A Unified Scaling Law for Bootstrapped DQNs

Roman Knyazhitskiy[1]
Supervisor(s): Neil Yorke-Smith[1]**, Pascal van der Vaart**[1]
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfillment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2025

Name of the student: Roman Knyazhitskiy
Final project course: CSE3000 Research Project
Thesis committee: Neil Yorke-Smith, Pascal van der Vaart, Matthijs Spaan

Abstract

We present a large-scale empirical study of Bootstrapped DQN (BDQN) and Randomized-Prior BDQN (RP-BDQN) in the DeepSea environment, aimed at characterizing their scaling properties. Our primary contribution is a unified scaling law that accurately models the probability of reward discovery as a function of task hardness and ensemble size. This law is parameterized by a method-dependent effectiveness factor, ψ. Under this framework, RP-BDQN demonstrates substantially higher effectiveness ($\psi \approx 0.87$) compared to BDQN ($\psi \approx 0.80$), enabling it to solve more challenging tasks. Our analysis reveals that this advantage stems from RP-BDQN's sustained ensemble diversity, which mitigates the posterior collapse observed in BDQN. Furthermore, we demonstrate diminishing returns in performance for ensemble sizes $K > 10$. These results offer practical guidance for ensemble configuration and raise new theoretical questions surrounding the effectiveness parameter ψ.

1 Introduction

In many areas of machine learning, algorithms learn from what are known as 'dense' rewards. This typically means that for every input the algorithm processes, there is a clearly defined desired output, or a straightforward way to calculate a 'goodness' score for the algorithm's prediction. Such a setup simplifies learning process, allowing for the direct application of backpropagation-based algorithms to adjust the model parameters. In contrast, the objective of Reinforcement Learning (RL) algorithms is to maximize the total reward accumulated over a sequence of interactions with an environment (Montague, 1999). Unlike typical supervised learning, successful RL algorithms must address two fundamental challenges stemming from this interaction model (Osband et al., 2020):

- **Exploration-exploitation tradeoff**: The agent needs to decide whether to exploit actions known to be effective or to explore new, potentially better (or worse) actions. This involves strategically prioritizing which feedback to learn from.
- **Long-term consequences**: The agent must consider how its current actions might affect opportunities and outcomes far into the future, beyond the immediate next step.

One approach to tackle the exploration-exploitation tradeoff is posterior sampling, where an agent maintains a distribution over optimal policies and acts according to samples from this belief. While effective in simpler settings, maintaining an exact posterior is computationally intractable for high-dimensional, non-linear function approximators, such as deep neural networks. This has driven the development of approximate methods that can capture the benefits of posterior sampling without the prohibitive computational cost.

In this work, we investigate the behavior of bootstrap-based posterior sampling for exploration. Bootstrapped DQN (BDQN) (Osband et al., 2016a) is a prominent example, training an ensemble of Q-value functions on different subsets of experience. While empirically successful, BDQN is known to suffer from posterior collapse, where all ensemble members converge to a single, overconfident estimate, thereby losing the diversity needed for sustained exploration. As a remedy, Randomized-Prior BDQN (RP-BDQN) (Osband et al., 2018) was introduced, which adds a unique, frozen prior network to each ensemble member. This modification is designed to anchor each member in a different part of the parameter space, thereby enforcing diversity and improving exploration, as illustrated in Figure 1.

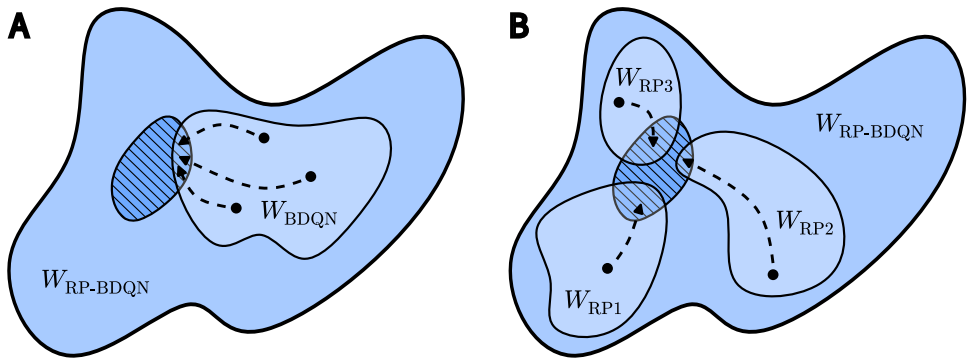

Figure 1: Schematic parameter-space view of how bootstrap ensembles form surrogate posteriors. **A** - Bootstrapped DQN with ensemble size $K = 3$. All ensemble members are attracted toward the *same* low-loss basin (hatched region in W_{BDQN}). The training dynamics (black arrows) funnel all members into the single low-loss region, causing the surrogate posterior to become sharply concentrated and epistemic uncertainty to collapse.
B - Bootstrapped DQN with Randomized Priors, ensemble size $K = 3$. Each ensemble member W_{RP1}, W_{RP2}, W_{RP3} carries an independent, frozen prior network that shifts its loss surface, producing distinct low-loss regions. The diversity is naturally preserved, and the posterior collapse is structurally prevented by the hard constraint on the parameter space.

In this work, we address the following research questions:

1. How do BDQN and RP-BDQN scale with the *ensemble size* and a task-specific *hardness* parameter?
2. Can we describe the probability of these methods discovering a solution using a closed-form scaling law that holds robustly across different hyperparameter settings?
3. Where does the identified scaling law break down, and are there any properties of the ensemble that affect it?

Contributions

- We present a systematic analysis of BDQN and RP-BDQN across over 40,000 configurations, revealing that a **single, unified scaling law** governs the convergence behavior of both methods. We show that the performance difference is captured by a single, method-dependent effectiveness parameter, ψ, which we link to the algorithm's ability to maintain ensemble diversity.
- We provide evidence that RP-BDQN extends the performance boundary beyond where BDQN breaks down and exhibits consistently more robust convergence.
- We offer a diagnostic framework and have open-sourced our experimental setup[1] to facilitate future studies on epistemic exploration.

[1]Accessible at `https://github.com/knyazer/bootstrapped-dqn-scaling`

Taken together, our results offer an easy-to-follow guide for practitioners: RP-BDQN consistently outperforms BDQN, and the effect of using ensemble sizes larger than 10 diminishes significantly. For theorists, our findings raise an intriguing question: What algorithmic properties are captured by the effectiveness parameter ψ, and why do the proposed scaling laws break down beyond a certain point? Can this scaling law inspire new exploration methods that directly optimize for the parameter ψ?

2 Background

Markov Decision Process. A further layer of complexity arises in many RL scenarios where the environment is a 'black box.' We model the environment as a Markov Decision Process (MDP) (S, A, P, R, γ), where S is a set of states, A is a set of actions, $P(s' \mid s, a)$ is the state transition probability, $R(s, a)$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. The optimal policy in such an environment is

$$\pi^\star \;=\; \arg\max_{\pi} \; \mathbb{E}_{\substack{a_t \sim \pi(\cdot|s_t), \\ s_{t+1} \sim P(\cdot|s_t, a_t)}} \Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\Big],$$

The internal dynamics, $P(s' \mid s, a)$, of such an environment are usually hidden from the learning algorithm, preventing methods that might, for instance, differentiate through the environment to directly calculate how changes in actions would affect rewards (Suh et al., 2022). This paper is concerned with epistemic exploration, a problem of tackling the exploration-exploitation dilemma by systematically reducing an agent's uncertainty about the environment's rewards. The goal is to gather just enough knowledge to make informed decisions about which actions are optimal without excessive exploration.

Q-Learning. Q-learning is a model-free algorithm that attempts to learn the optimal action-value function, $Q^\star(s, a)$, which quantifies the expected return of taking action a in state s and acting optimally thereafter Montague (1999). It operates by iteratively updating its estimate $Q(s, a)$ using observed transitions (s, a, r, s'). The update is based on the temporal-difference target, $r + \gamma \max_{a'} Q(s', a')$, which is derived from the Bellman optimality equation Bellman (1966). In simple cases with discrete states and actions, Q can be represented as a table, which simplifies the learning. Once learned, the optimal policy is to greedily select the action with the highest Q-value in a given state.

Deep Q-Learning. Deep Q-Networks (Mnih et al., 2013) combine Q-learning with non-linear function approximation. A neural network with parameters θ represents $Q_\theta(s, a)$, with (s, a) being the state-action pair. Training proceeds by stochastic gradient descent on the one-step temporal-difference error:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}}\Big[\big(r + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a)\big)^2\Big],$$

where γ is a TD factor, r is an immediate reward, (s', a') is the next-step state-action pair, θ^- are periodically updated (lagged) *target-network* parameters and $\mathcal{D}$ is a replay buffer of experience tuples.

Posterior sampling for exploration. Reinforcement learning agents often grapple with a data-collection bottleneck because the reward signals are frequently sparse and delayed. Consequently, agents must actively probe and experiment within their environment to uncover behaviors that lead to high cumulative returns. One of the issues with the greedy DQN approach is that it chooses the optimal action based on its belief. This leads to DQN getting stuck using a locally optimal policy and failing to conduct any exploration. In simpler, small-scale, tabular MDPs, where all states and actions can be explicitly enumerated, the exploration-exploitation tradeoff is well-understood and can be managed using principled methods like posterior sampling (Osband et al., 2016b; Strens, 2000). During posterior sampling, the agent maintains a posterior distribution (belief) over plausible value functions (or models), and acts according to samples drawn from this distribution. However, for the complex, non-linear function approximations (typically deep neural networks) that are central to modern deep reinforcement learning, performing exact posterior inference is computationally prohibitive. This creates a need for tractable surrogates: practical methods that can (i) approximate Bayesian-like exploration closely enough to guide the agent effectively and (ii) scale to models with potentially millions of parameters.

Several approaches aim to approximate the Bayesian posterior to achieve efficient exploration. Some methods aim for direct approximation, for instance, using Markov Chain Monte Carlo techniques (Hastings, 1970; Ishfaq et al., 2023), Sequential Monte Carlo (Van der Vaart et al., 2024) or variational approximations to the Bayesian posterior (Srivastava et al., 2014; Fortunato et al., 2018). In contrast, an alternative line of research, and a focus of this work, utilizes the stationary distribution of the optimization algorithm itself as a surrogate posterior (Osband et al., 2016a, 2018). While it has been shown that stochastic gradient descent iterates can yield a posterior that is approximately Bayesian under certain conditions (Mandt et al., 2017), the properties of stationary distributions for more complex optimizers, such as Adam (Kingma and Ba, 2015), are less characterized. Nevertheless, a significant benefit of using such surrogate posteriors is the simplicity and efficiency of sampling from them, making them an attractive choice for driving exploration.

Bootstrap Ensembles as Approximate Posterior Distributions. A prominent line of work in this area uses *bootstrap sampling* to mimic Thompson sampling (Thompson, 1933). This involves maintaining K different estimates of the value function (which predicts future rewards), each trained on a stochastically resampled subset of the collected data. In the linear setting (where value functions are linear models), Randomized Least-Squares Value Iteration (RLSVI) (Osband et al., 2016b) provided an early analytical connection, showing that bootstrap-style perturbations can approximate Gaussian posteriors and achieve polynomial sample efficiency. Osband et al. (2016a) extended this concept to deep neural networks. The method attaches a separate output head (a linear layer) for each of the K Q-value estimates and uses masks so that gradients for each head flow only through a specific subset of data mini-batches. This results in an ensemble of DQNs whose stationary weight distribution can approximate a Bayesian posterior, particularly in the limit of small learning rates (Mandt et al., 2017). In general, however, this method does not necessarily approximate a Bayesian posterior; notably, this does not make it less useful for efficient exploration. Each ensemble member is trained on its own target network, allowing for the correct propagation of time-discounted uncertainties (Osband et al., 2016a).

Injecting priors. Despite its empirical success, vanilla BDQN has a limitation: its belief about the environment is formed from the *same* training data for all the models. Consequently,

in regions of the state-action space that the agent has not visited, its uncertainty can diminish too quickly due to the alignment of training gradients for different ensemble members. To address this, Osband et al. (2018) proposed adding a frozen, untrainable *prior network* to each member of the ensemble. The resulting *Randomized-Prior BDQN* (RP-BDQN) maintains computational tractability while restoring a more robust (non-degenerate) posterior, and it maintains theoretical guarantees under linear function approximation. Meng et al. (2023) attempted to replace the priors with white noise; however, we were unable to reproduce their results. Therefore, our work focuses on the canonical and widely adopted RP-BDQN formulation. In RP-BDQN, each of the K ensemble members (with parameters θ_k and the frozen prior p_k) is trained to minimize the following loss:

$$L(\theta) = \frac{1}{K}\sum_{k=1}^{K} \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(y_k - (Q_k(s,a;\theta_k) + p_k(s,a))\right)^2\right]$$
$$\text{where} \quad y_k = r + \gamma \max_{a'}(Q_k(s',a';\theta_k^-) + p_k(s',a')).$$

Towards scaling laws for exploration. While BDQN and RP-BDQN are widely used and have demonstrated strong empirical performance, a systematic understanding of *how* their effectiveness scales with factors like task difficulty and ensemble size is still lacking. Drawing inspiration from the power-law tradition observed in fields like language modeling (Kaplan et al., 2020) and supervised deep learning (Hestness et al., 2017), our goal is to identify simple empirical regularities that can help practitioners make informed decisions about resource allocation when using these methods in deep RL.

3 Experimental Setup

To derive empirical scaling laws for exploration, we must first construct a methodology that allows us to (1) precisely control the difficulty of the exploration problem and (2) robustly measure an agent's ability to solve it. This section outlines our approach, detailing the environment used to parameterize exploration hardness, the metric used to quantify success, the specific agents subjected to this test, and the quantitative models that emerge from the results.

3.1 Parameterizing Exploration Hardness with DeepSea

A core challenge of epistemic exploration is correctly valuing knowledge collection when extrinsic rewards are sparse. To study this in isolation, we require an environment where exploration difficulty can be tuned via a single, interpretable parameter. We use the **DeepSea** environment for this purpose (Osband et al., 2018, 2020).

The DeepSea environment, depicted in Figure 2, is an $n \times n$ grid world where the agent starts in the state $(1, 1)$ and must navigate to a single, high-reward goal state at (n, n). A reward of $+1$ is given only upon reaching this goal; all transitions to the right yield a small negative reward of $-0.01/n$, and all transitions to the left yield no reward. The main feature of DeepSea is that there is a unique path to get a positive reward, and the length of this path is directly proportional to n. This parameter n thus serves as our **hardness parameter**: increasing n creates an exponentially more complex exploration problem by requiring the agent to commit to a longer, penalized path. This provides a clean, one-dimensional axis of

difficulty against which we can measure agent performance. To avoid the implicit bias of deep neural networks, a randomized action map is used: at each state, the action to move left or right is randomly swapped according to a random action map pre-initialized once before the training run.

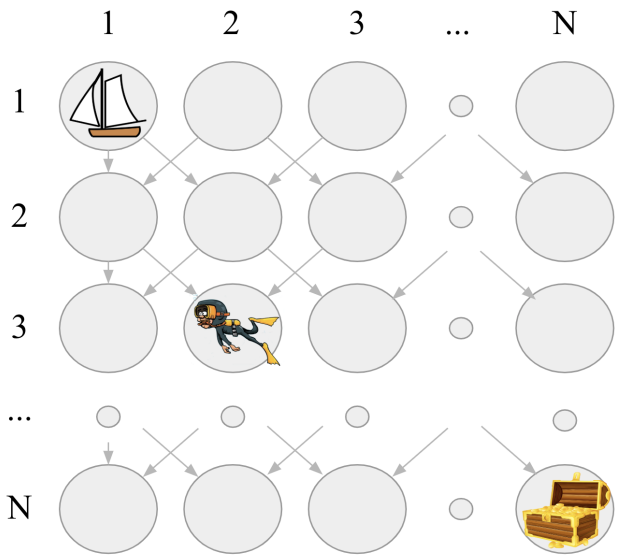

Figure 2: **DeepSea**(n) **environment.** The agent starts at the top-left corner; the only rewarding state is the bottom-right corner, and any right move incurs a small penalty, while any left move yields no reward. Larger grids require proportionally longer penalized action sequences, making exploration harder. Diagram adapted from Osband et al. (2020).

3.2 Measuring Exploration: The Probability of Discovery

Our research questions center on the probability of an agent converging to a good solution. Rather than measuring full convergence to an optimal policy, we focus on the event of **discovery**. We define a single experimental run as a 'success' if the agent finds the goal state and receives the +1 reward for the first time within a fixed budget of environment steps.

We adopt this metric because it isolates the exploration capability of the agent from its subsequent policy optimization. We assume that once a high-reward trajectory is discovered and added to the replay buffer, any standard Q-learning method will eventually learn the optimal policy. The actual bottleneck is finding that trajectory in the first place. Our metric, the *Probability of Discovery* (PoD), is therefore the fraction of runs (over multiple random seeds for each configuration) that achieve this discovery event within the episode budget, directly quantifying the likelihood that a given exploration strategy will work for a problem of a certain hardness.

3.3 Agents and Implementation Details

We use our testbed to compare the scaling properties of two key posterior sampling approximations: BDQN and RP-BDQN.

- **Bootstrapped DQN (BDQN):** An ensemble of K DQN networks. Our implementation, which does not use a shared network 'core,' is more aligned with the RP-BDQN architecture and can be viewed as a no-prior variant of Osband et al. (2018), with similar performance to that of Osband et al. (2016a).
- **Randomized-Prior BDQN (RP-BDQN):** An exact reproduction of Osband et al. (2018), which extends BDQN by adding the output of a fixed, randomly initialized prior network to each of the K separate ensemble members.

Environment and Compute. All agents are implemented in JAX (Bradbury et al., 2018) and Equinox (Kidger and Garcia, 2021). Our DeepSea implementation is based on Gymnax (Lange, 2022) with minor corrections, as shown in Appendix A, enabling fully GPU-accelerated training. To ensure computational equivalence, we scale both the learning rate and batch size linearly with ensemble size K. This is required to maintain the same first and second moments of the per-member updates for different ensemble sizes. For completeness, we include the derivation of this fact in Appendix B. The replay buffer size is held constant, and all ensemble members share transitions. For each algorithm and each pair (n, K), we perform 32 training runs using different seeds, with a limit of 50,000 episodes per run.

Network Architecture and Hyperparameters. To ensure a fair comparison, all agents use the same underlying Q-network and prior network architecture: a Multi-Layer Perceptron (MLP) with two hidden layers of 32 units and GLU activation function (Dauphin et al., 2017). Most of the hyperparameters are held constant across all experiments and are listed in Table 1. We train our model using the AdamW (Loshchilov and Hutter, 2019) optimizer, without learning rate decay.

Table 1: Default hyperparameters used for all agents, with K corresponding to ensemble size.

Hyperparameter	Value
Learning rate	3e-4$\times K$
Weight decay	1e-6
Replay buffer size	10,000
Discount factor (γ)	0.99
Batch size	$128 \times K$
Target network update frequency	500 steps
Prior scale (for RP-BDQN)	3.0

3.4 Experimental Protocol

Our investigation proceeds from a large-scale grid search in DeepSea across the ensemble size $K \in \{1, 2, 3, 4, 6, 10, 16, 20, 24, 32, 40\}$. For each algorithm and value of K, we start from $n = 3$ and increase it until we observe zero successful runs (out of 32 random seeds) for three consecutive values of n.

4 The Scaling Law

To answer the question 'What is the global performance landscape for each method?', we visualize the Probability of Discovery as a function of both task hardness (n) and ensemble

size (K). Figure 3 presents these results as a heatmap and an efficiency frontier.

For both BDQN and RP-BDQN, increasing the ensemble size yields diminishing returns, offering only a modest extension of the solvability frontier past $K = 20$. However, even a small RP-BDQN ensemble can solve problems that are entirely intractable for a BDQN ensemble 5 times larger. We note that using Randomized Priors increases the maximum solvable hardness of the problem by a factor of 1.5 using the hyperparameter configuration specified in Table 1.

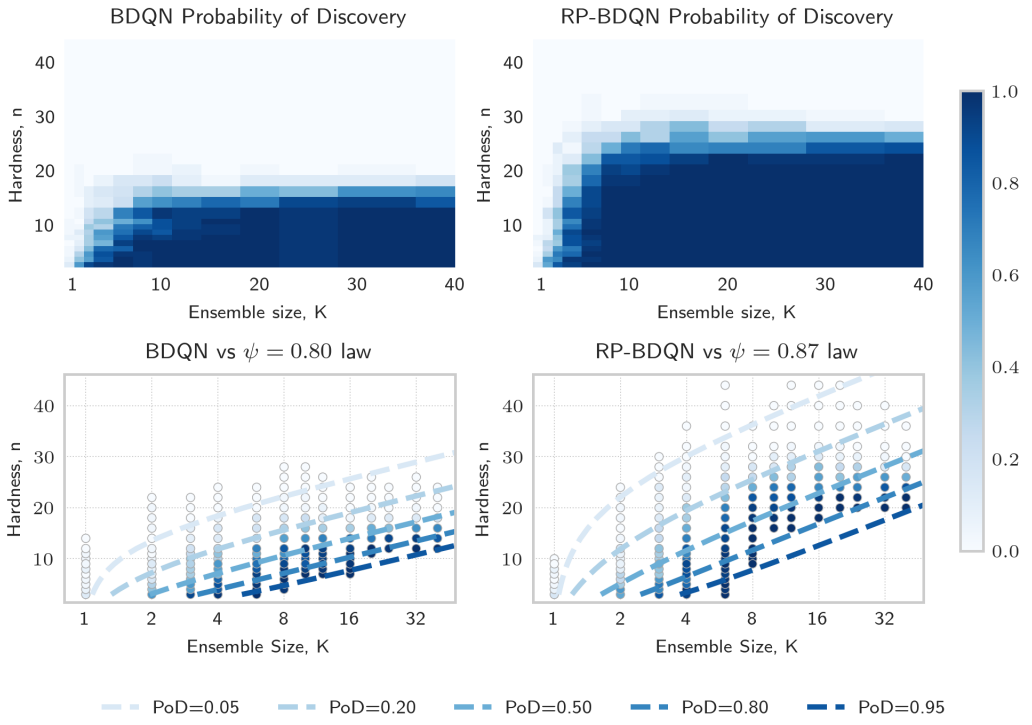

Figure 3: **Scaling laws of BDQN and RP-BDQN.** The probability of discovery (PoD) is the effective probability that at least one agent in an ensemble of size K solves a problem of hardness n. **Top row**: Empirical PoD for BDQN (left) and RP-BDQN (right). **Bottom row**: Raw experimental data, where each point is colored according to its PoD. The horizontal axis is logarithmic in ensemble size. The dashed lines are contours of constant PoD from the fitted model, $p = 1 - (1 - \psi^n)^K$, with $\psi = 0.80$ for BDQN and $\psi = 0.87$ for RP-BDQN. The model accurately captures the general scaling behavior, although a mismatch for large K can be observed: the $n \geq 32, K = 32$ interval should have a probability of 0.4 according to the law but has zero empirical discoveries across all 128 runs.

4.1 A Unified Scaling Law

The observed empirical trends can be captured by the following scaling law:

$$P(\text{discovery}) \approx 1 - (1 - \psi^n)^K \tag{1}$$

This mathematical form has a direct interpretation as a 'best-of-K' trial model. The law posits that an ensemble of size K succeeds if at least one of its members solves the task. The

probability of a single member solving an n-step problem is modeled as ψ^n, where ψ is the base probability of success at a single step. This framework models task hardness n as an exponential challenge for each member, while increasing ensemble size K improves success probability.

We fit this model to our experimental data for BDQN and RP-BDQN by maximizing the log-likelihood. The resulting parameters and goodness-of-fit metrics are shown in Table 2, with the fitted equiprobability curves plotted in the bottom row of Figure 3.

Table 2: Fitted parameters and goodness-of-fit for the unified scaling law. The parameter ψ represents the base effectiveness of a single ensemble member. The goodness-of-fit (R^2) measures the proportion of variance in the mean performance explained by the model. The **Dispersion** quantifies how well the model captures the data's variability; a value of 1 would indicate a perfect match. Our dispersions of 4.1 and 8.1 show that the model is oversimplified. Values after $\pm$ represent 95% confidence intervals computed via bootstrap.

Algorithm	**Parameter (ψ)**	**Goodness-of-fit (R^2)**	**Dispersion**	**MSE**
BDQN	0.80 ± 0.02	0.84	4.1	0.024
RP-BDQN	0.87 ± 0.01	0.69	8.1	0.049

4.2 Model Interpretation and Limitations

While the model's R^2 value of approximately 0.8 indicates it is a useful heuristic, its core assumption that the K members act as independent agents is flawed. Our experiments reveal a divergence from this idealized model. When fit to independent DQN runs, the effectiveness factor is merely $\psi \approx 0.4$, in sharp contrast to the $\psi \approx 0.80$ achieved by bootstrap-based variants. By sharing a replay buffer and collectively shaping the data-collection policy, ensemble members engage in implicit cooperation that isolated agents lack.

This failure of the independence assumption is also exposed by the model's high dispersion. Our measured dispersion values indicate that the observed performance is 4 to 8 times more variable than the simple scaling law anticipates, which would be considered a 'strong lack of fit' in natural sciences. This discrepancy is a consequence of the flawed assumption: complex interactions and shared experiences within the ensemble introduce sources of variability that a model of non-interacting agents cannot capture.

The parameter ψ should thus be interpreted as an 'effective' discovery probability that attempts to capture these emergent properties. The model's limitations become clearer still in the analysis of its residuals (Figure 4), which show a distinct U-shaped structure. The model consistently overestimates performance in two regimes: for small ensembles that lack cooperative benefits, and for very large ensembles where performance saturates and yields diminishing returns.

In summary, while the scaling law provides a valuable first-order approximation, its systematic failures underscore the complex and cooperative dynamics of ensemble-based exploration, which remain an area for future investigation.

4.3 The Underlying Mechanism: Ensemble Diversity

The performance heatmaps and scaling laws demonstrate that RP-BDQN is substantially more effective than BDQN, but they do not explain why. We hypothesize that the performance

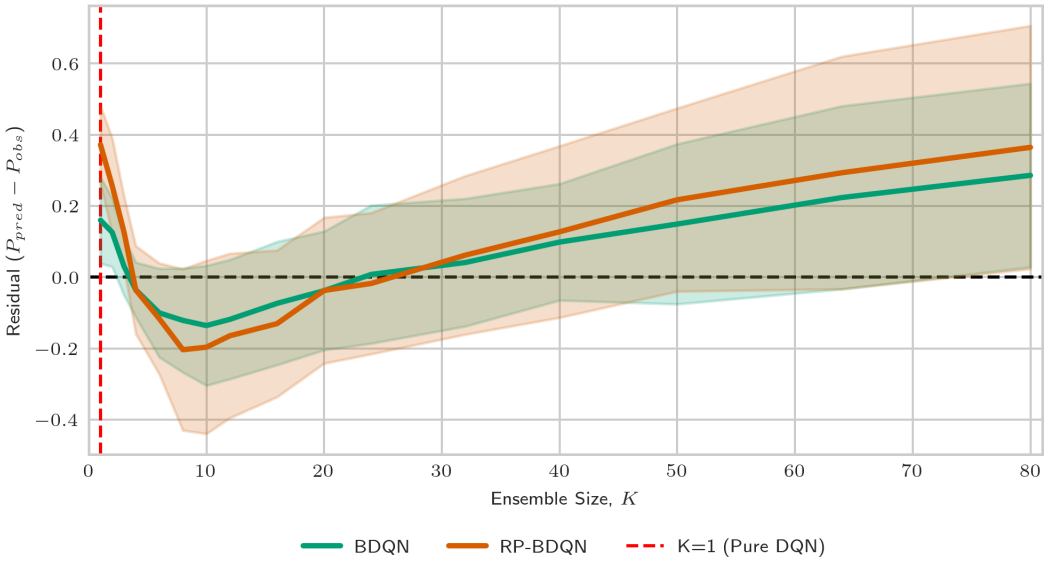

Figure 4: **Structured Residual errors of the Scaling Law.** The plot shows the prediction error, in terms of Probability of Discovery, of our scaling law as a function of ensemble size K, averaged over task hardness n. The plot is U-shaped, indicating that the law overestimates the performance for small and large numbers of ensembles.

gap stems directly from RP-BDQN's ability to maintain a diverse set of exploratory policies, whereas vanilla BDQN is prone to premature posterior collapse, where all members converge to a single, often suboptimal, policy.

To test this hypothesis, we need a quantitative measure of ensemble diversity. While diversity in the high-dimensional parameter space is difficult to track, we can readily measure it in the function space of Q-values. We propose the following metric for **Q-Diversity**:

1. At the beginning of each training run, we sample a fixed set of 64 probe states, $\mathcal{S}_{probe}$, from the environment.

2. Periodically during training, for each state $s \in \mathcal{S}_{probe}$, we compute the Q-values, $Q_k(s, a)$, for all actions a and for each of the K ensemble members.

3. The Q-Diversity at a given time is the standard deviation of these Q-values across the ensemble members, averaged over all 64 probe states.

A high Q-Diversity value indicates that the ensemble members disagree on the values of actions, suggesting they are pursuing different exploratory strategies. A low value signifies consensus and a collapse in exploration.

We analyze this metric in the most informative regime: a region of task difficulty where success is uncertain ($8 \leq n \leq 12$ and $3 \leq K \leq 6$). We partition the runs from this region into two groups: 'convergent' (those that successfully find the goal) and 'non-convergent' (those that do not). Figure 5 plots the evolution of the average Q-Diversity for these groups.

The results provide clear, empirical evidence for our hypothesis. For RP-BDQN, Q-Diversity remains high throughout the training process; the randomized priors effectively prevent the ensemble from collapsing. Runs that eventually converge maintain high diversity

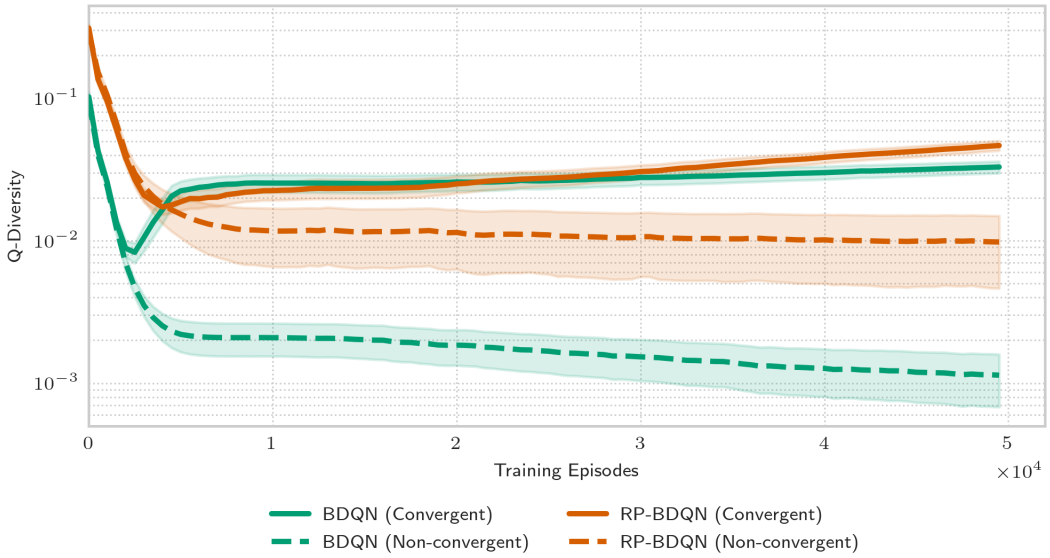

Figure 5: **Ensemble Collapse Throughout Training.** The plot shows the evolution of Q-Diversity (standard deviation of Q-values across the ensemble members) throughout training, with shaded regions corresponding to the standard deviation of the Q-Diversity across distinct training runs. Data is aggregated from a region where the PoD is approximately 0.5: $8 \leq n \leq 12$ and $3 \leq K \leq 6$. One can observe that the collapse for RP-BDQN happens later, and the overall diversity is higher even for non-convergent runs, indicating that randomized priors help to avoid posterior collapse.

for a significant portion of the training, only decreasing after the optimal policy is likely to be found. In contrast, the BDQN runs show a premature drop in diversity, and the non-convergent runs exhibit tenfold lower variability than RP-BDQN counterparts. This indicates that the ensemble members quickly agree on a suboptimal greedy policy, effectively ending meaningful exploration and trapping the agent in a suboptimal solution.

This analysis confirms that the superior performance of RP-BDQN is not merely an incidental benefit but likely a consequence of its mechanism: the enforced preservation of ensemble diversity.

4.4 Robustness Across Hyperparameters

An important question is whether our proposed scaling law and the observed performance gap between BDQN and RP-BDQN are merely artifacts of our chosen hyperparameters or if they represent a more general principle. To validate the robustness of our findings, we conducted a series of hyperparameter sweeps around our default configuration. We systematically varied the learning rate ($\eta \in \{8 \cdot 10^{-5}, 5 \cdot 10^{-4}, 10^{-3}\}$), replay buffer size ($|\mathcal{D}| \in \{5000, 20000, 40000\}$), and, for RP-BDQN, the prior scale ($\beta \in \{1.0, 5.0, 10.0\}$).

For each new hyperparameter configuration, we re-ran the entire evaluation across the lower-resolution range of task hardness (n) and ensemble size (K) and re-fit our scaling law (Equation 1) to determine the effective *best-of-one* success parameter, ψ. This allows us to measure how the fundamental effectiveness of each algorithm changes rather than just observing performance on a single task. The results, summarized in Figure 6, demonstrate

the remarkable robustness of our central finding.

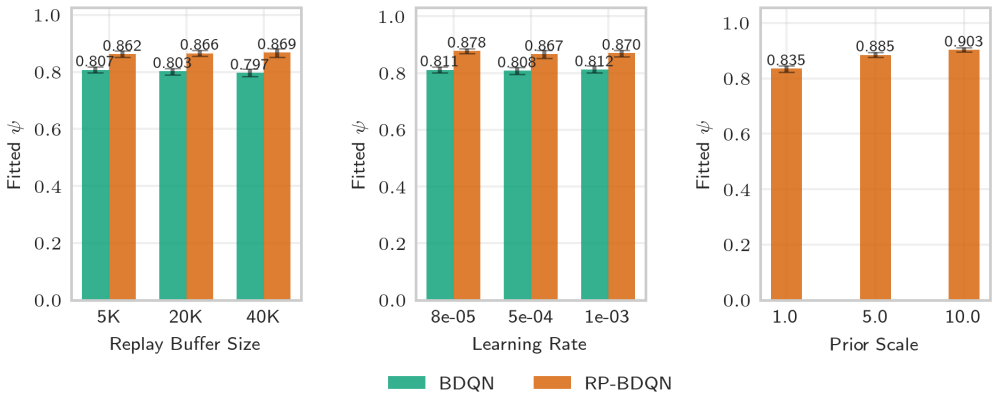

Figure 6: **Robustness of the Scaling Law Parameter ψ Across Hyperparameters.** The fitted *best-of-one* success parameter, ψ, is plotted for sweeps over (a) prior scale, (b) learning rate, and (c) replay buffer size. The error bars represent the 95% confidence interval from the model fit.

To view the complete heatmaps and efficiency frontiers for all hyperparameter settings, interested readers should consult subsection B.5. While the choice of hyperparameters modulates the absolute value of ψ, the ψ for RP-BDQN remains consistently higher than that of BDQN. We observe several secondary trends:

- **Learning Rate and Replay Buffer:** For both algorithms, performance is sensitive to these standard hyperparameters. However, the scaling law parameter is *not* sensitive, which shows that the scaling law is robust with respect to these hyperparameters.
- **Prior Scale (β):** The prior scale, which controls the magnitude of the randomized prior's contribution, has a notable impact on RP-BDQN's performance. Our default value of 3.0 is already strong, but the sweep suggests that, in general, increasing this prior scale significantly improves performance. The prior work Osband et al. (2018) suggested that a prior scale of 5 could be optimal. However, we find no saturation of performance past this value. This could be attributed to the slight differences in our training pipeline.

5 Responsible Research

Climate Change. All experiments in this paper were conducted on a single Nvidia RTX3090 or Nvidia RTX4090 GPU under 500 hours of total active compute time, which corresponds to a total of 500 kWh. We estimate the total emissions to be 120kg CO2 (Nowtricity, 2025), which we compensate for by buying carbon credits. The full certificate is available upon request.

Unethical Applications. Our paper is concerned with empirical properties of scaling a certain family of reinforcement learning algorithms. While we are not aware of any existing applications of these algorithms in any way that could harm people, we are able to conceive plausible harmful applications. However, we believe the benefit of having this paper published

outweighs the harm induced by possible unethical applications. If anyone has concerns about harm induced by this paper, they should immediately contact the author to discuss taking down the work.

Reproducibility. To support reproducibility, we provide an easy setup to produce all the plots and results used in the paper. We use uv for Python dependency management and Nix (Dolstra et al., 2004) for OS dependency management, which we wrap into a Docker container. The code is released under the MIT License.

6 Discussion

Our large-scale study reveals a simple scaling law, $P(\text{discovery}) \approx 1 - (1 - \psi^n)^K$, governing bootstrapped exploration. This law unifies task hardness (n), ensemble size (K), and algorithmic effectiveness (ψ), providing a quantitative lens to compare strategies. It reduces the complex behavior of BDQN and RP-BDQN to a single parameter, ψ, directly measuring their relative efficacy.

While our BDQN results align with prior work (Osband et al., 2018, 2016a), our RP-BDQN implementation solves tasks up to $n \approx 30$, compared to their reported $n \approx 50$. We attribute this gap to hyperparameter sensitivity rather than a flaw in the algorithm. Their work aimed to push performance limits, whereas ours focuses on characterizing the underlying scaling behavior. This highlights that while RP-BDQN is robustly superior to BDQN, achieving its state-of-the-art potential requires careful tuning.

Practically, our findings are clear: RP-BDQN is the superior algorithm, and ensemble benefits diminish significantly beyond $K = 10$. Practitioners with a fixed computational budget should prefer a moderately-sized RP-BDQN ensemble over an excessively large one.

However, the scaling law is an approximation. U-shaped residuals shown in Figure 4 reveal that the model's independence assumption is an oversimplification. For small ensembles, the model overestimates performance by ignoring cooperative benefits; for large ensembles ($K > 25$), it is again too optimistic, likely due to saturated diversity and correlated failures.

The primary limitation of this study is its reliance on the DeepSea grid world. The clean structure and parameterized hardness of this environment may not generalize to high-dimensional settings like Atari or robotics, where even defining "hardness" is a challenge. Nonetheless, this work establishes a methodological template for investigating the scaling properties of deep RL exploration.

7 Conclusions and Future Work

We presented a unified scaling law that models reward discovery as a function of task hardness and ensemble size, governed by an effectiveness parameter, ψ, in deterministic, sparse-reward settings. We showed that RP-BDQN is substantially more effective than BDQN due to its ability to maintain ensemble diversity, and we offered practical guidance, noting that returns diminish for ensemble sizes $K > 10$.

Our findings open several avenues for future research:

- **Optimizing for ψ:** Moving from a descriptive to a prescriptive use of our scaling law by designing algorithms or regularization schemes that explicitly aim to maximize ψ.

- **Generalization to Complex Environments:** Verifying whether similar scaling laws hold in more complex domains, such as Atari games or robotic control, and investigating how to define task hardness in such settings.
- **Refining the Scaling Law:** Developing a more nuanced model that incorporates terms to account for the cooperative effects at small K and the correlated failures at large K, thereby better capturing ensemble dynamics.

By establishing a quantitative framework for scaling, we hope to advance the field toward more predictable and practical exploration algorithms.

References

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966. doi: 10.1126/science.153.3731.34. URL `https://www.science.org/doi/abs/10.1126/science.153.3731.34`.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *34th International Conference on Machine Learning, ICML 2017*, volume 2, 2017.

Eelco Dolstra, Andres de Jonge, and Eelco Visser. Nix: A safe and policy-free system for software deployment. In *18th USENIX Conference on System Administration (LISA)*, pages 79–92. USENIX Association, 2004. URL `https://dspace.library.uu.nl/bitstream/handle/1874/18006/dolstra_04_nix_a_safe.pdf`.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57, 1970. ISSN 00063444. doi: 10.1093/biomet/57.1.97.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically, 2017. URL `https://arxiv.org/abs/1712.00409`.

Haque Ishfaq, Qingfeng Lan, Pan Xu, Ashique Rupam Mahmood, Doina Precup, Anima Anandkumar, and Kamyar Azizzadenesheli. Provable and practical: Efficient exploration in reinforcement learning via langevin monte carlo. *ArXiv*, abs/2305.18246, 2023. URL `https://api.semanticscholar.org/CorpusID:258959015`.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL `https://arxiv.org/abs/2001.08361`.

Patrick Kidger and Cristian Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

Robert Tjarko Lange. gymnax: A JAX-based reinforcement learning environment library, 2022. URL `http://github.com/RobertTLange/gymnax`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18, 2017. ISSN 15337928.

Li Meng, Morten Goodwin, Anis Yazidi, and Paal E. Engelstad. Improving the diversity of bootstrapped dqn by replacing priors with noise. *IEEE Transactions on Games*, 15, 2023. ISSN 24751510. doi: 10.1109/TG.2022.3185330.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. In *NIPS Deep Learning Workshop*, 2013.

P.Read Montague. Reinforcement learning: An introduction, by sutton, r.s. and barto, a.g. *Trends in Cognitive Sciences*, 3, 1999. ISSN 13646613. doi: 10.1016/s1364-6613(99)01331-5.

Nowtricity. Co_2 emissions per kwh in netherlands, 2025. URL `https://www.nowtricity.com/country/netherlands/`.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, 2016a.

Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. In *33rd International Conference on Machine Learning, ICML 2016*, volume 5, 2016b.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 2018-December, 2018.

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvari, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado Van Hasselt. Behaviour suite for reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 2014. ISSN 15337928.

Malcolm Strens. A bayesian framework for reinforcement learning. *Proc of the 17th International Conference on Machine Learning*, 2000.

H. J.Terry Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In *Proceedings of Machine Learning Research*, volume 162, 2022.

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933. doi: 10.2307/2332286.

Pascal R. Van der Vaart, Neil Yorke-Smith, and Matthijs Spaan. Bayesian ensembles for exploration in deep reinforcement learning. In *Proceedings of the 2024 International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2024)*, Auckland, New Zealand, 2024.

A Environment implementation.

Our study required a high-performance implementation of the DeepSea environment to facilitate large-scale experimentation. We chose to build upon the JAX-based `gymnax` library (Lange, 2022) to enable fully GPU-accelerated training and environment stepping, thereby minimizing CPU-GPU data transfer overhead.

During initial validation, we observed anomalously high performance, with agents solving tasks of hardness $n = 100$ with small ensemble sizes. An investigation revealed that the existing `gymnax` implementation had the randomized action map disabled by default - a "debug" configuration from the original environment specification (Osband et al., 2020). We corrected this discrepancy to enforce the proper exploration challenge. To guarantee correctness, we verified our fixed implementation against the original by running approximately $20,000$ in automated tests confirming trajectory-wise reward equivalence between the two versions.

Performance Optimization. To maximize memory bandwidth and training throughput, we implemented a transparent observation compression mechanism within our replay buffer. While the agent interacts with standard (n, n)-shaped observation tensors, the buffer internally represents each observation as a single integer. This optimization resulted in a threefold improvement in overall training performance.

B Consistent moments of the per-member gradients.

B.1 Implementation: Averaged Loss over Split Batches

For each optimization step:

1. A total data batch of size $B_{\text{total}} = K \cdot B_1$ is sampled, where K is the ensemble size and B_1 is the baseline batch size.

2. This batch is split into K unique, non-overlapping mini-batches $(D_1, D_2, \ldots, D_K)$, each of size B_1.

3. Each ensemble member k, with its independent parameters θ_k, calculates its loss $\ell_k(\theta_k; D_k)$ on its corresponding mini-batch D_k.

4. The final loss function to be differentiated is the average of these individual losses: $L_{\text{avg}} = \frac{1}{K}\sum_{j=1}^{K} \ell_j(\theta_j; D_j)$.

5. The learning rate is scaled linearly: $\eta_K = K \cdot \eta_1$.

B.2 Derivation of the Per-Member Gradient

The key insight comes from calculating the gradient of the total average loss L_{avg} with respect to the parameters θ_k of a single, specific member k. Because the members do not share parameters, the loss of member j (ℓ_j) does not depend on the parameters of member k for any $j \neq k$. Therefore, $\nabla_{\theta_k} \ell_j(\theta_j; D_j) = 0$ for all $j \neq k$.

The gradient for member k is:

$$\begin{aligned}
\nabla_{\theta_k} L_{\text{avg}} &= \nabla_{\theta_k} \left(\frac{1}{K} \sum_{j=1}^{K} \ell_j(\theta_j; D_j) \right) \\
&= \frac{1}{K} \sum_{j=1}^{K} \nabla_{\theta_k} \ell_j(\theta_j; D_j) \\
&= \frac{1}{K} \left(\nabla_{\theta_k} \ell_1 + \cdots + \nabla_{\theta_k} \ell_k + \cdots + \nabla_{\theta_k} \ell_K \right) \\
&= \frac{1}{K} \nabla_{\theta_k} \ell_k(\theta_k; D_k)
\end{aligned}$$

Let $g_k = \nabla_{\theta_k} \ell_k(\theta_k; D_k)$ be the gradient for member k on its own mini-batch. The gradient used by the optimizer for member k's parameters is simply $\frac{1}{K} g_k$.

B.3 Analysis of the Per-Member Parameter Update

The parameter update $\Delta\theta_k$ for member k is calculated using the scaled learning rate η_K and its specific gradient component $\frac{1}{K} g_k$:

$$\Delta\theta_k = \eta_K \cdot \left(\frac{1}{K} g_k \right)$$

Substituting the scaled learning rate $\eta_K = K\eta_1$:

$$\Delta\theta_k = (K\eta_1) \cdot \left(\frac{1}{K} g_k \right) = \eta_1 g_k$$

This is a powerful result: the scaling of the learning rate by K and the down-scaling from the averaged loss by $1/K$ perfectly cancel. The resulting update rule for an ensemble member, $\Delta\theta_k = \eta_1 g_k$, is identical in form to the update rule for a baseline single agent, where g_k is computed on a batch of size B_1.

B.4 Moment Equivalence

We can now confirm that the moments of the per-step update are identical to the baseline case.

- **First Moment (Mean):** The expectation of the update for an ensemble member is:

$$\mathbb{E}[\Delta\theta_k] = \mathbb{E}[\eta_1 g_k] = \eta_1 \mu_k$$

  This is identical to the baseline agent's mean update.

- **Second Moment (Covariance):** The covariance of the update for an ensemble member is:

$$\text{Cov}(\Delta\theta_k) = \text{Cov}(\eta_1 g_k) = \eta_1^2 \text{Cov}(g_k) = \frac{\eta_1^2}{B_1}\Sigma_k$$

  This is identical to the baseline agent's update covariance.

B.5 Conclusion

The "split batch" implementation with a scaled learning rate ensures that the statistical properties (mean and covariance) of the per-step parameter update for each ensemble member are identical to those of a single agent trained with baseline hyperparameters (B_1, η_1). The total batch size is scaled to $K \cdot B_1$ to make efficient use of parallel hardware, allowing K models to be trained for the wall-clock time of one, thus achieving a K-fold speedup in terms of data processed over time. This methodology provides a foundation for achieving computationally equivalent training across different ensemble sizes.

C Hyperparameter sweep details.

This appendix contains the full performance landscape plots (probability of discovery heatmaps and solvability frontiers) for each configuration tested in the hyperparameter sweep (Section 4.4).

C.1 Learning Rate Sweep

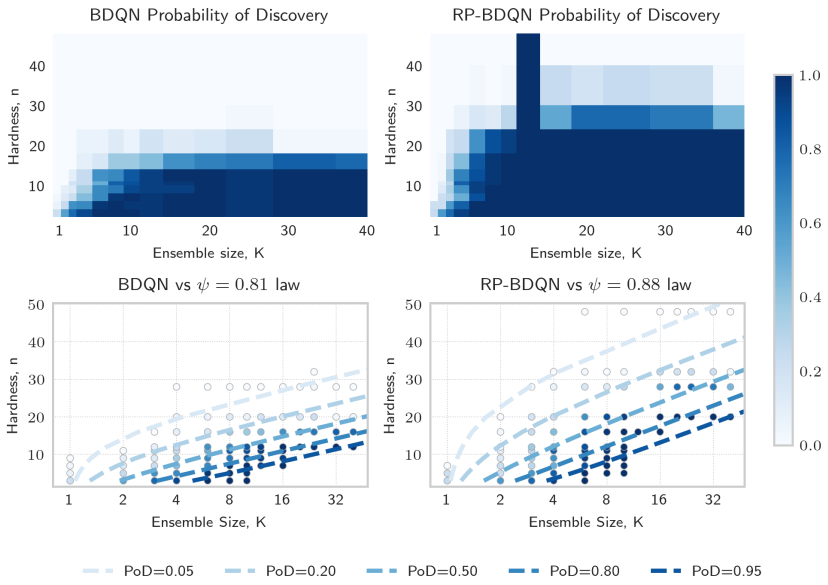

Figure 7: Performance landscape for Learning Rate $= 8 \times 10^{-5}$.

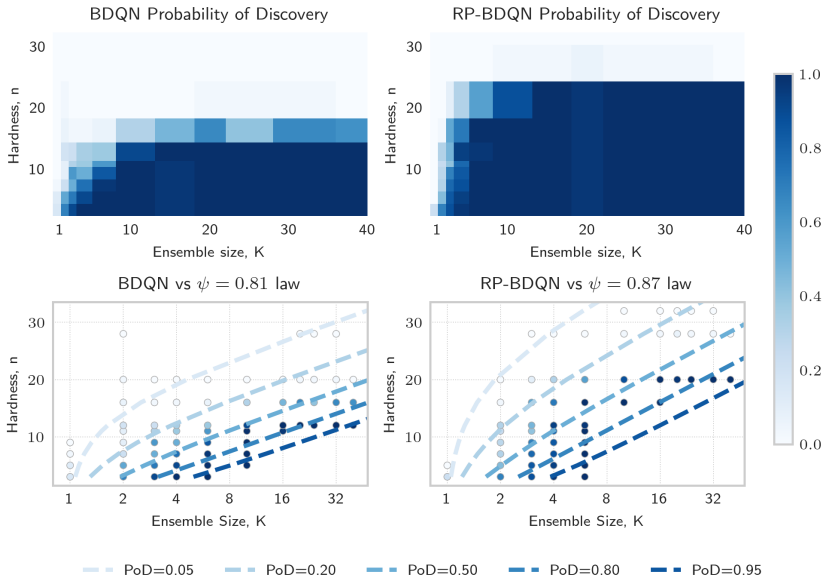

Figure 8: Performance landscape for Learning Rate $= 5 \times 10^{-4}$.

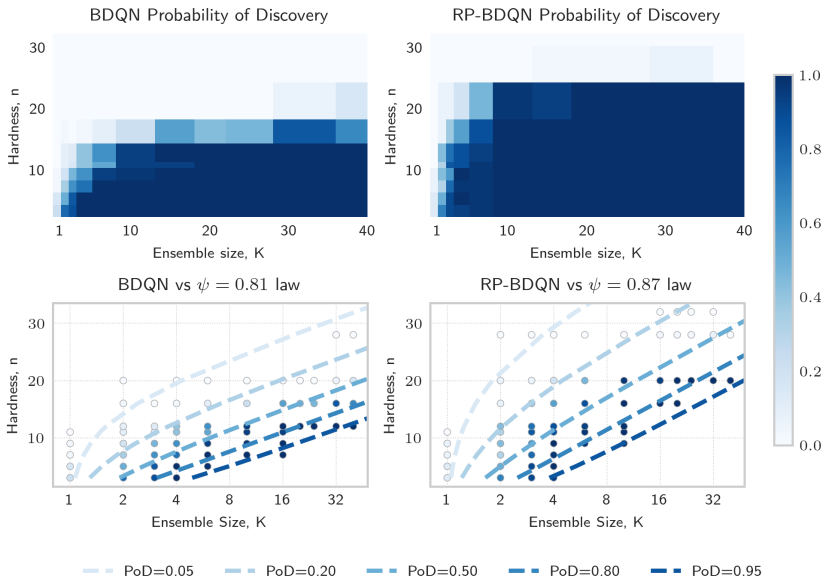

Figure 9: Performance landscape for Learning Rate = 10^{-3}.

C.2 Replay Buffer Size Sweep

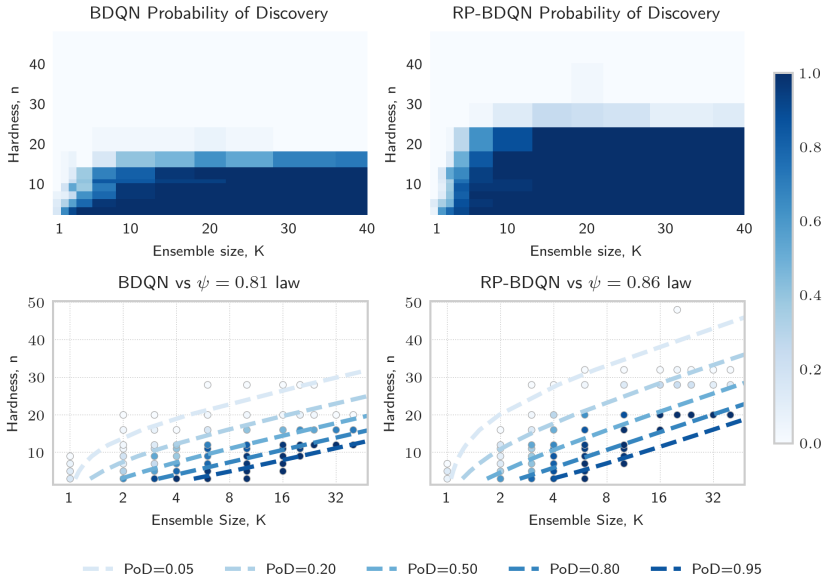

Figure 10: Performance landscape for Replay Buffer Size = 5,000.

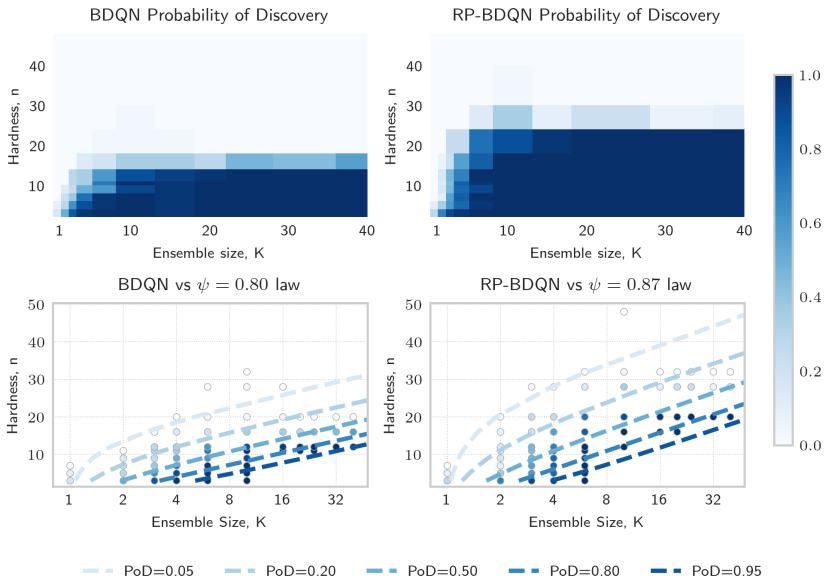

Figure 11: Performance landscape for Replay Buffer Size = 20,000.

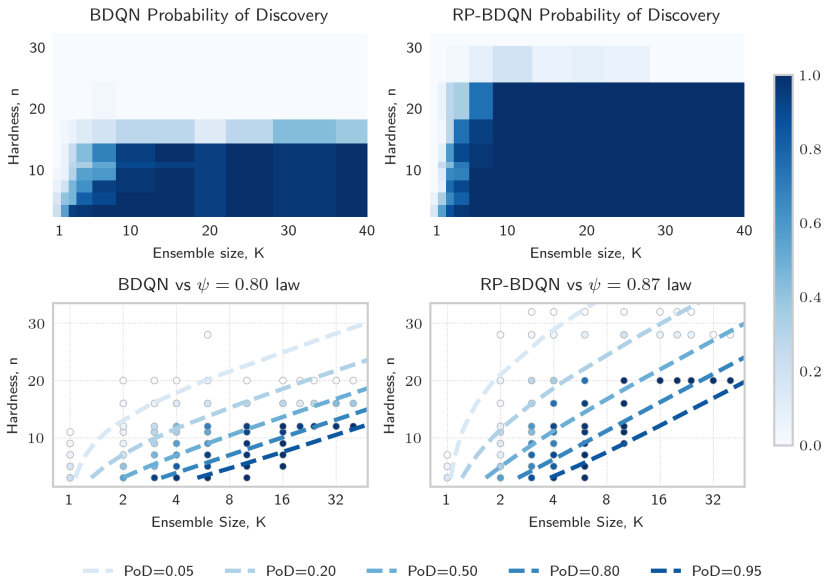

Figure 12: Performance landscape for Replay Buffer Size = 40,000.

C.3 Prior Scale Sweep (RP-BDQN only)

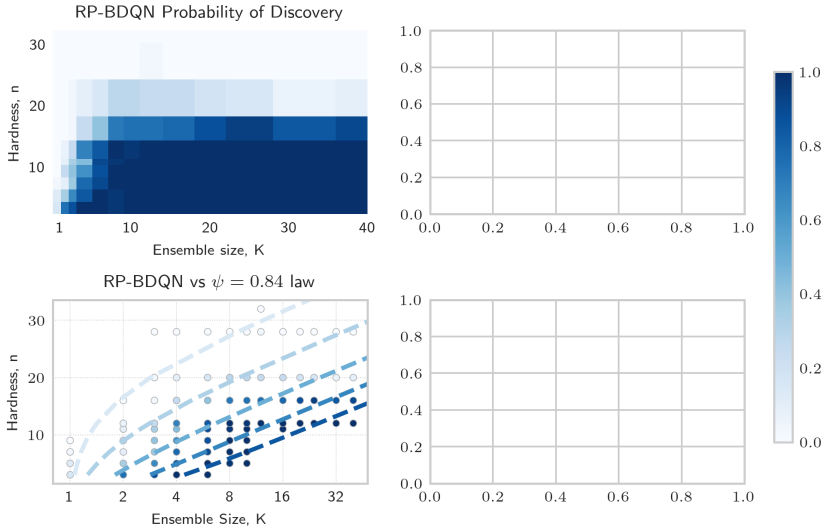

Figure 13: Performance landscape for Prior Scale $\beta = 1.0$.

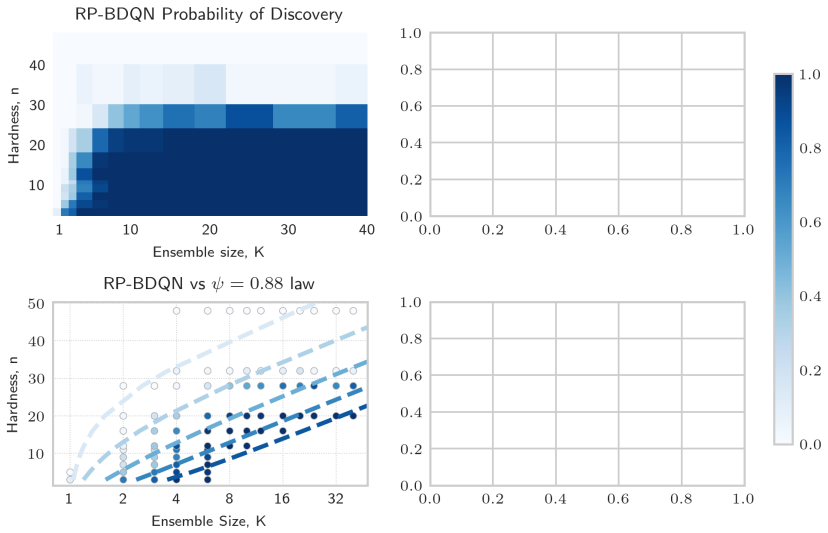

Figure 14: Performance landscape for Prior Scale $\beta = 5.0$.

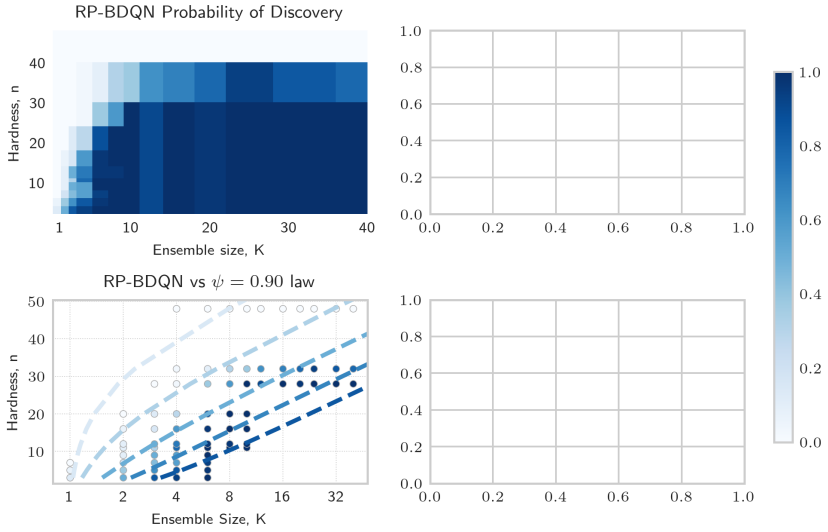

Figure 15: Performance landscape for Prior Scale $\beta = 10.0$.