π collects when executed from (s, a). In other words, the equality $T_\pi Q(s,a) = Q(s,a)$ holds if and only if $Q(s,a) = Q_\pi(s,a), \forall (s,a)$. Any other Q incurs an error $(T_\pi Q(s,a) - Q(s,a))^2$, called the *Bellman error*. Common deep actor-critic methods approximate the true action-value function Q_π by one-step Temporal Difference (TD) learning that minimizes $L(Q,\pi) := \mathbb{E}_{s\sim P_\pi}[(T_\pi Q(s,a) - Q(s,a))^2]$ with respect to Q, given a data set $\mathcal{D}$ and $P_\pi(s' \in A) = \mathbb{E}_{s\sim P_0}\left[\sum_{t>0} P(s_t \in A | s_0 = s, \pi(s))\right]$ which is defined as the state-visitation distribution of policy π for some event A that belongs to the σ-algebra of the transition probability distribution. Because the transition probabilities are unknown, the expectation term in Eq. 1 cannot be computed. Instead, the observed transitions are used to approximate it with a single-sample Monte Carlo estimate, yielding the training objective below:

$$\widetilde{L}(Q) := \mathbb{E}_{s\sim P_\pi}\Big[\mathbb{E}_{s'\sim P(\cdot|s,\pi(s))}\left[(r(s,a) + \gamma Q(s',\pi(s')) - Q(s,a))^2\right]\Big].$$

A deep actor-critic algorithm fits a neural-network function approximator Q, referred to as the critic, to a set of observed tuples (s, a, s') stored in a replay buffer $\mathcal{D}$ by minimizing an empirical estimate of the stochastic loss: $\widehat{L}_{\mathcal{D}}(Q) := 1/|\mathcal{D}| \sum_{(s,a,s')\in\mathcal{D}} (\widetilde{T}_\pi Q(s,a,s') - Q(s,a))^2$. The critic is then used to train a policy network, or actor, $\pi' \leftarrow \arg\max_\pi \mathbb{E}_{s\sim P_\pi}[Q(s,\pi(s))]$. It is common practice to adopt the *Maximum-Entropy Reinforcement Learning* approach [Haarnoja et al., 2018a,b] to balance exploration and exploitation, thereby ensuring effective training. The approach supplements the reward function with a policy-entropy term $r_{\text{MaxEnt}}(s,a) = r(s,a) + \alpha \mathbb{H}[\pi(\cdot|s)]$, where $\alpha \geq 0$ is a scaling hyperparameter tuned jointly with the actor and critic.

Performing off-policy TD learning with deep neural nets is notoriously unstable which is often attributed to the *deadly triad* [Sutton and Barto, 2018]. The main source of instability is the accumulation of errors from approximating $T_\pi Q$ by its Monte Carlo estimate. Strategies to improve stability include maintaining Polyak-updated target networks [Lillicrap et al., 2016] and learning twin critics while using the minimum of their target-network outputs in Bellman target calculation [Fujimoto et al., 2018]. Empirically, training an ensemble of critic networks in a maximum-entropy setup largely mitigates these stability issues. We adopt REDQ [Chen et al., 2021], a state-of-the-art actor-critic method for model-free continuous control, as our representative approach. This choice is pragmatic rather than restrictive allowing us to trade the computational cost of a broader exploration of algorithms for a deeper, more comprehensive empirical evaluation of a single one.

2.2 Developing risk certificates with PAC-Bayes bounds

PAC-Bayes [McAllester, 1999, Alquier et al., 2024] offers a powerful way to understand and control how well learning algorithms generalize by blending prior beliefs with what we learn from data. *PAC-Bayesian learning* uses modern machine learning techniques to model ρ with complex function approximators and fit them to data. It has been successfully applied in both image classification [Dziugaite and Roy, 2017, Wu et al., 2024] and regression tasks [Reeb et al., 2018]. Its application to reinforcement learning has so far been limited to the design of critic training losses without rigorously quantifying the tightness of the performance guarantees [Tasdighi et al., 2024a,b].

Notation. Let $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ be a set of feasible hypotheses and $\ell : \mathcal{Y} \times \mathcal{Y} \to [0,1]$ be a bounded loss function.[1] Further, let $L(h) = \mathbb{E}_{(x,y)\sim P_D}[\ell(h(x), y)]$ be the expected error, where P_D is a distribution on $\mathcal{X} \times \mathcal{Y}$. The empirical loss is $\hat{L}(h) = \frac{1}{N}\sum_{i=1}^{N} \ell(h(x_i), y_i)$ for a data set $\mathcal{D} = \{(x_n, y_n) : n \in \{1, \ldots, N\}\}$ of size N with $(x_n, y_n) \sim P_D$. $\mathcal{P}$ is the set of distributions on $\mathcal{H}$. For two distributions ρ, ρ_0 on $\mathcal{H}$, the Kullback-Leibler (KL) divergence is defined as $\mathrm{KL}(\rho \,\|\, \rho_0) \triangleq \mathbb{E}_{h\sim\rho}[\log \rho(h) - \log \rho_0(h)]$. We use $\mathrm{kl}(p \,\|\, q) \triangleq p \log(p/q) + (1-p)\log((1-p)/(1-q))$ to denote the KL divergence between two Bernoulli distributions. PAC-Bayesian analysis [McAllester, 1999, Shawe-Taylor and Williamson, 1997, Alquier et al., 2024] develops bounds on the *expected loss* $\mathbb{E}_{h\sim\rho}[L(h)]$, under a posterior distribution ρ with respect to a prior distribution ρ_0, that hold with high probability. That is, they provide *risk certificates* for the generalization error. For brevity, we will use $\mathbb{E}_\rho[\cdot] = \mathbb{E}_{h\sim\rho}[\cdot]$ throughout this paper. In the context of PAC-Bayes, the terms *posterior* and *prior* refer to distributions dependent and independent of the validation data, respectively. They are not to be understood in a Bayesian manner as being linked by a likelihood.[2] Which bounds one should choose to get the tightest risk

[1] Our discussion generalizes directly to any bounded loss within an interval $[a, b]$ with $a, b \in \mathbb{R}$.

[2] See Germain et al. [2016] for results linking PAC-Bayes and Bayesian inference.

certificates depends on the specific use case; see, e.g., Alquier et al. [2024] for a recent introduction and a survey of various bounds. In this work we rely on bounds derived from the kl divergence as they are tighter than the alternatives when no additional information about the data distribution is available, while noting that the same arguments apply to any other PAC-Bayesian bound.

2.2.1 PAC-Bayes-kl bound

Assuming the definitions given above, the *PAC-Bayes-*kl *bound* is given by

Theorem 2.1 (PAC-Bayes-kl bound [Seeger, 2002, Maurer, 2004])**.** *For any probability distribution $\rho_0 \in \mathcal{P}$ that is independent of $\mathcal{D}$ and any $\delta \in (0,1)$, we have*

$$\mathbb{P}\Big(\exists \rho \in \mathcal{P} : \mathrm{kl}\big(\mathbb{E}_\rho[\hat{L}(h)] || \mathbb{E}_\rho\left[L(h)\right]\big) \geq \big(\mathrm{KL}\left(\rho \parallel \rho_0\right) + \ln(2\sqrt{N}/\delta)\big)/N\Big) \leq \delta.$$

Proof. See, e.g., Maurer [2004] for a proof of the bound. □

We define the upper inverse of $\mathrm{kl}\left(\cdot \parallel \cdot\right)$ as $\mathrm{kl}^{-1,+}(\hat{p}, \varepsilon) \triangleq \max\{p : p \in [0,1] \mid \mathrm{kl}\left(\hat{p} \parallel p\right) \leq \varepsilon\}$ and the lower one as $\mathrm{kl}^{-1,-}(\hat{p}, \varepsilon) \triangleq \min\{p : p \in [0,1], \mathrm{kl}\left(\hat{p} \parallel p\right) \leq \varepsilon\}$ and cite the following inequality.

Lemma 2.2 (kl-inequality [Langford, 2005, Foong et al., 2021, 2022])**.** *Let $Z_1, \ldots, Z_N$ be i.i.d. random variables taking values on an interval $[0,1]$ and $\mathbb{E}\left[Z_n\right] = p$ for all n. Let their empirical mean be $\hat{p} = \frac{1}{N}\sum_{n=1}^{N} Z_n$. Then, for any $\delta \in (0,1)$ we have*

$$\mathbb{P}\left(\mathrm{kl}(\hat{p}||p) \geq \ln(1/\delta)/N\right) \leq \delta,$$

the inverse of which is given by

$$\mathbb{P}\left(p \geq \mathrm{kl}^{-1,+}\left(\hat{p}, \ln(1/\delta)/N\right)\right) \leq \delta, \quad \textit{and} \quad \mathbb{P}\left(p \leq \mathrm{kl}^{-1,-}\left(\hat{p}, \ln(1/\delta)/N\right)\right) \leq \delta.$$

Proof. See Langford [2005], Corollary 3.7 for a proof of the bound. □

2.2.2 PAC-Bayes-Split-kl bound

Wu and Seldin [2022] generalize these bounds to random variables that take values in intervals $[a,b]$ splitting each into two components that individually satisfy the constraints of the kl-inequality.

Let $Z \in [a,b]$, with $a, b \in \mathbb{R}$, be a random variable and set $p = \mathbb{E}\left[Z\right]$. For $\mu \in [a,b]$ define $Z^+ = \max\{0, Z-\mu\}$ and $Z^- = \max\{0, \mu - Z\}$, so that $Z = \mu + Z^+ - Z^-$. Let $p^+ = \mathbb{E}\left[Z^+\right]$ and $p^- = \mathbb{E}\left[Z^-\right]$ be their respective expectations, and let $\hat{p}^+ = \frac{1}{N}\sum_{n=1}^{N} Z_n^+$ and $\hat{p}^- = \frac{1}{N}\sum_{n=1}^{N} Z_n^-$ be their empirical means for an i.i.d. sample $Z_1, \ldots, Z_N$. The *split-*kl *inequality* is stated below.

Lemma 2.3 (Split-kl inequality [Wu and Seldin, 2022])**.** *For any $\mu \in [a,b]$ and $\delta \in (0,1)$*

$$\mathbb{P}\left(p \leq \mu + (b-\mu)\mathrm{kl}^{-1,+}\left(\frac{\hat{p}^+}{b-\mu}, \frac{\ln(2/\delta)}{N}\right) - (\mu - a)\mathrm{kl}^{-1,-}\left(\frac{\hat{p}^-}{\mu - a}, \frac{\ln(2/\delta)}{N}\right)\right) \geq 1 - \delta.$$

Proof. The lemma follows by applying Lemma 2.2 to each of the kl terms and a union bound. □

For the PAC-Bayesian analogue, define $\tilde{\ell} : \mathcal{Y} \times \mathcal{Y} \to [a,b]$, where $a, b \in \mathbb{R}$. For $\mu \in [a,b]$, define $\tilde{\ell}^+ = \max\{0, \tilde{\ell} - \mu\}$ and $\tilde{\ell}^- = \max\{0, \mu - \tilde{\ell}\}$. $\tilde{L}^+(h) = \mathbb{E}_{(x,y)\sim P_D}[\tilde{\ell}^+(h(x), y)]$ and $\hat{\tilde{L}}^+(h) = \frac{1}{N}\sum_{n=1}^{N} \tilde{\ell}^+(h(x_n), y_n)$ are the expected and empirical losses. L^- and $\hat{\tilde{L}}^-$ are defined analogously. With these definitions, we now cite the PAC-Bayes-split-kl inequality.

Theorem 2.4 (PAC-Bayes-Split-kl inequality [Wu and Seldin, 2022])**.** *Let $\tilde{\ell}$ and the remaining loss terms be defined as above. Then for any ρ_0 on $\mathcal{H}$ independent of $\mathcal{D}$, any $\mu \in [a,b]$, and any $\delta \in (0,1)$*

$$\mathbb{P}\Bigg(\exists \rho \in \mathcal{P} : \mathbb{E}_\rho[\tilde{L}(h)] \geq \mu + (b-\mu)\mathrm{kl}^{-1,+}\left(\frac{\mathbb{E}_\rho[\hat{\tilde{L}}^+(h)]}{b-\mu}, \frac{\mathrm{KL}\left(\rho \parallel \rho_0\right) + \ln(4\sqrt{N}/\delta)}{N}\right)$$
$$- (\mu - a)\mathrm{kl}^{-1,-}\left(\frac{\mathbb{E}_\rho[\hat{\tilde{L}}^-(h)]}{\mu - a}, \frac{\mathrm{KL}\left(\rho \parallel \rho_0\right) + \ln(4\sqrt{N}/\delta)}{N}\right)\Bigg) \leq \delta.$$

Proof. The theorem follows by applying Lemma 2.3 to the decomposition $\mathbb{E}_\rho[\tilde{L}(h)] = \mu + \mathbb{E}_\rho[\tilde{L}^+(h)] - \mathbb{E}_\rho[\tilde{L}^-(h)]$. □

2.2.3 Recursive PAC-Bayes bound

Data-informed prior. The tightness of PAC-Bayesian bounds is dominated by the KL divergence between the posterior ρ and the prior ρ_0. The better the prior guess is, the tighter the bound. Because the prior must be be independent of the observed data, a common choice is to select a prior that is as uniform as possible over the hypothesis space. To improve upon this naïve choice, Ambroladze et al. [2006] proposed splitting the observed data into two disjoint subsets S_0 and S_1, i.e., $\mathcal{D} = S_0 \cup S_1$, using S_0 to infer a *data-informed prior* and S_1 to subsequently evaluate the bound. This approach balances the benefit of a better prior with the cost of having fewer observations to evaluate the bound.

Excess loss. The *excess loss* $L^{\text{exc}}(h)$ with respect to a reference hypothesis $h^* \in \mathcal{H}$ is defined as $L^{\text{exc}}(h) = L(h) - L(h^*)$. The excess-loss concept allows us to decompose the expected loss as $\mathbb{E}_\rho\left[L(h)\right] = \mathbb{E}_\rho\left[L(h) - L(h^*)\right] + L(h^*)$. Using S_0 to construct both the prior ρ_0 and the reference h^* , Mhammedi et al. [2019] showed that, assuming $L(h^*)$ is close to $L(h)$, the excess loss has lower variance and thus yields a more efficient bound, while a bound on $L(h^*)$ is independent of $\mathrm{KL}\left(\rho \parallel \rho_0\right)$ and can be obtained using standard generalization guarantees.

Recursive PAC-Bayes. Wu et al. [2024] generalized the excess loss further by introducing a scaling factor $\kappa < 1$ to maintain a diminishing effect of recursions: $\mathbb{E}_\rho\left[L(h)\right] = \mathbb{E}_\rho\left[L(h) - \kappa\mathbb{E}_{\rho_0}\left[L(h^*)\right]\right] + \kappa\mathbb{E}_{\rho_0}\left[L(h^*)\right]$. Here, the first term reflects the excess loss with respect to a scaled version of the expected reference hypothesis loss under the prior ρ_0. The second term in turn is an expected loss again similar to the one on the left-hand side of the equation. Instead of adhering to a binary split $\mathcal{D} = S_0 \cup S_1$ such that $S_0 \cap S_1 = \emptyset$, they propose to extend this decomposition recursively, by partitioning $\mathcal{D}$ into T disjoint subsets, $\mathcal{D} = \bigcup_{t=1}^T S_t$ and and they define $S_{\leq t} = \bigcup_{s=1}^t S_s$ and $S_{\geq t} = \bigcup_{s=t}^T S_s$. Their recursion is given by

$$\mathbb{E}_{\rho_t}\left[L(h)\right] = \mathbb{E}_{\rho_t}\left[L(h) - \kappa_t\mathbb{E}_{\rho_{t-1}}\left[L(h)\right]\right] + \kappa_t\mathbb{E}_{\rho_{t-1}}\left[L(h)\right], \tag{2}$$

for $t \geq 2$, and $\kappa_1, \ldots, \kappa_T$ are scaling factors. The distributions $\rho_1, \ldots, \rho_T \in \mathcal{H}$ form a sequence such that ρ_t depends solely on $S_{\leq t}$ and $S_{\geq t}$ to estimate $\mathbb{E}_{\rho_t}\left[L(h)\right]$.

While Wu et al. [2024] formulate their final recursive bound directly for a zero-one loss and PAC-Bayes split-kl bounds [Wu and Seldin, 2022], we present their result first in a general loss-agnostic form before we construct a specific bound in the next section.

Theorem 2.5. *(*Recursive PAC-Bayes bound.*) Let $\mathcal{D} = S_1 \cup \cdots \cup S_T$ be a disjoint decomposition of the set of observations $\mathcal{D}$. Let $S_{\leq t}$ and $S_{\geq t}$ be as defined above, $N = |\mathcal{D}|$, and $N_t = |S_{\geq t}|$. Let $\kappa_1, \ldots, \kappa_T$ be a sequence of scaling factors, where κ_t is allowed to depend on $S_{\leq t-1}$. Let $\mathcal{P}_t$ be the set of distributions on $\mathcal{H}$ which are allowed to depend on $S_{\leq t}$, and $\rho_t \in \mathcal{P}_t$. Then, for any $\delta \in (0,1)$,*

$$\mathbb{P}\left(\exists t \in [T], \rho_t \in \mathcal{P}_t \text{ such that } \mathbb{E}_{\rho_t}\left[L(h)\right] \geq \mathcal{B}_t(\rho_t)\right) \leq \delta,$$

where $\mathcal{B}_t(\rho_t)$ is a generic PAC-Bayesian bound on $\mathbb{E}_{\rho_t}\left[L(h)\right]$ defined recursively as follows.

$$\mathcal{B}_t(\rho_t) = \mathcal{E}_t(\rho_t, \kappa_t) + \kappa_t\mathcal{B}_{t-1}(\rho^*_{t-1}),$$

where $\mathcal{B}_1(\rho_1)$ is a PAC-Bayes bound on $\mathbb{E}_{\rho_1}\left[L(h)\right]$ with an uninformed prior and $\mathcal{E}_t(\rho_t, \kappa_t)$ is a PAC-Bayes bound on the excess loss $\mathbb{E}_{\rho_t}\left[L(h) - \kappa_t\mathbb{E}_{\rho^*_{t-1}}\left[L(h')\right]\right]$.

Proof. Because $\mathcal{B}_1(\rho_1)$ and $\mathcal{E}_t(\rho_t, \kappa_t)$ are PAC-Bayes bounds by assumption, we have

$$\mathbb{P}\left(\exists \rho_1 \in \mathcal{P}_1 : \mathbb{E}_{\rho_1}\left[L(h)\right] \geq \mathcal{B}_1(\rho_1)\right) \leq \delta/T,$$

$$\text{and} \quad \mathbb{P}\left(\exists \rho_t \in \mathcal{P}_t : \mathbb{E}_{\rho_t}\left[L(h) - \kappa_t\mathbb{E}_{\rho^*_{t-1}}\left[L(h')\right]\right] \geq \mathcal{E}_t(\rho_t, \kappa_t)\right) \leq \delta/T \text{ for } t \in \{2, \ldots, T\}.$$

The claim follows by expected loss decomposition and the recursion. □

3 Recursive PAC-Bayesian risk certificates for reinforcement learning

Obtaining risk certificates involves four steps, following our conceptual structure in Figure 1.

(i) Training an agent. The chosen actor-critic algorithm, REDQ [Chen et al., 2021], which we use in our experiments, is trained until convergence or until a computational budget is exhausted, after which we freeze its policy parameters, e.g., the weights of the corresponding neural net.

(ii) Collecting data. After training the policy, we run an agent acting according to this policy for several episodes. Although a PAC-Bayesian bound gets tighter as the number of data points increases, we observe that even a relatively small number of evaluation roll-outs is sufficient to get tight results.

(iii) Fitting the posteriors. We rely on the discounted return as the prediction target rather than a plain sum of rewards for several reasons. Short-term risks tend to be more relevant for decisions, as longer-term risks depend on an increasing set of external, usually unaccountable, factors. Discounted rewards also serve as a proxy for lifelong learning and policy evaluation as they generalize to non-episodic data. That said, even though the original policy might be trained on discounted returns in step *(i)*, a valid bound could also be constructed by computing the non-discounted rewards from data collected in *(ii)*. As discussed in Section 2.2.3, we split the data into T disjoint subsets and train a series of T last-layer Bayesian neural nets via first-visit Monte Carlo to infer distributions over $S_{\leq t}$.

(iv) Construction of the bound. As discussed above, we focus on a generally well-performing set of kl-based bounds. We construct the following bounds for $\mathcal{B}_1$ and $\mathcal{E}_t$ ($t \in \{1, \dots, T\}$).

A bound for $\mathcal{B}_1$. As $\hat{L}(h)$ is bounded between $[0, B]$, we rescale its expectation and choose

$$\mathcal{B}_1(\rho_1) = B\mathrm{kl}^{-1,+}\left(\frac{\mathbb{E}_\rho[\hat{L}(h)]}{B}, \frac{\mathrm{KL}\left(\rho_1 \parallel \rho_0^*\right) + \ln(2T\sqrt{N}/\delta)}{N}\right),$$

where ρ_0^* is a data-independent prior distribution on $\mathcal{H}$. Given the result in Theorem 2.1, this is a PAC-Bayesian bound on $\mathbb{E}_{\rho_1}\left[L(h)\right]$, i.e., $\mathbb{P}\big(\exists \rho_1 \in \mathcal{P}_1 : \mathbb{E}_{\rho_1}\left[L(h)\right] \geq \mathcal{B}_1(\rho_1)\big) \leq \delta/T$.

A bound for $\mathcal{E}_t$. Let $L_t^{\mathrm{exc}}(h) = L(h) - \kappa_t \mathbb{E}_{\rho_{t-1}}\left[L(h')\right] \in [-\kappa_t B, B]$. For $\mu \in [-\kappa_t B, B]$, define $L_t^{\mathrm{exc}+}(h) = \max\{0, L_t^{\mathrm{exc}}(h) - \mu\}$ and $L_t^{\mathrm{exc}-}(h) = \max\{0, \mu - L_t^{\mathrm{exc}}(h)\}$, with $\hat{L}_t^{\mathrm{exc}+}(h)$ and $\hat{L}_t^{\mathrm{exc}-}(h)$ as their empirical analogous. We set

$$\begin{aligned}\mathcal{E}_t(\rho_t) = \mu &+ (B-\mu)\mathrm{kl}^{-1,+}\left(\frac{\mathbb{E}_{\rho_t}[\hat{L}_t^{\mathrm{exc}+}(h)]}{B-\mu}, \frac{\mathrm{KL}\left(\rho_t \parallel \rho_{t-1}^*\right) + \ln(4T\sqrt{N_t}/\delta)}{N_t}\right) \\ &- (\mu + \kappa_t B)\mathrm{kl}^{-1,-}\left(\frac{\mathbb{E}_{\rho_t}[\hat{L}_t^{\mathrm{exc}-}(h)]}{\mu + \kappa_t B}, \frac{\mathrm{KL}\left(\rho_t \parallel \rho_{t-1}^*\right) + \ln(4T\sqrt{N_t}/\delta)}{N_t}\right),\end{aligned}$$

where ρ_{t-1}^* is a distribution on $\mathcal{H}$ informed by $S_{\leq t-1}$. Via Theorem 2.4 this is a PAC-Bayesian bound on $\mathbb{E}_{\rho_t}\left[L_t^{\mathrm{exc}}\right]$ that holds with a probability greater than $1 - \delta/T$, i.e.,

$$\mathbb{P}\big(\exists \rho_t \in \mathcal{P}_t \text{ such that } \mathbb{E}_{\rho_t}\left[L_t^{\mathrm{exc}}(h)\right] \geq \mathcal{E}_t(\rho_t)\big) \leq \delta/T.$$

Applying this construction recursively with T steps therefore gives us a recursive PAC-Bayesian bound that holds with probability greater than $1 - \delta$.

4 Experiments

We perform experiments to answer the following three questions: (**Q1**) Can the test-time return of a policy π be predicted with high precision across a range of environments and policies of varying expertise? (**Q2**) What is the influence of a PAC-Bayes bound's structure? (**Q3**) How does the validation set size influence the tightness of the risk certificate guarantee?

4.1 Experiment design

To evaluate our certificate-generation pipeline at an error tolerance of $\delta = 0.025$, we choose REDQ [Chen et al., 2021] as a representative state-of-the-art, sample-efficient, model-free continuous control algorithm. All REDQ hyperparameters follow those in the original paper. We first train a REDQ agent for 300 000 steps using an ensemble of ten critics, randomly sampling two at each Bellman-target evaluation for min-clipping. The learned policy is then run in evaluation mode for 100 episodes. The resulting state transitions and rewards are stored as the data set used for bound fitting. Subsequently, we run the trained policy for another 100 episodes to obtain a test dataset to compute a proxy for the

Figure 2: *Correlation plots.* PAC-Bayes bounds, one in each column, are plotted on the x-axis against true test errors on the y-axis for each method across all environments, policy instances, and repetitions to visualize correlation. Environments are color-coded as follows: `Ant` (blue circle), `Half-Cheetah` (orange cross), `Hopper` (green square), `Humanoid` (red plus), and `Walker2d` (purple diamond).

generalization performance. We predict the discounted return of the policy on the test set by fitting a PAC-Bayes bound using observations from the validation set.

We evaluate and compare the final posterior loss ρ on the full training data, and on the held-out test data, alongside the corresponding PAC-Bayes bounds across all methods and environments. To mitigate the overfitting common in continuous-control settings, where consecutive samples are highly correlated, we apply a thinning strategy that reduces redundancy while preserving data diversity. Full details on each experiment are provided in Appendix D. We provide an implementation at `anonymous`.

Policy instances. We define a policy instance as the output of a single policy-training round. In our experiments, we consider five policy instances, each obtained by running the REDQ algorithm with a different initial seed. Due to the stochastic nature of initialization and training, each instance follows a unique trajectory. We construct individual bounds for each instance and report them in Appendix D. To account for randomness in the risk certificate generation process, we repeat the procedure five times for every policy instance. To address question (Q2), we create separate risk certificates for three training stages of each policy, each reflecting a different level of expertise: *Starter (S)* for a policy trained for 100 000 steps, *Intermediate (I)* for 200 000 steps, and *Expert (E)* for 300 000 steps, after which no performance improvement observed.

Environments. We evaluate five MuJoCo environments: `Ant`, `Half-Cheetah`, `Hopper`, `Humanoid`, and `Walker2d` [Todorov et al., 2012] due to their widespread use in the community and the representative value of the platforms for real-world use cases. Risk certificates may be particularly interesting for mobile platforms that interact with their surroundings as well as humans.

Baselines. We design our baselines with the following points in mind: 1. how well a PAC-Bayes bound predicts test-time performance, 2. whether informative priors yield tighter guarantees, 3. whether the bound gets tighter when the recursive scheme is used, and 4. whether increasing the recursion depth improve tightness. As this is the first work to evaluate generalization bounds tailored for continuous control with deep actor-critics, there are no existing baselines for comparison. We consider two non-recursive (NR) baselines: *non-informed (NR-NI)*, a PAC-Bayes-kl inverse bound (see Theorem 2.1) with a non-informative prior that is independent of the training data, and *informed (NR-I)*, a data-informed variant in which the dataset is split equally into $\mathcal{D} = \mathcal{D}_{\text{prior}} \cup \mathcal{D}_{\text{bound}}$, allowing the prior to depend on $\mathcal{D}_{\text{prior}}$ and the empirical loss to be computed on $\mathcal{D}_{\text{bound}}$. We evaluate two recursion (R) depths, *depth two (R-I T=2)* and *depth six (R-I T=6)*, to test the effect of recursion.

Performance metrics. We evaluate the bounds based on three metrics: *Normalized bound value*: To ensure comparability across environments with different reward scales, we normalize the squared

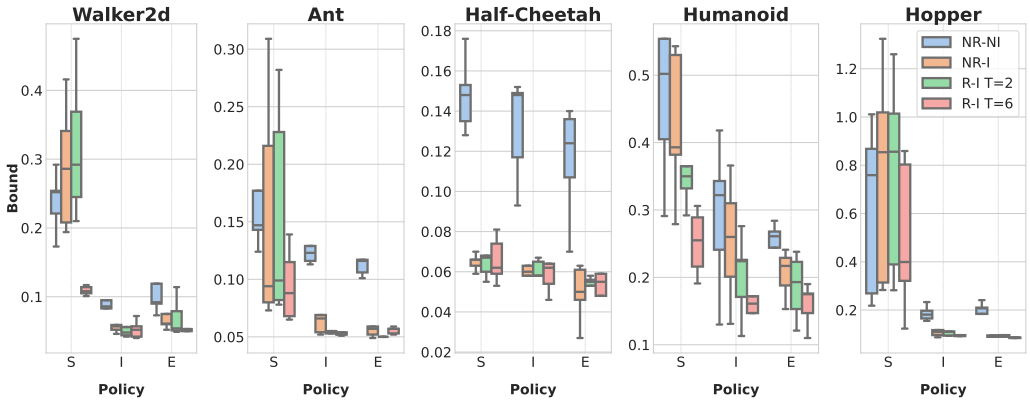


Figure 3: *Bound values.* Normalized bound values for all baselines across five MuJoCo environments over three policy qualities. Results are aggregated over all policy instances and repetitions.

discounted return prediction errors by the maximum observed return during training. A value close to zero implies that the bound closely follows the actual returns. *Tightness*: The difference between the predicted bound and the actual test error; smaller values indicate more accurate estimates of the discounted return prediction error. *Correlation*: We expect a linear correlation between the risk certificates and the observed test errors across policy instances.

Computational requirements. We conduct our experiments on a single computer equipped with a GeForce RTX 4090 GPU, an Intel(R) Core(TM) i7-14700K CPU (5.6 GHz), and 96 GB of memory. Training five policy instances to convergence in each environment takes about 30 minutes per instance, totaling 150 minutes. Collecting validation and test episodes requires around 20 minutes per policy level, or 60 minutes in total. Model training and PAC-Bayes bound computation across five policy instances, five repetitions, four baselines, and three policies takes four minutes per run, totaling roughly 1200 minutes per environment, 7000 minutes in total (about five days).

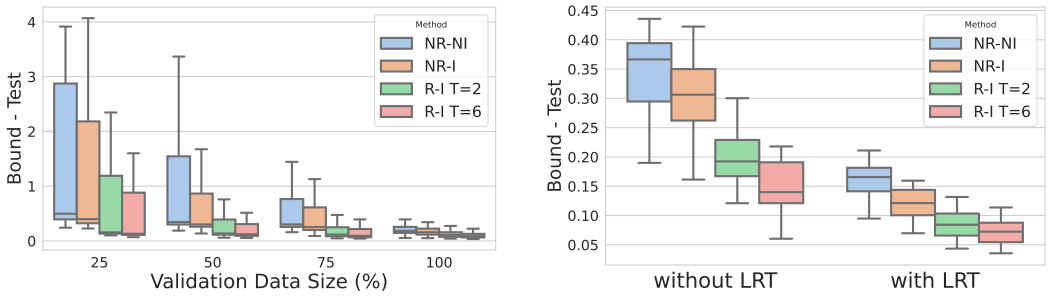
4.2 Results

We present full results on every environment, policy instance and repetition in Appendix D and restrict ourselves to discussing aggregated results in the main text.

Strong correlation between bounds and test errors. In Figure 2, we present scatter plots of all the PAC-Bayes bounds discussed in 4.1, policy instances, and repetitions against their respective test set errors across environments and levels of policy expertise. For every bound, the correlation between the bound and the test error increases with policy expertise. Within a fixed expertise level, the correlation also improves as the bound becomes more advanced, a trend that is already evident in more noisy *starter* policy. For example, in the brittle `Hopper` environment, which exhibits the weakest correlations overall, moving from NR-NI to R-I with T=6 raises the Pearson correlation from 0.4 to 0.65. At higher expertise levels, our recursive bounds achieve correlations above 0.9 in almost all environments. Overall we see a clear linear trend, which demonstrates that our bounds are tight. There appears an increasing scatter as the expertise level decreases. This is expected, as the effects of an unconverged policy function on environment dynamics are less predictable. The bounds therefore provide a good prediction of the test-time return, answering Q1.

Tightness improves with recursive depth. In Figure 3 we plot the normalized bounds aggregated over policy instances and repetitions for each of the five environments. Smaller values reflect tighter bounds. Data-informed priors improve bounds across all environments for intermediate and expert policies, though this effect is less clear for the starter level policy. Introducing recursion (R-I, with T=2 and T=6) further tightens bounds, with deeper recursion generally yielding the tightest results. These improvements are most evident in environments with brittle dynamics such as `Humanoid` and `Hopper` where the locomotor has to keep its balance and less so in simpler environments such as `Half-Cheetah`. We see that while the correlation between bound and test-set error is already high, better, recursive, bounds provide improved tightness guarantees answering Q2.

Recursion improves sample efficiency. Collecting validation data from physical robots is often costly. Hence, the sample efficiency of a risk-certificate generation pipeline is of particular interest.



(a) The effect of validation data size on tightness

(b) The effect of local reparameterization on tightness.

Figure 4: *Bound tightness; smaller is better. Results are provided for the* `Humanoid` *environment, using five policy instances and five repetitions.* *(a)* Tightness scores aggregated across three policy qualities and various validation set sizes, expressed as percentages of the full validation dataset. *(b)* Effect of the local reparameterization trick on bound tightness, illustrated for the expert-level policy.

Figure 4a shows the tightness scores of the bounds across different validation data sizes in the `Humanoid` environment, while keeping the test size fixed. As expected, larger validation sets lead to tighter bounds, but the effect is most pronounced for our proposed recursive bounds. R-I T=6 achieves tightness results comparable to those that the non-recursive bounds (non-informed, NR-NI, and data-informed, NR-I) attain with twice as many data points. These findings demonstrate the ability of recursive bounds to significantly improve sample efficiency, addressing Q3.

Local reparameterization improves tightness. To train our model, we use a Bayesian neural network (BNN) that represents uncertainty by learning distributions over neural network parameters. To our knowledge, prior work on PAC-Bayesian risk certificate building with BNNs has relied exclusively on Blundell et al. [2015]'s *Bayes by backprop* approach [see, e.g., Pérez-Ortiz et al., 2021]. We show with Figure 4b that using the *local reparameterization trick (LRT)* [Kingma et al., 2015] to compute the empirical risk term in the bound calculation greatly improves bound tightness of all four evaluated bounds. This effects holds even in the already saturated expert-level policy of the challenging `Humanoid` environment. Further details can be found in Appendix D.

5 Limitations, future work, and broader impact

We restricted our empirical investigation to a single actor-critic algorithm and a single physics engine. This was a conscious choice to facilitate interpretation and maintain feasibility. Given the brittleness of the MuJoCo locomotion environments, we do not expect meaningful additional information to come from extending the same pipeline to RL suites with a similar level of fidelity. The next major step forward would be to implement our pipeline on a physical platform under controlled conditions. We considered only dense-reward locomotion scenarios with rigid locomotors, as this is the natural first step. The applicability of our findings to more advanced control settings, such as sparse-reward scenarios that require goal-conditioned or hierarchical RL algorithm design is subject to further investigation. We leave this enterprise to future work as the deep learning-based solutions for such setups have not yet reached the level of maturity to move beyond simulations. Another significant leap would be to proceed from our current self-certified policy evaluation approach to self-certified policy optimization in an online setting. This would necessitate training the policy via a PAC-Bayes bound. However, RL is a feedback-loop system in which assuring convergence, numerical stability, and optimal trade-offs between exploration and exploitation are major determinants of a stable training. While promising preliminary results exist [Tasdighi et al., 2024a,b], the problem is fundamental and requires a dedicated research program—an effort that goes beyond the scope of a single paper.

Our work contributes to the trustworthy development of agentic AI technologies, thereby promoting their adoption by society. Public concerns about such technologies will be even more pronounced when they are deployed on physical systems that are in direct contact with humans. Thanks to reliable risk certificates, such safety-critical technologies are likely to receive wider adoption. This, in turn, will further accelerate their development by expanding the pool of practice and observations.

References

S. Agrawal and R. Jia. Optimistic posterior sampling for reinforcement learning: worst-case regret bounds. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

P. Alquier et al. User-friendly introduction to PAC-Bayes bounds. *Foundations and Trends in Machine Learning*, 2024.

A. Ambroladze, E. Parrado-Hernandez, and J. Shawe-Taylor. Tighter PAC-Bayes bounds. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2006.

P. Auer and R. Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2006.

P. Auer, T. Jaksch, and R. Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2008.

M. G. Azar, I. Osband, and R. Munos. Minimax regret bounds for reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2017.

P. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

G. Brockman. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

O. Catoni. PAC-Bayesian supervised classification: The thermodynamics of statistical learning. *Lecture Notes-Monograph Series*, 2007.

X. Chen, C. Wang, Z. Zhou, and K. Ross. Randomized ensembled double q-learning: Learning fast without a model. *International Conference on Learning Representations (ICLR)*, 2021.

P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

C. Dann, T. Lattimore, and E. Brunskill. Unifying PAC and regret: Uniform PAC bounds for episodic reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

DeepSeek-AI et al. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

G. Dziugaite and D. Roy. Computing non-vacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

G. Dziugaite and D. Roy. Data-dependent PAC-Bayes priors via differential privacy. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

G. K. Dziugaite, K. Hsu, W. Gharbieh, G. Arpino, and D. Roy. On the role of data in PAC-Bayes bounds. In *Proceedings on the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

M. Fard, J. Pineau, and C. Szepesvári. PAC-Bayesian policy evaluation for reinforcement learning. In *Proceedings on the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

A. Foong, W. Bruinsma, D. Burt, and R. Turner. How tight can PAC-Bayes be in the small data regime? *Advances in Neural Information Processing Systems (NeurIPS)*, 34:4093–4105, 2021.

A. Y. Foong, W. P. Bruinsma, and D. R. Burt. A note on the chernoff bound for random variables in the unit interval. *arXiv preprint arXiv:2205.07880*, 2022.

S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

P. Germain, F. Bach, A. Lacoste, and S. Lacoste-Julien. PAC-Bayesian theory meets Bayesian inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018a.

T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse control tasks through world models. *Nature*, 2025.

K.-C. Hsu, A. Z. Ren, D. P. Nguyen, A. Majumdar, and J. F. Fisac. Sim-to-lab-to-real: Safe reinforcement learning with shielding and generalization guarantees. *Artificial Intelligence*, 2022.

N. Jiang. PAC reinforcement learning with an imperfect model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 2015.

I. Kostrikov, L. M. Smith, and S. Levine. Demonstrating a walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. In *Robotics: Science and Systems*, 2023.

A. Krishnamurthy, A. Agarwal, and J. Langford. PAC reinforcement learning with rich observations. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

J. Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research (JMLR)*, 6(3), 2005.

T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

S. Lotfi, M. Finzi, S. Kapoor, A. Potapczynski, M. Goldblum, and A. G. Wilson. PAC-Bayes compression bounds so tight that they can explain generalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

A. Maurer. A note on the PAC bayesian theorem. *arXiv preprint cs/0411099*, 2004.

S. D. Mbacke, F. Clerc, and P. Germain. PAC-Bayesian generalization bounds for adversarial generative models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023a.

S. D. Mbacke, F. Clerc, and P. Germain. Statistical guarantees for variational autoencoders using PAC-Bayesian theory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023b.

D. McAllester. PAC-Bayesian model averaging. In *Proceedings of the Conference on Learning Theory (COLT)*, 1999.

D. A. McAllester. Some PAC-Bayesian theorems. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 230–234, 1998.

Z. Mhammedi, P. Grünwald, and B. Guedj. PAC-Bayes un-expected bernstein inequality. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

K. Nozawa, P. Germain, and B. Guedj. PAC-Bayesian contrastive unsupervised representation learning. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2020.

I. Osband, D. Russo, and B. Van Roy. (more) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

M. Pérez-Ortiz, O. Rivasplata, J. Shawe-Taylor, and C. Szepesvári. Tighter risk certificates for neural networks. *Journal of Machine Learning Research (JMLR)*, 2021.

M. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

D. Reeb, A. Doerr, S. Gerwinn, and B. Rakitsch. Learning Gaussian processes by minimizing PAC-Bayesian generalization bounds. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

O. Rivasplata, V. M. Tankasali, and C. Szepesvari. PAC-Bayes with backprop. *arXiv preprint arXiv:1908.07380*, 2019.

M. Seeger. PAC-Bayesian generalisation error bounds for Gaussian process classification. *Journal of Machine Learning Research (JMLR)*, 2002.

W. Shang, K. Sohn, D. Almeida, and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

J. Shawe-Taylor and R. Williamson. A PAC analysis of Bayesian estimator. In *Proceedings of the Conference on Learning Theory (COLT)*, 1997.

A. Strehl, L. Li, E. Wiewiora, J. Langford, and M. Littman. PAC model-free reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2006.

A. L. Strehl, L. Li, and M. L. Littman. Reinforcement learning in finite mdps: PAC analysis. *Journal of Machine Learning Research (JMLR)*, 2009.

R. Sutton and A. Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

B. Tasdighi, A. Akgül, M. Haussmann, K. K. Brink, and M. Kandemir. PAC-Bayesian soft actor-critic learning. In *Advances in Approximate Bayesian Inference (AABI)*, 2024a.

B. Tasdighi, M. Haussmann, N. Werge, Y.-S. Wu, and M. Kandemir. Deep exploration with PAC-Bayes. *arXiv preprint arXiv:2402.03055*, 2024b.

N. Thiemann, C. Igel, O. Wintenberger, and Y. Seldin. A strongly quasiconvex PAC-Bayesian bound. In *International Conference on Algorithmic Learning Theory*, pages 466–492. PMLR, 2017.

D. Tiapkin, D. Belomestny, D. Calandriello, É. Moulines, R. Munos, A. Naumov, M. Rowland, M. Valko, and P. Ménard. Optimistic posterior sampling for reinforcement learning with few samples and tight guarantees. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022a.

D. Tiapkin, D. Belomestny, E. Moulines, A. Naumov, S. Samsonov, Y. Tang, M. Valko, and P. Ménard. From dirichlet to rubin: Optimistic exploration in rl without bonuses. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2022b.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Y.-S. Wu and Y. Seldin. Split-kl and PAC-Bayes-split-kl inequalities for ternary random variables. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Y.-S. Wu, Y. Zhang, B.-E. Chérief-Abdellatif, and Y. Seldin. Recursive PAC-Bayes: A frequentist approach to sequential prior updates with no information loss. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

W. Zhang, G. Wang, J. Sun, Y. Yuan, and G. Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences. 2019.

APPENDIX

A Related work

Theoretical analysis of reinforcement learning. Exploration in finite-horizon Markov decision processes has been addressed in several prior works. UCBVI [Azar et al., 2017] offers a simple approach based on optimism under uncertainty, combining computational efficiency with conceptual clarity for exploration in finite-horizon MDPs. BayesUCBVI [Tiapkin et al., 2022b] builds on this by introducing a tabular, stage-dependent, episodic reinforcement learning algorithm that uses a quantile of the posterior distribution of Q-values for optimism, eliminating the need for explicit bonus terms. However, it remains uncertain whether the PSRL [Osband et al., 2013] approach can achieve optimal problem-independent regret bounds. UCRL2 [Auer et al., 2008], an extension of UCRL [Auer and Ortner, 2006], employs upper confidence bounds (UCB) to guide policy selection, encouraging exploration of uncertain state-action pairs and reducing regret over time, resulting in a near-optimal regret bound that grows logarithmically with the number of episodes. Agrawal and Jia [2017] propose a PSRL-inspired algorithm for multi-armed bandits, using posterior sampling to incorporate uncertainty and to form an optimistic policy, achieving near-optimal regret bounds. However, it is unclear whether it can achieve the problem-independent lower bound. OPSRL [Tiapkin et al., 2022a] extends PSRL by using multiple posterior samples instead of a single one, providing a high-probability regret-bound guarantees and ensuring strong performance.

PAC-Bayes analysis. PAC-Bayesian analysis provides a frequentist framework for integrating prior knowledge into learning algorithms [Shawe-Taylor and Williamson, 1997, McAllester, 1998, Alquier et al., 2024]. These methods leverage priors that sustain high confidence during learning [McAllester, 1999, Seeger, 2002, Catoni, 2007, Thiemann et al., 2017, Rivasplata et al., 2019, Pérez-Ortiz et al., 2021], but the resulting confidence intervals rely on data excluded from prior formation, making prior data allocation a critical challenge. Strategies for crafting effective PAC-Bayesian priors address this limitation [Ambroladze et al., 2006, Dziugaite et al., 2021, Wu et al., 2024].

PAC-Bayesian risk certificate. In recent years, PAC-Bayesian methods have also been used to compress neural nets [Lotfi et al., 2022], and to provide risk certificates for both uninformed [Dziugaite and Roy, 2017] and data-informed priors [Dziugaite and Roy, 2018, Dziugaite et al., 2021, Pérez-Ortiz et al., 2021]. Reeb et al. [2018] introduce PAC-Bayesian bounds for Gaussian process-based regression, and an increasing body of literature now provides risk certificates for deep generative models, such as VAEs [Mbacke et al., 2023b], GANs [Mbacke et al., 2023a], and contrastive learning frameworks [Nozawa et al., 2020], among others. Hsu et al. [2022] used PAC-Bayes to provide realistic safety guarantees, focusing primarily on worst-risk and out-of-distribution safety.

PAC-Bayesian RL. PAC analysis in RL has led to algorithms that provide formal guarantees on sample complexity and performance. Strehl et al. [2006] introduced Delayed Q-learning, an efficient model-free RL method. Strehl et al. [2009] established PAC bounds for both model-free and model-based methods in finite MDPs, highlighting sample efficiency. For complex observation spaces, Krishnamurthy et al. [2016] proposed Least Squares Value Elimination, extending contextual bandit frameworks to sequential settings with PAC guarantees. In model-based RL, Dann et al. [2017] introduced the Uniform-PAC framework for finite-state episodic MDPs, achieving optimal sample complexity and regret bounds. Jiang [2018] improved model-based efficiency with a polynomial-complexity algorithm. PAC-Bayesian frameworks have also been applied in RL for policy evaluation and stability. Fard et al. [2012] introduced a PAC-Bayesian method for policy evaluation with probabilistic guarantees which was subsequently extended to soft actor-critic approaches [Tasdighi et al., 2024a] and deep exploration with sparse rewards [Tasdighi et al., 2024b]. Despite these advancements, the tightness, i.e., the quality, of the risk certificates remained an open question.

B Algorithms

In this section, we present two pseudocodes. Algorithm 1 illustrates the overall pipeline of our method, capturing the main components from policy training to PAC-Bayes bound calculation.

Algorithm 2 indicates the Recursive PAC-Bayes framework in detail, which outlines the step-by-step calculation of recursive bound across different splits of the training data, ultimately leading to the final bound. This framework supports multiple levels of recursion (e.g., two or six levels in our experiments). Details on the size of each data split and the selection strategy are provided in Appendix C.

Algorithm 1 Predicting Policy Return via PAC-Bayes Bound Fitting

Input: Environment, Actor-Critic algorithm, BNN architecture, PAC-Bayes bounds
Output: Predicted test-time discounted returns and PAC-Bayes generalization bounds

Policy Training
Train policy π using REDQ Actor-Critic algorithm
Save policy parameters after specific training steps

Data Collection
Disable further learning
Run saved policy for few roll-outs and collect data
Split data into training and test sets

Bayesian Model Training
Initialize Bayesian Neural Network (BNN)
Train Bayesian model on training data using PAC-Bayes-inspired loss as training objective

Bound Construction and Evaluation
for each bound type $\in$\{NR-NI, NR-I, R-I T=2, R-I T=6\} **do**
    Specify prior and posterior distributions
    Compute PAC-Bayes bound using model predicted outputs
end for

Return: Model predictions on test data and bound evaluations with train data

Algorithm 2 Recursive PAC-Bayes bound loss computation

Input: Training dataset split $\mathcal{D} = S_1 \cup \cdots \cup S_T$ with total size $N = |\mathcal{D}|$, where $N_t = |S_{\geq t}|$; scaling factors $\kappa_1, \ldots, \kappa_T$; posterior parameters $\{\rho_1, \ldots, \rho_T\}$; and fixed confidence levels δ, δ'.

Calculate the loss for first portion: $\hat{L}(h) \in [0, B]$

$$\mathbb{E}_{\rho_1}[\hat{L}^+(h)] \leq \mathrm{kl}^{-1,+}\left(\frac{1}{N_t}\sum\left(\max\left\{0, \hat{L}(h) - \mu\right\}\right), \frac{\ln(T/\delta')}{N_t}\right)$$

Specify the first portion of recursive bound:

$$\mathcal{B}_1(\rho_1) = B\mathrm{kl}^{-1,+}\left(\frac{\mathbb{E}_{\rho_1}[\hat{L}^+(h)]}{B}, \frac{\mathrm{KL}\left(\rho_1 \parallel \rho_0^*\right) + \ln(2T\sqrt{N}/\delta)}{N}\right),$$

for each of the portions from $t = 2$ to End: **do**

Calculate the excess loss:

$$L_t^{\mathrm{exc}}(h) = L(h) - \kappa_t \mathbb{E}_{\rho_{t-1}}\left[L(h')\right] \in [-\kappa_t B, B]$$

we have:

$$\mathbb{E}_{\rho_t}[\hat{L}_t^{\mathrm{exc}+}(h)] \leq \mathrm{kl}^{-1,+}\left(\frac{1}{N_t}\sum\left(\max\left\{0, \hat{L}_t^{\mathrm{exc}}(h) - \mu\right\}\right), \frac{\ln(2T/\delta')}{N_t}\right)$$

$$\mathbb{E}_{\rho_t}[\hat{L}_t^{\mathrm{exc}-}(h)] \leq \mathrm{kl}^{-1,-}\left(\frac{1}{N_t}\sum\left(\max\left\{0, \mu - \hat{L}_t^{\mathrm{exc}}(h)\right\}\right), \frac{\ln(2T/\delta')}{N_t}\right)$$

calculate $\mathcal{E}_t(\rho_t)$:

$$\begin{aligned}\mathcal{E}_t(\rho_t) = \mu &+ (B-\mu)\mathrm{kl}^{-1,+}\left(\frac{\mathbb{E}_{\rho_t}[\hat{L}_t^{\mathrm{exc}+}(h)]}{B-\mu}, \frac{\mathrm{KL}\left(\rho_t \parallel \rho_{t-1}^*\right) + \ln(4T\sqrt{N_t}/\delta)}{N_t}\right) \\ &- (\mu + \kappa_t B)\mathrm{kl}^{-1,-}\left(\frac{\mathbb{E}_{\rho_t}[\hat{L}_t^{\mathrm{exc}-}(h)]}{\mu + \kappa_t B}, \frac{\mathrm{KL}\left(\rho_t \parallel \rho_{t-1}^*\right) + \ln(4T\sqrt{N_t}/\delta)}{N_t}\right),\end{aligned}$$

Update the whole bound iteratively:

$$\mathcal{B}_t(\rho_t) = \mathcal{E}_t(\rho_t, \kappa_t) + \kappa_t \mathcal{B}_{t-1}(\rho_{t-1}^*),$$

$t = t + 1$

end for

With probability higher than 1-δ:

$$\mathbb{E}_{\rho_t}\left[L(h)\right] \leq \mathcal{B}_t(\rho_t)$$

C Experimental Details

In this section, we provide the details and design choices for our environments, dataset, model training and how to replicate our experimental results. We consider five environments from version 'v4' of MuJoCo suite environments [Todorov et al., 2012]. All implementations in this work utilize the PyTorch framework [Brockman, 2016] version '2.5.1' and the OpenAI Gym environment [Paszke et al., 2019] with version '0.29.0' .

To assess the effectiveness of our proposed approach, we consider five popular locomotor tasks with complex state and action dimensionalities: `Ant`, `Half-cheetah`, `Hopper`, `Walker2d`, `Humanoid`.

C.1 Specific hyperparameters

For recursive PAC-Bayes we use κ_t = 1/2 for all t, and study the impact of recursion for data split over different portions. In our experiments we use δ equal to 0.025 following Wu et al. [2024]'s implementation `https://github.com/pyijiezhang/rpb`.

In addition to the main confidence parameter δ that controls the overall probability with which the PAC-Bayes bound holds, we introduce an auxiliary confidence parameter δ', following the approach of Wu et al. [2024]. This parameter accounts for the approximation error introduced when estimating expected values of empirical loss and empirical excess loss via sampling [Pérez-Ortiz et al., 2021], since these quantities lack closed-form expressions in our setting. Specifically, δ' ensures that these sample-based estimates are accurate with high probability. We set $\delta' = 0.01$, consistent with the choice in the referenced work, and apply a union bound to combine the confidence levels. As a result, the overall bound holds with probability at least $1 - \delta - \delta'$, covering both the generalization guarantee and the estimation accuracy. This consideration is applied only during the evaluation phase for estimating the bounds, and not the optimization process.

C.2 Constructing validation and test datasets

Our methodology consists of two main phases: *training* and *evaluation*. To collect the transition data required for constructing validation and test datasets, we train a generic reinforcement learning agent, REDQ—a powerful off-policy actor-critic method [Chen et al., 2021]. The REDQ agent uses an ensemble of ten critic networks and a single actor network. We use a replay ratio of one and perform one gradient update per environment step. At each Bellman target computation, two critics are randomly sampled from the ensemble for min-clipping, following the original work.

Training is conducted for 300 000 environment steps, after which we save the resulting policy as the *expert policy*. We also snapshot the policy at 100 000 and 200 000 steps, referring to them as the *starter* and *intermediate* level policies, respectively, to enable analysis across varying levels of agent proficiency.

In the evaluation phase, we disable the learning process and run the agent with exploration turned off and no policy updates performed. Using the frozen policy (at each of the three expertise levels), we collect 100 episodes of state transitions and rewards to construct a validation dataset for fitting the PAC-Bayes bound. An additional 100 episodes are collected under the same conditions to form our test dataset. This test data is used to compute a proxy for the generalization performance by evaluating the predicted discounted returns of the frozen policy on previously unseen trajectories.

C.3 Neural network architectures

REDQ agent training. We use an ensemble of ten critic networks along with a single actor network. Each neural network consists of three layers, with layer normalization after each layer to regularize the network, following the approach of Ball et al. [2023]. Additionally, we use the concatenated ReLU activation function [Shang et al., 2016] which combines the positive and negative parts of two ReLU activations and concatenates them. This leads to richer feature representations and enables the learning of more complex patterns.

During training, we employ 10 000 warm-up steps without policy updates where the agent only interacts with the environment, during the evaluation phase we don't have any warm-up steps. We use a replay ratio (RR) of one in both training and evaluation phase, which provides training stability and sample efficiency. We employ the Huber loss in the critic to measure the discrepancy between predicted Q-values and target Q-values as it consistently provides performance advantages across various baselines. Furthermore, the temperature parameter α is automatically tuned during training to regulate policy entropy, balancing exploration and exploitation, following the method introduced by Haarnoja et al. [2018b].

All these architectural and training choices were empirically found to improve the model's overall performance. Table 1 summarizes the hyperparameter and network configurations used in our experiments.

Bayesian model training. We aim to predict the test-time discounted return of a policy π by fitting a PAC-Bayes generalization bound using training data. To this end, we use a Bayesian neural network (BNN), which allows us to capture model uncertainty and compute valid generalization guarantees.

The BNN is a feedforward neural network with two hidden layers and one output layer. Each layer is implemented as a Variational Bayesian Linear layer, where weights are modeled as independent Gaussian distributions with learnable means and variances. ReLU activations are used between layers to introduce non-linearity.

During training, we apply the local reparameterization trick [Kingma et al., 2015], which samples the layer outputs rather than the weights. This reduces the variance of the gradient estimates, improves training stability, and reduces computational cost.

We train the BNN using a PAC-Bayes-inspired objective based on McAllester's bound [McAllester, 1998], which balances empirical loss and a KL divergence term. The KL term acts as a regularizer that penalizes deviation from a prior distribution, effectively controlling model complexity. This enables us to compute reliable generalization bounds and estimate the expected test-time return of the policy. Architectural details are summarized in Table 1.

C.4 Baselines details

We evaluate two non-recursive (NR) baselines and two recursive (R) variants with different levels of depth. For the uninformed baseline (NR-NI), we use a PAC-Bayes-kl inverse bound where the prior is initialized without training data dependency . The posterior is learned using the entire training dataset, and the bound is also evaluated on this full dataset.

For the informed case (NR-I), we split the training dataset in half: the first half is used to learn a data-informed prior, and the second half is used to learn the posterior, both steps minimizing a PAC-Bayes bound. The bound is evaluated using the same subset used to learn the posterior.

In the recursive settings, we apply these splits iteratively. For example, with recursion depth two, we divide the data in two 50, 50 splits. For the deeper recursion with six steps, we split the training set into increasingly smaller partitions based on the number of episodes (3, 4, 6, 12, 25, 50). This allows us to allocate fewer data points for prior learning early on while reserving more for later refinement, enabling progressively more targeted fine-tuning.

Like Wu et al. [2024] we apply a relaxation of the PAC-Bayes-kl bound in our experiments by optimizing using the MCAllester bound [McAllester, 1998] while the bound evaluation is done using the split kl-based bound. In all approaches, we utilize a set of factorized Gaussian distributions to represent the priors and posteriors associated with every trainable parameter within the classifiers. These distributions take the form $\pi = \mathcal{N}(w, \sigma I)$, where $w \in \mathbb{R}^d$ is the mean vector, and σ denotes the scalar variance parameter. To start, an uninformed prior $\pi_0 = \mathcal{N}(w_0, \sigma_0 I)$ is employed, which does not depend on the training data. Here, the mean parameter is randomly initialized using Kaiming uniform initialization, while the log-variance parameter is initialized to a fixed value of (-4.6) . We train all models using the hyperparameters provided in Table 1, specified with header 'PAC-Bayes bound fitting'.

D Results

The main results for all five MujoCo environments are provided below. We considered three level of policy quality and five policy instances per environment. The scores represent the normalized bound and loss values with respect to the maximum discounted return for each environment, where the results are aggregated over five repetitions. Normalization applied using the maximum observed discounted return over all policy instances. Table 2 to Table 16 summarize the normalized bound values along with train and test error, across various environment and policy qualities.

Mitigate Sample Correlation. To reduce overfitting caused by high correlation among consecutive samples—where states are very similar and their associated target values nearly identical—we apply a thinning strategy. This strategy selectively samples fewer data points to break the correlation. Specifically, in environments like `Ant` and `Half-Cheetah`, which tend to produce longer episodes, we retain every fifth sample. In contrast, for environments with shorter episodes such as `Humanoid`, `Hopper`, and `Walker2d`, we keep every third sample. This approach reduces redundancy while ensuring that the dataset still contains sufficient information for training.

Tightness of the bounds. Figure 5 shows the tightness of all baselines, measured as the difference between the normalized bound score and the normalized test error. Values are aggregated across policy instances and repetitions for each of the five environments, with smaller values indicating tighter bounds. Box plots display the distribution of these differences: the box covers the interquartile range (25th to 75th percentile), the median is marked inside, and whiskers extend to points within 1.5

Table 1: *Hyper-parameters used in our experiments.* This includes both hyperparameters applied Actor-Critic setup in agent training and Bayesian model training for PAC-Bayes bound fitting.

Agent training:	
Evaluation episodes (evaluation mode)	1
Evaluation frequency (evaluation mode)	1
Evaluation episodes (training mode)	-
Evaluation frequency (training mode)	-
Discount factor (γ)	0.99
n-step returns	1 step
Replay ratio	1
Number of critic networks	10
Replay buffer size	100,000
Maximum timesteps*	300,000
Number of hidden layers for all networks	2
Number of hidden units per layer	256
Nonlinearity	CReLU
Mini-batch size	256
Network regularization method	Layer Normalization (LN) [Ball et al., 2023]
Actor/critic optimizer	Adam [Kingma and Ba, 2015]
Optimizer learning rates (η_ϕ, η_θ)	3e-4
Polyak averaging parameter (τ)	5e-3
PAC-Bayes bound fitting:	
Number of hidden layers	2
Number of hidden units per layer	256
Nonlinearity	ReLU
Network regularization method	Layer Normalization (LN) [Ball et al., 2023]
BNN optimizer	Adam [Kingma and Ba, 2015]
Optimizer learning rate	2e-2
Gradient Clipping Type	max-norm
Gradient Clipping Threshold	1
Learning rate scheduler	StepLR
Learning rate decay factor	0.5
scheduler step size (epochs)	10

times the interquartile range, illustrating typical variability. Outliers beyond this range are omitted for clarity. This effectively captures both the central tendency and the spread of the data across policies and methods. The plot reveals that using data-informed priors generally leads to tighter bounds, although this effect is less pronounced at the Starter level. Moreover, applying recursion and increasing its depth further tightens the bounds. The improvements from recursion are more noticeable in challenging environments and less so in simpler tasks.

Validation data size. Figure 6 examines the Pearson correlation between normalized bound scores and normalized test errors across policy levels and varying validation data sizes for the Humanoid environment—our most challenging setting due to its high-dimensional state and action spaces. Each heatmap displays a point estimate of these correlations for all baselines across different validation set sizes for specific policy aggregated over policy instances and repetitions. Although correlations are generally weaker and more variable under the Starter policy, for intermediate and expert level policies they tend to improve with stronger policies and larger validation sets. Recursive bounds, especially those with greater recursion depth, consistently show stronger positive correlations. These findings emphasize that both policy strength and validation data size affect the bound's ability to predict generalization, with recursion providing the most reliable alignment between bounds and test errors.

Effect of Local reparameterization trick We incorporate the Local Reparameterization Trick (LRT) into our variational Bayesian neural network architecture, where it has not been used in previous

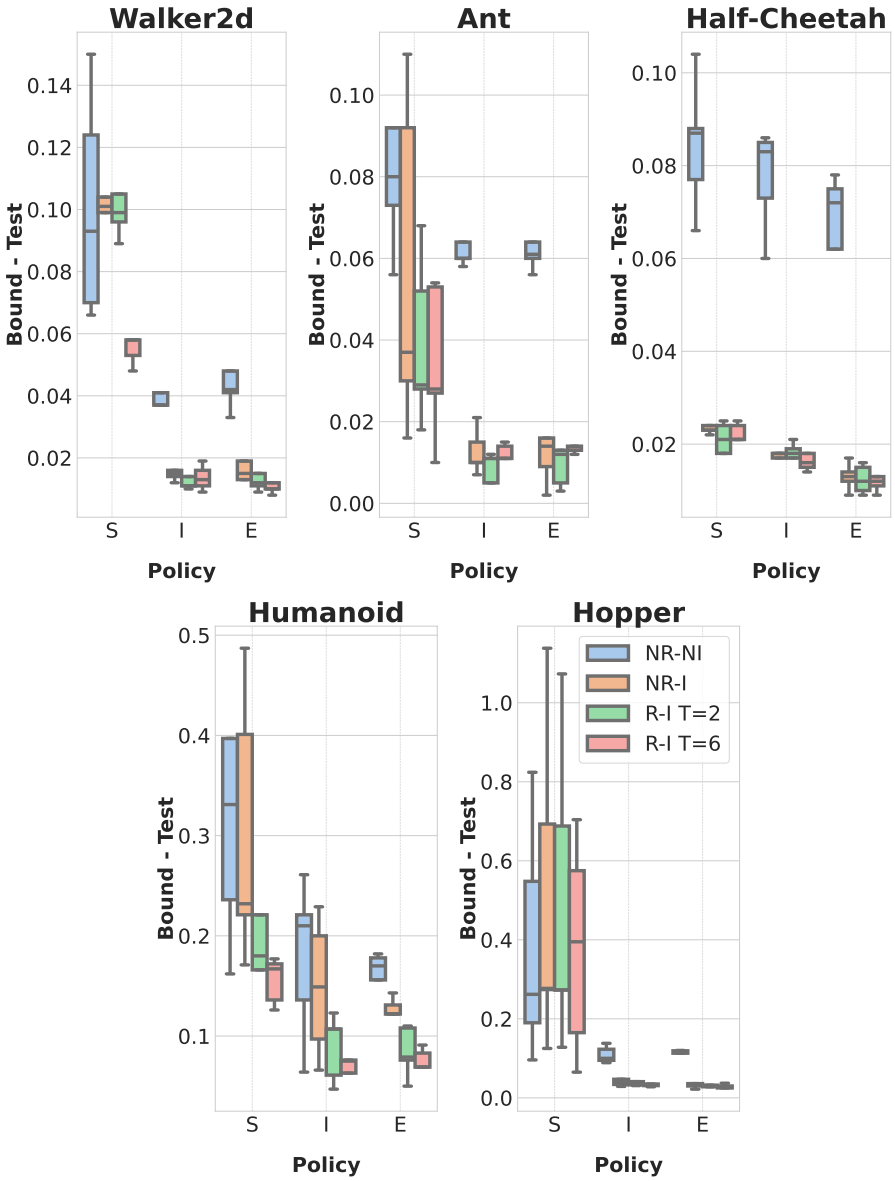


Figure 5: *(Bound - test) values.* Tightness of the bound over all baselines an dpolicy levels considered for five MuJoCo environments, the plots aggregated over policy instances and their repetitions.

Bayesian approaches for reinforcement learning to our knowledge. Unlike standard sampling methods that draw one set of weights per layer and propagate forward, LRT enables sampling at the level of individual pre-activations, significantly reducing the variance of gradient estimates during training. This leads to more stable optimization and improved posterior quality. To evaluate its impact, we compare the tightness of all our baselines on the Humanoid environment for two versions of our model, with and without LRT. As shown in Figure 4b, applying LRT consistently results in tighter bounds. Results are aggregated over five policy instances and five repetition each, while only the expert level policy. In both cases the results are a clear improvement. These expert policy improvements suggest improvements in other policy levels as well.

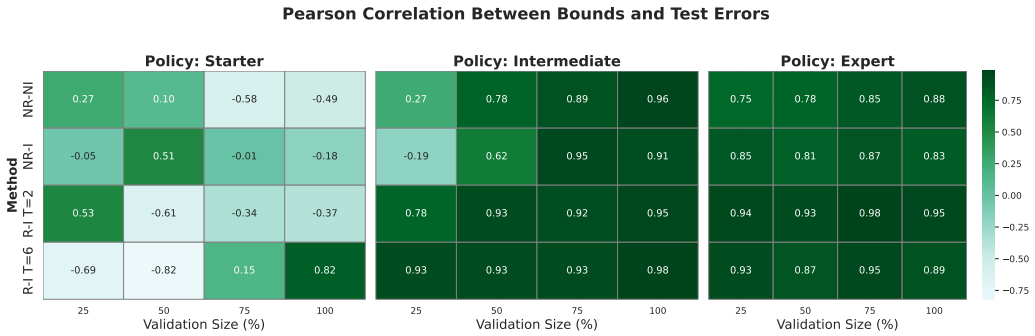


Figure 6: Pearson correlation between bound scores and test error considering five policy instances and five repetitions for each, across various validation sizes and policy levels.

Table 2: Normalized test error, train error and bound comparison for **Ant** considering **Starter** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.074	0.070	0.143		0.061	0.067	0.147		0.112	0.085	0.177		0.070	0.068	0.124		0.195	0.156	0.281	
NR-I	0.051	0.043	0.080		0.052	0.057	0.073		0.137	0.106	0.216		0.066	0.064	0.094		0.268	0.217	0.309	
R-I T=2	0.060	0.053	0.123	0.082	0.056	0.060	0.122	0.078	0.198	0.160	0.259	0.228	0.072	0.071	0.127	0.099	0.283	0.230	0.348	0.282
R-I T=6	**0.049**	**0.040**	0.118	**0.068**	**0.048**	**0.055**	0.099	**0.065**	**0.078**	**0.061**	0.149	**0.115**	**0.063**	**0.061**	0.116	**0.088**	**0.103**	**0.086**	0.183	**0.139**

Table 3: Normalized test error, train error and bound comparison for **Ant** considering **Intermediate** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.057	0.055	0.129		0.070	0.089	0.149		0.046	0.056	0.116		0.046	0.049	0.113		0.055	0.065	0.123	
NR-I	0.053	0.051	0.066		0.069	0.086	0.107		0.051	0.059	0.069		0.036	0.042	0.052		0.038	0.047	0.054	
R-I T=2	0.044	0.042	0.093	0.053	0.065	0.081	0.138	0.092	0.043	0.050	0.093	0.055	0.038	0.043	0.093	0.055	**0.034**	0.044	0.087	**0.049**
R-I T=6	**0.039**	**0.038**	0.068	**0.052**	**0.046**	**0.049**	0.080	**0.064**	**0.039**	**0.041**	0.068	**0.052**	**0.035**	**0.040**	0.074	**0.051**	0.041	**0.043**	0.070	0.054

Table 4: Normalized test error, train error and bound comparison for **Ant** considering **Expert** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.039	0.040	0.101		0.053	0.057	0.117		0.042	0.050	0.106		0.058	0.056	0.135		0.052	0.052	0.116	
NR-I	0.041	0.043	0.057		0.046	0.050	0.059		**0.039**	0.047	**0.049**		0.053	0.050	0.081		**0.037**	**0.036**	0.052	
R-I T=2	**0.035**	**0.037**	0.087	**0.050**	**0.041**	0.045	0.096	**0.050**	**0.039**	0.047	0.089	0.050	0.069	0.068	0.133	0.094	0.038	0.037	0.087	**0.049**
R-I T=6	0.040	0.040	0.073	0.053	0.043	**0.044**	0.075	0.057	0.044	**0.043**	0.075	0.057	**0.040**	**0.038**	0.078	**0.059**	0.040	0.040	0.068	0.052

Table 5: Normalized test error, train error and bound comparison for **Half-cheetah** considering **Starter** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.071	0.072	0.176		0.058	0.058	0.135		0.062	0.062	0.128		0.066	0.065	0.153		0.060	0.061	0.148	
NR-I	0.043	0.043	**0.066**		0.039	0.039	0.063		0.037	0.037	0.059		**0.047**	**0.047**	0.070		0.042	0.042	0.066	
R-I T=2	**0.042**	**0.042**	0.126	0.067	0.044	0.044	0.106	0.068	0.037	0.037	0.094	0.055	0.051	0.050	0.133	**0.068**	**0.039**	**0.039**	0.108	**0.060**
R-I T=6	0.056	0.057	0.116	0.081	**0.037**	**0.038**	0.086	**0.059**	**0.032**	**0.032**	0.085	**0.053**	0.050	0.049	0.106	0.074	0.041	0.041	0.100	0.062

Table 6: Normalized test error, train error and bound comparison for **Half-cheetah** considering **Intermediate** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.066	0.066	0.149		0.068	0.067	0.152		0.033	0.033	0.093		0.044	0.044	0.117		0.062	0.062	0.148	
NR-I	**0.042**	**0.043**	**0.060**		0.045	0.045	0.063		0.030	0.030	0.044		0.040	0.040	0.058		**0.043**	**0.043**	0.063	
R-I T=2	0.047	0.047	0.115	0.065	**0.041**	**0.041**	0.113	**0.058**	**0.029**	**0.029**	0.076	**0.041**	**0.039**	**0.039**	0.114	0.058	0.046	0.046	0.120	0.067
R-I T=6	0.046	0.046	0.085	0.064	0.046	0.046	0.085	0.064	0.032	0.032	0.064	0.046	**0.039**	**0.039**	0.073	**0.054**	0.046	0.046	0.083	**0.062**

Table 7: Normalized test error, train error and bound comparison for **Half-cheetah** considering **Expert** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.045	0.045	0.107		0.050	0.052	0.124		0.029	0.029	0.070		0.063	0.058	0.136		0.061	0.065	0.140	
NR-I	**0.033**	**0.033**	**0.046**		**0.043**	**0.046**	0.063		**0.018**	**0.018**	**0.027**		**0.041**	**0.036**	**0.050**		0.045	0.049	0.061	
R-I T=2	0.040	0.040	0.093	0.056	**0.043**	**0.046**	0.102	**0.058**	**0.018**	**0.018**	0.050	**0.027**	0.044	0.038	0.096	0.053	**0.041**	**0.045**	0.098	**0.055**
R-I T=6	0.037	0.037	0.064	0.048	0.045	**0.046**	0.076	0.059	0.021	0.021	0.040	0.030	0.042	0.042	0.071	0.055	0.046	0.047	0.078	0.059

Table 8: Normalized test error, train error and bound comparison for **Hopper** considering **Starter** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.567	0.569	0.759		0.341	0.320	0.868		0.007	0.007	**0.269**		0.201	0.187	1.011		0.122	0.122	0.218	
NR-I	0.575	0.577	0.854		0.347	0.326	1.019		0.009	0.009	0.283		0.200	0.186	1.324		0.189	0.189	0.314	
R-I T=2	0.580	0.583	0.909	0.856	0.346	0.326	1.593	1.014	0.009	0.009	0.896	0.282	0.200	0.187	1.570	1.260	0.261	0.261	0.398	0.389
R-I T=6	**0.156**	**0.156**	0.528	**0.321**	**0.243**	**0.228**	2.148	**0.803**	**0.004**	**0.004**	1.629	0.399	**0.166**	**0.155**	2.850	**0.859**	**0.058**	**0.058**	0.215	**0.123**

Table 9: Normalized test error, train error and bound comparison for **Hopper** considering **intermediate** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.079	0.081	0.181		0.075	0.076	0.165		0.073	0.074	0.197		0.096	0.095	0.233		0.060	0.060	0.155	
NR-I	0.069	0.070	0.109		0.059	0.058	0.087		0.070	0.070	0.117		0.115	0.116	0.260		0.062	0.062	0.096	
R-I T=2	0.073	0.074	0.167	0.111	0.062	0.062	0.138	0.093	0.069	0.069	0.192	0.110	0.234	0.233	0.337	0.348	0.061	0.061	0.151	0.094
R-I T=6	**0.064**	**0.063**	0.138	**0.094**	**0.054**	**0.053**	0.113	**0.081**	**0.061**	**0.061**	0.149	**0.096**	**0.071**	**0.071**	0.171	**0.115**	**0.059**	**0.059**	0.134	**0.092**

Table 10: Normalized test error, train error and bound comparison for **Hopper** considering **Expert** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.072	0.071	0.184		0.121	0.121	0.241		0.053	0.053	0.100		0.095	0.088	0.207		0.094	0.092	0.210	
NR-I	0.058	0.057	0.089		**0.060**	**0.061**	0.091		0.046	0.046	0.068		0.106	0.098	0.244		0.060	0.059	0.095	
R-I T=2	0.058	0.058	0.162	0.090	0.062	0.062	0.147	0.090	0.044	0.044	0.090	0.060	0.196	0.181	0.288	0.305	0.066	0.065	0.157	0.096
R-I T=6	**0.056**	**0.056**	0.129	**0.086**	0.061	**0.061**	0.119	**0.086**	**0.040**	**0.041**	0.069	**0.054**	**0.058**	**0.058**	0.139	**0.095**	**0.057**	**0.056**	0.123	**0.084**

Table 11: Normalized test error, train error and bound comparison for **Humanoid** considering **Starter** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.159	0.157	0.554		0.125	0.129	0.291		0.052	0.053	0.954		0.171	0.169	0.405		0.176	0.171	0.502	
NR-I	0.143	0.142	0.543		0.108	0.108	0.279		0.044	0.043	0.530		0.160	0.161	0.393		0.167	0.161	0.382	
R-I T=2	0.147	0.144	0.598	0.365	0.125	0.126	0.444	0.292	0.051	0.053	0.799	0.428	0.166	0.166	0.453	0.332	0.172	0.170	0.466	0.350
R-I T=6	**0.115**	**0.112**	0.563	**0.289**	**0.092**	**0.090**	0.401	**0.216**	**0.018**	**0.019**	0.931	**0.191**	**0.117**	**0.119**	0.558	**0.255**	**0.142**	**0.139**	0.643	**0.306**

Table 12: Normalized test error, train error and bound comparison for **Humanoid** considering **Intermediate** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.151	0.157	0.418		0.103	0.105	0.241		0.066	0.066	0.130		0.110	0.112	0.322		0.116	0.122	0.343	
NR-I	0.133	0.137	0.366		0.101	0.104	0.201		0.064	0.065	0.131		0.110	0.111	0.260		0.106	0.110	0.310	
R-I T=2	0.150	0.153	0.343	0.276	0.107	0.110	0.212	0.171	0.065	0.066	0.142	0.113	0.115	0.117	0.319	0.224	0.117	0.119	0.286	0.226
R-I T=6	**0.117**	**0.120**	0.317	**0.227**	**0.080**	**0.084**	0.225	**0.147**	**0.046**	**0.049**	0.185	**0.092**	**0.094**	**0.096**	0.283	**0.172**	**0.082**	**0.086**	0.243	**0.161**

Table 13: Normalized test error, train error and bound comparison for **Humanoid** considering **Expert** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	**0.095**	**0.098**	0.268		0.073	0.071	0.177		0.081	0.088	0.244		**0.082**	**0.083**	0.261		0.106	0.102	0.284	
NR-I	0.104	0.107	0.229		**0.067**	**0.066**	0.153		**0.062**	**0.066**	0.188		0.085	0.086	0.217		**0.101**	**0.098**	0.241	
R-I T=2	0.109	0.114	0.285	0.193	0.072	0.071	0.161	0.121	0.074	0.077	0.216	0.153	0.115	0.115	0.305	0.223	0.135	0.128	0.323	0.238
R-I T=6	0.102	0.107	0.269	**0.176**	**0.067**	0.067	0.163	**0.110**	0.076	0.078	0.229	**0.147**	0.091	0.092	0.270	**0.175**	**0.101**	0.099	0.288	**0.190**

Table 14: Normalized test error, train error and bound comparison for **Walker2d** considering **Starter** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.150	0.142	0.292		0.103	0.103	0.173		0.182	0.186	0.252		0.158	0.161	0.254		0.097	0.097	0.221	
NR-I	0.283	0.272	0.416		0.104	0.104	0.208		0.183	0.187	0.286		0.236	0.240	0.341		0.096	0.095	0.194	
R-I T=2	0.360	0.347	0.491	0.475	0.105	0.105	0.233	0.210	0.189	0.193	0.310	0.292	0.269	0.273	0.383	0.369	0.158	0.156	0.290	0.245
R-I T=6	**0.063**	**0.059**	0.191	**0.117**	**0.031**	**0.030**	0.217	**0.101**	**0.045**	**0.047**	0.176	**0.105**	**0.059**	**0.061**	0.177	**0.114**	**0.060**	**0.060**	0.183	**0.108**

Table 15: Normalized test error, train error and bound comparison for **Walker2d** considering **Intermediate** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.046	0.046	0.083		0.056	0.056	0.086		0.046	0.046	0.083		0.054	0.054	0.095		0.071	0.069	0.134	
NR-I	0.034	0.034	0.046		0.042	0.042	0.057		0.043	0.043	0.059		**0.038**	**0.038**	**0.052**		0.073	0.072	0.098	
R-I T=2	0.037	0.037	0.066	0.048	0.032	0.032	0.059	**0.042**	**0.033**	**0.033**	0.069	**0.044**	0.042	0.042	0.080	0.056	0.063	0.061	0.116	0.084
R-I T=6	**0.031**	**0.031**	0.051	**0.040**	**0.031**	**0.031**	0.053	**0.042**	0.040	0.041	0.073	0.057	0.039	0.039	0.065	**0.052**	**0.054**	**0.053**	0.093	**0.072**

Table 16: Normalized test error, train error and bound comparison for **Walker2d** considering **Expert** policy.

	E-SEED-01				E-SEED-02				E-SEED-03				E-SEED-04				E-SEED-05			
Method	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion	Train	Test	Bound	Last Portion
NR-NI	0.088	0.086	0.181		0.039	0.040	0.073		0.048	0.048	0.090		0.051	0.051	0.092		0.071	0.071	0.119	
NR-I	0.090	0.091	0.130		0.045	0.047	0.060		**0.039**	**0.039**	0.052		0.056	0.056	0.075		0.046	0.046	0.061	
R-I T=2	0.084	0.084	0.157	0.114	0.039	0.040	0.070	0.049	0.040	0.040	0.075	0.051	0.043	0.042	0.076	0.054	0.064	0.064	0.107	0.079
R-I T=6	**0.076**	**0.072**	0.127	**0.106**	**0.034**	**0.035**	0.054	**0.043**	0.040	0.040	0.063	**0.050**	**0.041**	**0.041**	0.068	**0.053**	**0.043**	**0.043**	0.068	**0.053**