Q-Star Meets Scalable Posterior Sampling: Bridging Theory and Practice via HyperAgent

Yingru Li [1 2 *] **Jiawei Xu** [1 *] **Lei Han** [3] **Zhi-Quan Luo** [1 2]

Abstract

We propose HyperAgent, a reinforcement learning (RL) algorithm based on the hypermodel framework for exploration in RL. HyperAgent allows for the efficient incremental approximation of posteriors associated with an optimal action-value function ($Q^\star$) without the need for conjugacy and follows the greedy policies w.r.t. these approximate posterior samples. We demonstrate that HyperAgent offers robust performance in large-scale deep RL benchmarks. It can solve Deep Sea hard exploration problems with episodes that optimally scale with problem size and exhibits significant efficiency gains in the Atari suite. Implementing HyperAgent requires minimal code addition to well-established deep RL frameworks like DQN. We theoretically prove that, under tabular assumptions, HyperAgent achieves logarithmic per-step computational complexity while attaining sublinear regret, matching the best known randomized tabular RL algorithm.

1. Introduction

Practical reinforcement learning (RL) in complex environments faces challenges such as large state spaces and an increasingly large volume of data. The per-step computational complexity, defined as the computational cost for the agent to make a decision at each interaction step, is crucial. Under resource constraints, any reasonable design of an RL agent must ensure bounded per-step computation, a key requirement for *scalability*. If per-step computation scales polynomially with the volume of accumulated interaction data, computational requirements will soon become unsustainable, which is untenable for scalability. *Data efficiency* in sequential decision-making demands that the agent learns the optimal policy with as few interaction steps as possible, a fundamental challenge given the need to balance exploration of the environment to gather more information and exploitation of existing information (Thompson, 1933; Lai & Robbins, 1985; Thrun, 1992). Scalability and efficiency are both critical for the practical deployment of RL algorithms in real-world applications with limited resources. There appears to be a divergence between the development of practical RL algorithms, which mainly focus on scalability and computational efficiency, and RL theory, which prioritizes data efficiency, to our knowledge. This divergence raises an important question:

Can we design a practically efficient RL agent with provable guarantees on efficiency and scalability?

1.1. Key Contributions

This work affirmatively answers the posed question by proposing a novel reinforcement learning algorithm, `HyperAgent`, based on the hypermodel framework (Li et al., 2022; Dwaracherla et al., 2020; Osband et al., 2023b). We highlight the advantages of `HyperAgent` below:

- *Algorithmic Simplicity*. `HyperAgent`'s implementation[1] requires only the addition of a single module, the *last-layer linear hypermodel*, to the conventional DDQN framework and a minor modification for action selection. The added module facilitates efficient incremental approximation and sampling for the posteriors associated with the $Q^\star$ function. Practically, `HyperAgent` can be a replacement for the ε-greedy method in most of its usecases. This simplicity contrasts with state-of-the-art methods for the Atari benchmarks, which often rely on multiple complex algorithmic components and extensive tuning.
- *Practical Efficiency*. The `HyperAgent` algorithm demonstrates exceptional scalability and efficiency in challenging environments. Notably, in the Deep Sea exploration scenario (Osband et al., 2020), it efficiently solves problems up to 120×120 in size with optimal

[*]Equal contribution [1]The Chinese University of Hong Kong, Shenzhen [2]Shenzhen Research Institue of Big Data [3]Tencent AI and Robotics X. Correspondence to: Yingru Li <szrlee@gmail.com or yingruli@link.cuhk.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

[1]We provide the open-source code at https://github.com/szrlee/HyperAgent.

	Practice in Deep RL			**Theory** in Tabular RL	
Algorithm	**Tractable**	**Incremental**	**Efficient**	**Regret**	**Per-step Computation**
PSRL	✗	✗	✗	$\tilde{O}(H^2\sqrt{SAK})$	$O(S^2A)$
RLSVI	✓	✗	✗	$\tilde{O}(H^2\sqrt{SAK})$	$O(S^2A)$
Ensemble+	✓	✓	🟡	N/A	N/A
HyperAgent	✓	✓	✓	$\tilde{O}(H^2\sqrt{SAK})$	$\tilde{O}(\log(K)SA + S^2A)$

Table 1. Milestones of RL algorithms evaluated in both practice and theory: PSRL (Strens, 2000; Osband & Van Roy, 2017), RLSVI (Osband et al., 2016b; 2019), Ensemble+ (Osband et al., 2018; 2019), and our HyperAgent. In deep RL, 🟡 indicate intermediate outcomes between ✓ and ✗. In tabular RL, we consider the number of states (S), actions (A), horizons (H), and episodes (K).

episode complexity, as shown in Figure 3. Furthermore, HyperAgent achieves human-level performance on the Atari benchmark suite (Bellemare et al., 2013), as detailed in Figure 1, requiring only 15% of the interaction data (1.5M interactions) and 5% of the network parameters necessary for DDQN† (Double Deep Q-Networks, van Hasselt et al. (2016)) and BBF (Bigger, Better, Faster, Schwarzer et al. (2023)), respectively. In contrast, Ensemble+ (Osband et al., 2019), a randomized exploration method, achieves a mere 0.22 IQM score with 1.5M interactions but uses double the parameters of our approach.

- *Provable Guarantees.* We prove that HyperAgent achieves sublinear regret in a tabular, episodic setting with $\tilde{O}(\log K)$ per-step computation over K episodes. This performance is supported by an incremental posterior approximation argument central to our analysis (Lemma 4.1). This argument is proved by a reduction to the sequential random projection (Li, 2024a).

HyperAgent effectively bridges the theoretical and practical aspects of RL in complex, large-scale environments. See Table 1 for representative milestones of the RL algorithms.

1.2. Related Works

The modern development of practical RL algorithms provides scalable solutions to the challenges posed by large state spaces and the increasing size of interaction data under resource constraints. These algorithms' per-step computation complexity scales sub-linearly with (1) the problem size, thanks to function approximation techniques (Bertsekas & Tsitsiklis, 1996; Mnih et al., 2015); and (2) the increasing size of interaction data, due to advancements in temporal difference learning (Sutton & Barto, 2018), Q-learning (Watkins & Dayan, 1992), and incremental SGD with finite buffers (Mnih et al., 2015). These advances yielded impressive results in simulated environments and attracted significant interest (Mnih et al., 2015; Schrittwieser et al., 2020). However, data efficiency remains a barrier to transferring this success to the real world (Lu et al., 2023).

To address data-efficiency empirically, recent deep RL algorithms have incorporated increasingly complex heuristic and algorithmic components, such as DDQN (van Hasselt et al., 2016), Rainbow (Hessel et al., 2018), EfficientZero (Ye et al., 2021), and BBF (Schwarzer et al., 2023) whose details are around Table 3 in Appendix A. Moreover, these algorithms lack theoretical efficiency guarantees; for example, BBF employs ε-greedy exploration, which is provably data inefficient, requiring an exponential number of samples (Kakade, 2003; Strehl, 2007; Osband et al., 2019; Dann et al., 2022). ε-greedy strategy is still popular in practice due to its simplicity in implementation, requiring very few additional lines of code. We aim to develop a simple replacement for ε-greedy by HyperAgent for practical concerns while achieving data efficient exploration with performance guarantees.

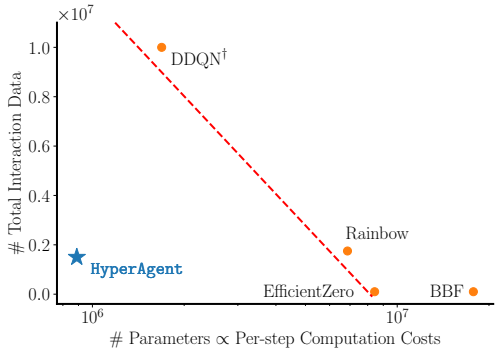


Figure 1. This evaluation explores the relationship between the amount of training data required and the model parameters necessary to achieve human-level performance, quantified by a 1.0 IQM score. It is assessed across 26 Atari games using the Interquartile Mean (IQM) metric (Agarwal et al., 2021) using recent state-of-the-art (SOTA) algorithms. The number of parameters is directly proportional to the computational cost, as they predominantly influence the calculation during each SGD update per interaction step. HyperAgent, denoted by ★, achieves a 1.0 IQM score with a comparatively minimal number of interactions and parameters.

Efficient exploration in reinforcement learning hinges on decisions driven not only by expectations but also by epistemic uncertainty (Russo et al., 2018; Osband et al., 2019). Such decisions are informed by immediate and subsequent observations over a long horizon, embodying the concept of deep exploration (Osband et al., 2019). Among the pivotal exploration strategies in sequential decision-making is Thompson Sampling (TS), which bases decisions on a

posterior distribution over models, reflecting the degree of epistemic uncertainty (Thompson, 1933; Strens, 2000; Russo et al., 2018). In its basic form, TS involves sampling a model from the posterior and selecting an action that is optimal according to the sampled model. However, exact posterior sampling remains computationally feasible only in simple environments–like Beta-Bernoulli and Linear-Gaussian Bandits, as well as tabular MDPs with Dirichlet priors over transition vectors–where conjugacy facilitates efficient posterior updates (Russo et al., 2018; Strens, 2000).

To extend TS to more complex environments, approximations are indispensable (Russo et al., 2018), encompassing both function approximation for scalability across large state spaces and posterior approximation for epistemic uncertainty estimation beyond conjugate scenarios. Randomized Least-Squares Value Iteration (RLSVI) represents another value-based TS approach, aiming to approximate posterior sampling over the optimal value function without explicitly representing the distribution. This method achieves tractability for value function approximation by introducing randomness through perturbations, thus facilitating deep exploration and enhancing data efficiency (Osband et al., 2019). Despite avoiding explicit posterior maintenance, RLSVI demands significant computational effort to generate new point estimates for each episode through independent perturbations and solving the perturbed optimization problem anew, without leveraging previous computations for incremental updates. Consequently, while RLSVI remains feasible under value function approximation, its scalability is challenged by growing interaction data, a limitation shared by subsequent methods (Ishfaq et al., 2021).

Bridging the Gap. The divergence between theoretical and practical realms in reinforcement learning (RL) is expanding, with theoretical algorithms lacking practical applicability and practical algorithms exhibiting empirical and theoretical inefficiencies. Ensemble sampling, as introduced by Osband et al. (2016a; 2018; 2019), emerges as a promising technique to approximate the performance of RLSVI. Further attempts such as Incre-Bayes-UCBVI and Bayes-UCBDQN (Tiapkin et al., 2022) incorporate ensemble-based empirical bootstraps to approximate Bayes-UCBVI, which aims to bridge this gap. These methods maintain multiple point estimates, updated incrementally, essential for scalability. However, the computational demand of managing an ensemble of complex models escalates, especially as the ensemble size must increase to accurately approximate complex posterior distributions (Dwaracherla et al., 2020; Osband et al., 2023b; Li et al., 2022; Qin et al., 2022). An alternative strategy involves leveraging a hypermodel (Dwaracherla et al., 2020; Li et al., 2022) or epistemic neural networks (ENN) (Osband et al., 2023b;a) to generate approximate posterior samples. This approach, while promising, demands a representation potentially more intricate than simple point estimates. The computational overhead of these models, including ensembles, hypermodels, and ENNs, is theoretically under-explored. More discussion of related works can be found in Appendix A.

2. Reinforcement Learning & Hypermodel

We consider the episodic RL setting in which an agent interacts with an unknown environment over a sequence of episodes. We model the environment as a Markov Decision Problem (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, s_{\text{terminal}}, \rho)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\text{terminal} \in \mathcal{S}$ is the terminal state, and ρ is the initial state distribution. For each episode, the initial state S_0 is drawn from the distribution ρ. At each time step $t = 1, 2, \ldots$ within an episode, the agent observes a state $S_t \in \mathcal{S}$. If $S_t \neq s_{\text{terminal}}$, the agent selects an action $A_t \in \mathcal{A}$, transits to a new state $S_{t+1} \sim P(\cdot \mid S_t, A_t)$, with reward $R_{t+1} = r(S_t, A_t, S_{t+1})$. An episode terminates once the agent arrives at the terminal state. Let τ be the termination time , i.e., $S_\tau = s_{\text{terminal}}$. A policy $\pi : \mathcal{S} \to \mathcal{A}$ maps a state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$. For each MDP $\mathcal{M}$ and each policy π, we define the associated action-value function as

$$Q^\pi_{\mathcal{M}}(s, a) := \mathbb{E}_{\mathcal{M},\pi}\left[\sum_{t=1}^{\tau} R_t \mid S_0 = s, A_0 = a\right],$$

where the subscript π under the expectation indicates that actions over the time periods are selected according to the policy π. Let $V^\pi_{\mathcal{M}}(s) := Q^\pi_{\mathcal{M}}(s, \pi(s))$. We further define the optimal value function $V^\star_{\mathcal{M}}(s) = \max_\pi V^\pi_{\mathcal{M}}(s)$ for all $s \in \mathcal{S}$ where it takes the maximum on optimal policy π^*. Optimal policy also corresponds to the optimal action-value function, denoted $Q^\star$ and defined as

$$Q^\star(s, a) = Q^{\pi^*}_{\mathcal{M}}(s, a) \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A}. \tag{1}$$

In the reinforcement learning problem, the agent is given knowledge about $\mathcal{S}, \mathcal{A}, r, s_{\text{terminal}}$, and ρ, but is uncertain about transition P. The unknown MDP $\mathcal{M}$, together with the unknown transition function P, are modeled as random variables drawn from a prior distribution. As a consequence, the optimal action-value function $Q^\star$ is also a random variable that the agent is uncertain about at the beginning. Thus, the agent needs to explore the environment and gather information to resolve this uncertainty.

2.1. Hypermodel

As maintaining the degree of uncertainty (Russo et al., 2018) is crucial for data-efficient sequential decision-making, we build RL agents based on the hypermodel (Li et al., 2022; Dwaracherla et al., 2020) framework for epistemic uncertainty estimation. The hypermodel takes an input $x \in \mathbb{R}^d$

and a random index $\xi \sim P_\xi$ from a fixed reference distribution, producing an output $f_\theta(x, \xi)$ that reflects a sample from the approximate posterior, measuring the degree of uncertainty. The variation in the hypermodel's output with ξ captures the model's degree of uncertainty about x, providing a dynamic and adaptable approach to uncertainty representation. This design, combining a trainable parameter θ with a constant reference distribution P_ξ, allows the hypermodel to adjust its uncertainty quantification over time, optimizing its performance and decision-making capabilities in dynamic environments. **(1)** For example, a special case of a linear hypermodel is $f_\theta(x, \xi) = \langle x, \mu + \mathbf{A}\xi \rangle$ with $\theta = (\mathbf{A} \in \mathbb{R}^{d\times M}, \mu \in \mathbb{R}^d)$ and $P_\xi = N(0, I_M)$, which is essentially the Box-Muller transformation: one could sample from a linear-Gaussian model $N(x^\top \mu, x^\top \Sigma x)$ via a linear hypermodel if $\mathbf{A}\mathbf{A}^\top = \Sigma$. **(2)** Another special case is the ensemble sampling: with a uniform distribution $P_\xi = \mathcal{U}(e_1, \ldots, e_M)$ and an ensemble of models $\theta = \mathbf{A} = [\tilde{\theta}_1, \ldots, \tilde{\theta}_M] \in \mathbb{R}^{d\times M}$ such that $\tilde{\theta}_m \sim N(\mu, \Sigma)$, one can uniformly sample from these ensembles by a form of hypermodel $f_\theta(x, \xi) := \langle x, \mathbf{A}\xi \rangle$. In general, the hypermodel $f_\theta(\cdot)$ can be any function approximator, e.g., neural networks, transforming the reference distribution P_ξ to an arbitrary distribution. We adopt a class of hypermodel that can be represented as an additive function of a learnable function and a fixed prior model (additive prior assumption),

$$\underbrace{f_\theta(x,\xi)}_{\text{“Posterior” Hypermodel}} = \underbrace{f_\theta^L(x,\xi)}_{\text{Learnable function}} + \underbrace{f^P(x,\xi)}_{\text{Fixed prior model}} \quad (2)$$

The prior model f^P represents the prior bias and prior uncertainty, and it has no trainable parameters. The learnable function is initialized to output values near zero and is then trained by fitting the data. The resultant sum $f_\theta(x, \cdot)$ produces reasonable predictions for all probable values of ξ, capturing epistemic uncertainty. This additive prior assumption can be validated under linear-Gaussian model (Osband et al., 2018), also related to Matheron's rule (Journel & Huijbregts, 1976; Hoffman & Ribak, 1991; Doucet, 2010) with applications in Gaussian processes (Wilson et al., 2020).

As described in Figure 2, we design the hypermodel for feed-forward neural networks (NN) with the *last-layer linear hypermodel* assumption: the degree of uncertainty can be approximated by a linear hypermodel $f_\theta(\cdot, \xi) = \langle \phi_w(\cdot), w_{\text{predict}}(\xi) \rangle$ over the last-layer $w_{\text{predict}}(\xi) = \mathbf{A}\xi + b$. This new assumption is unconventional to the literature (Dwaracherla et al., 2020; Li et al., 2022; Osband et al., 2023b), and can be validated when the hidden-layer feature mapping $\phi_w(\cdot)$ is fixed through the learning process, as proved in Section 4. We discuss the details of the last-layer linear hypermodel and clarify the critical differences and advantages compared with prior works in Appendix C.3.

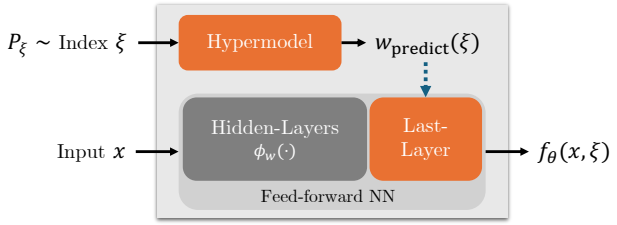


Figure 2. Last-layer linear hypermodel.

3. Algorithm design

We now describe HyperAgent, a DQN-type algorithm for large-scale complex environments. HyperAgent consists of three key components:

1. A hypermodel that maintains an approximate distribution over optimal value function $Q^\star$.
2. An incremental update mechanism to update the hypermodels.
3. An index sampling scheme that uses the hypermodels for exploration.

In the context of reinforcement learning, we define the action-value function with hypermodel as $f_\theta : \mathcal{S} \times \mathcal{A} \times \Xi \to \mathbb{R}$ parameterized by θ, where Ξ is the index space. As we introduce random index following reference distribution P_ξ as an input, the f_θ is essentially a randomized value function. Under the last-layer linear hypermodel assumption, we define the action-value function $f_\theta(s, a, \xi)$ as

$$\underbrace{\langle \mathbf{A}^{(a)}\xi + b^{(a)}, \phi_w(s) \rangle}_{\text{Learnable } f_\theta^L(s,a,\xi)} + \underbrace{\langle \mathbf{A}_0^{(a)}\xi + b_0^{(a)}, \phi_{w_0}(s) \rangle}_{\text{Fixed prior } f^P(s,a,\xi)} \quad (3)$$

where θ includes a set of learnable parameters $\{w, \mathbf{A}^{(a)}, b^{(a)}\}$ and fixed parameters $\{w_0, \mathbf{A}_0^{(a)}, b_0^{(a)}\}$ for each action $a \in \mathcal{A}$. The action-value function f_θ based on the hypermodel is then trained by minimizing the loss function motivated by fitted Q-iteration (FQI), a classical method (Ernst et al., 2005) for batch-based function approximation for optimal action-value $Q^\star$, with a famous online extension called DQN (Mnih et al., 2015). An important notion we introduce that differentiate HyperAgent to DQN is the *index mapping* $\boldsymbol{\xi}^- : \mathcal{S} \to \Xi$. The *index sampling* procedure is to produce random variables $\boldsymbol{\xi}^-(s)$ following P_ξ for each $s \in \mathcal{S}$ (line 4) and to pergorm greedy action selection (line 7). Intuitively, this procedure introduces noises that are independent across episodes, and induces diverse exploration behavior.

For training, HyperAgent maintains two hypermodels: one for the main value function f_θ and the other for the target value function f_{θ^-} where θ^- is the target parameters. It

Algorithm 1 HyperAgent

1: **Input:** Reference P_ξ. Perturbation $P_{\mathbf{z}}$. Buffer D. Initialize $\theta = \theta^- = \theta_{\text{init}}$, train step $j = 0$.
2: **for** each episode $k = 1, 2, \ldots$ **do**
3: **Sample** an *index mapping* $\boldsymbol{\xi}_k(\cdot) \sim P_\xi$. Set $t = 0$ and **observe** $S_{k,0} \sim \rho$
4: **repeat**
5: **Select** $A_{k,t} = \arg\max_{a \in \mathcal{A}} f_\theta(S_{k,t}, a, \boldsymbol{\xi}_k(S_{k,t}))$
6: **Observe** $S_{k,t+1}$ from the environment and $R_{k,t+1} = r(S_{k,t}, A_{k,t}, S_{k,t+1})$
7: **Sample** perturbation random vector $\mathbf{z}_{k,t+1} \sim P_{\mathbf{z}}$; Add $(S_{k,t}, A_{k,t}, R_{k,t+1}, S_{k,t+1}, \mathbf{z}_{k,t+1})$ to buffer D
8: Increment step counter $t \leftarrow t + 1$; Compute $\theta, \theta^-, j \leftarrow \texttt{update}(D, \theta, \theta^-, \boldsymbol{\xi}^- = \boldsymbol{\xi}_k, t, j)$ with Algorithm 2
9: **until** $S_{k,t} = s_{\text{terminal}}$
10: **end for**

also maintains a buffer of transitions $D = \{(s, a, r, s', \mathbf{z})\}$, where $\mathbf{z} \in \mathbb{R}^M$ is the algorithmic perturbation vector sampled from the perturbation distribution $P_{\mathbf{z}}$ (described in line 9). Let γ be the discounted factor. We denote the *perturbed temporal difference* (TD) loss for a given index ξ and a transition tuple d as $\ell^{\gamma,\sigma}(\theta; \theta^-, \boldsymbol{\xi}^-, \xi, d)$, defined as

$$[r + \sigma \xi^\top \mathbf{z} + \gamma \max_{a' \in \mathcal{A}} f_{\theta^-}(s', a', \boldsymbol{\xi}^-(s')) - f_\theta(s, a, \xi)]^2,$$

where σ is a hyperparameter to control the variance of algorithmic random perturbation. HyperAgent updates the hypermodel by minimizing the loss $L^{\gamma,\sigma,\beta}(\theta; \theta^-, \boldsymbol{\xi}^-, D)$ as

$$\mathbb{E}_{\xi \sim P_\xi}\left[\sum_{d \in D} \frac{1}{|D|} \ell^{\gamma,\sigma}(\theta; \theta^-, \boldsymbol{\xi}^-, \xi, d)\right] + \frac{\beta}{|D|}\|\theta\|^2, \quad (4)$$

where $\beta \geq 0$ is for prior regularization. We optimize the loss function Equation (4) using SGD with a mini-batch of data $\tilde{D}$ and a batch of indices $\tilde{\Xi}$ from P_ξ. That is, we take gradient descent w.r.t. the sampled loss $\tilde{L}(\theta; \theta^-, \boldsymbol{\xi}^-, \tilde{D})$ as

$$\frac{1}{|\tilde{\Xi}|} \sum_{\xi \in \tilde{\Xi}} \sum_{d \in \tilde{D}} \frac{1}{|\tilde{D}|} \ell^{\gamma,\sigma}(\theta; \theta^-, \boldsymbol{\xi}^-, \xi, d) + \frac{\beta}{|D|}\|\theta\|^2. \quad (5)$$

Then, the target parameters θ^- are periodically updated to θ. We summarized the HyperAgent in Algorithm 1 where the update function through Equations (4) and (5) is described in Algorithm 2.

For practitioners, the primal benefits and motivations include straightforward implementation as a plug-and-play alternative to DQN-type methods. It can also replace the ε-greedy exploration strategy. Extending actor-critic type deep reinforcement learning algorithms to incorporate similar advantages as in HyperAgent can be readily achieved.

4. Theoretical insights and analysis

In this section, we provide insights on how perturbed TD loss and index sampling works and why it performs efficient incremental posterior approximation for optimal value $Q^\star$ without reliance on conjugacy and faciliates deep exploration. For clarity, we focus on tabular representations when $\phi_w(s) = \phi_{w_0}(s) = \mathbb{1}_s \in \mathbb{R}^{|\mathcal{S}|}$ is a fixed one-hot vector.

Tabular setups. Let us define short notations $m_{sa} = \left(\mathbf{A}^{(a)}\right)^\top \phi_w(s)$ and $\mu_{sa} = \left(b^{(a)}\right)^\top \phi_w(s)$ for ease of exposition. Similarly, define $m_{0,sa}$ and $\mu_{0,sa}$ from $w_0, \mathbf{A}_0^{(a)}$ and $b_0^{(a)}$, respectively. Following Equation (3),

$$f_\theta(s, a, \xi) = \underbrace{\mu_{sa} + m_{sa}^\top \xi}_{\text{Learnable } f_\theta^L(s,a,\xi)} + \underbrace{\mu_{0,sa} + m_{0,sa}^\top \xi}_{\text{Fixed Prior } f^P(s,a,\xi)}$$

where $\theta = (\mu \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}, m \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times M})$ are the parameters to be learned; $m_{0,sa} := \sigma_0 \mathbf{z}_{0,sa}$ where $\mathbf{z}_{0,sa} \in \mathbb{R}^M$ is an independent random vector sampled from $P_{\mathbf{z}}$ and $\mu_{0,sa}, \sigma_0$ are prior mean and prior variance for each $(s, a) \in \mathcal{S} \times \mathcal{A}$. The regularizer in Equation (4) now becomes $\beta\|\theta\|^2 = \beta \sum_{s,a} \left(\mu_{sa}^2 + \|m_{sa}\|^2\right)$.

History. Denote the sequence of observations in episode k by $\mathcal{O}_k = (S_{k,t}, A_{k,t}, R_{k,t+1}, S_{k,t+1})_{t=0}^{\tau_k - 1}$ where $S_{k,t}, A_{k,t}, R_{k,t+1}$ are the state, action, and reward at the t-th time step of the k-th episode, and τ_k is the termination time at episode k. We denote the history of observations made prior to episode k by $\mathcal{H}_k = (\mathcal{O}_1, \ldots, \mathcal{O}_{k-1})$. Without loss of generality, we assume that under any MDP $\mathcal{M}$ and policy π, the termination time $\tau < \infty$ is finite with probability 1. The agent's behavior is governed by the agent policy $\pi_{\text{agent}} = (\pi_k)_{k=1}^K$, which uses the history $\mathcal{H}_k$ to select a policy $\pi_k = \text{agent}(\mathcal{S}, \mathcal{A}, r, \mathcal{H}_k)$ for the k-th episode.

Therefore, we define some statistics related to the history $\mathcal{H}_k$. Let $E_k = \{0, 1, \ldots, \tau_k - 1\}$ denote the time index in episode k and $E_{k,sa} = \{t : (S_{k,t}, A_{k,t}) = (s, a), t \in E_k\}$ record the time index the agent encountered (s, a) in the k-th episode. Assume $T = \sum_{k=1}^K |E_k|$ total interactions encountered within K episodes. Let $N_{k,sa} = \sum_{\ell=1}^{k-1} |E_{\ell,sa}|$ denote the counts of visitation for state-action pair (s, a) prior to episode k. For every pair (s, a) with $N_{k,sa} > 0$,

$\forall s' \in \mathcal{S}$, the empirical transition up to episode k is

$$\hat{P}_{k,sa}(s') := \sum_{\ell=1}^{k-1} \sum_{t \in E_\ell} \frac{\mathbb{1}_{(S_{\ell,t},A_{\ell,t},S_{\ell,t+1})=(s,a,s')}}{N_{k,sa}}.$$

In case $N_{k,sa} = 0$, define $\hat{P}_{k,sa}(s') = 1$ for arbitrary s'.

Stochastic Bellman operator. For the ease of explanation, let us use the vector notation $f_{\theta,\boldsymbol{\xi}}(s,a) := f_\theta(s,a,\boldsymbol{\xi}(s))$. Initially, let $\theta_0 := (\mu_0 = \mathbf{0}, m_0 = \mathbf{0})$. At episode k, HyperAgent will act greedily w.r.t. the action-value vector $f_{\theta_k,\boldsymbol{\xi}_k}$ where $\theta_k = (\mu_k, m_k)$. That is, the agent chose policy $\pi_k(s) \in \arg\max_{a\in\mathcal{A}} f_{\theta_k,\boldsymbol{\xi}_k}(s,a)$.

Now we explain how HyperAgent performs incremental updates from θ_{k-1} to θ_k. Suppose that, at the beginning of episode k, it maintains parameters $\theta_k^{(0)} := \theta_{k-1} = (\mu_{k-1}, m_{k-1})$, the buffer $D = \mathcal{H}_k$, and index mapping $\boldsymbol{\xi}^- = \boldsymbol{\xi}_k$. By iteratively solving its objective Equation (4) with $\beta = \sigma^2/\sigma_0^2, \theta^- = \theta_k^{(i)}$ and obtaining the closed-form solution $\theta_k^{(i+1)}$ from $i = 0$ until converging to θ_k, HyperAgent would yield the closed-form iterative update rule (1) $m_{k-1} \to m_k$ and (2) $\theta_k^{(i)} = (\mu_k^{(i)}, m_k) \to \theta_k^{(i+1)} = (\mu_k^{(i+1)}, m_k)$ as follows. Using short notation $\tilde{m}_{k,sa} = m_{k,sa} + \sigma_0 \mathbf{z}_{0,sa}$, for all (s,a,k), we have

$$\begin{aligned}&(N_{k,sa}+\beta)\tilde{m}_{k,sa} - (N_{k-1,sa}+\beta)\tilde{m}_{k-1,sa} \\ &= \sum_{t\in E_{k-1,sa}} \sigma \mathbf{z}_{k-1,t+1}. \end{aligned} \tag{6}$$

More interestingly, the iterative process on θ_k^i, for $i = 0, 1, \ldots$ can be described using the following equation

$$f_{\theta_k^{(i+1)},\boldsymbol{\xi}_k} = F_k^\gamma f_{\theta_k^{(i)},\boldsymbol{\xi}_k}, \tag{7}$$

where F_k^γ can be regarded as a stochastic Bellman operator induced by HyperAgent in episode k: i.e., for any Q-value function, $F_k^\gamma Q(s,a)$ is defined as

$$\frac{\beta\mu_{0,sa} + N_{k,sa}(r_{sa} + \gamma V_Q^\top \hat{P}_{k,sa})}{N_{k,sa}+\beta} + \tilde{m}_{k,sa}^\top \boldsymbol{\xi}_k(s), \tag{8}$$

where $V_Q(s) := \max_a Q(s,a)$ is the greedy value with respect to Q. The derivations for Equations (6) to (8) can be found in Appendix C.4, which are crucial for understanding.

True Bellman operator. Before digging into Equations (6) to (8), let us first examine the true Bellman operator $F_{\mathcal{M}}^\gamma$: when applied to fixed and bounded Q, for any $(s,a) \in \mathcal{S}\times\mathcal{A}$

$$F_{\mathcal{M}}^\gamma Q(s,a) = r_{sa} + \gamma V_Q^\top P_{sa}. \tag{9}$$

Note the true Bellman operator has a fixed point on optimal action-value function, i.e., $Q^\star = F_{\mathcal{M}}^\gamma Q^\star$. Since the transition P is a random variable in our setup, the true Bellman operator is also a random variable that propagates uncertainty for $Q^\star$. As will be discussed in Lemma D.11, conditioned on the history $\mathcal{H}_k$ up to episode k, the posterior variance of then Bellman equation in Equation (9) is inversely proportion to the visitation counts on (s,a)

$$\mathrm{Var}\left(F_{\mathcal{M}}^\gamma Q(s,a) \mid \mathcal{H}_k\right) \propto \frac{1}{N_{k,sa}+\beta}. \tag{10}$$

Intuitively, more experience at (s,a) leads to less epistemic uncertainty and smaller posterior variance on $Q^\star(s,a)$.

Understanding Equation (6). This is an incremental update with computational complexity $O(M)$. A key property of the hypermodel in HyperAgent is that, with logarithmically small M, it can approximate the posterior variance sequentially in every episode via incremental update in Equation (6). This is formalized as the key lemma.

Lemma 4.1 (Incremental posterior approximation)**.** *For $\tilde{m}_k$ recursively defined in Equation* (6) *with $\mathbf{z} \sim \mathcal{U}(\mathbb{S}^{M-1})$. For any $k \geq 1$, define the good event of ε-approximation*

$$\mathcal{G}_{k,sa}(\varepsilon) := \left\{ \|\tilde{m}_{k,sa}\|^2 \in \left(\frac{(1-\varepsilon)\sigma^2}{N_{k,sa}+\beta}, \frac{(1+\varepsilon)\sigma^2}{N_{k,sa}+\beta} \right) \right\}.$$

The joint event $\cap_{(s,a)\in\mathcal{S}\times\mathcal{A}} \cap_{k=1}^K \mathcal{G}_{k,sa}(\varepsilon)$ holds with probability at least $1-\delta$ if $M \simeq \varepsilon^{-2}\log(|\mathcal{S}||\mathcal{A}|T/\delta)$.

A direct observation from Lemma 4.1 is larger M results in smaller ε, implying more accurate approximation of posterior variance over $Q^\star(s,a)$ using their visitation counts and appropriately chosen σ. As shown in Appendix D.1, the difficulty in proving Lemma 4.1 arises from the sequential dependence among high-dimensional random variables. It is resolved due to the first probability tool (Li, 2024a) for sequential random projection. Additionally, in the regret analysis, we will show constant approximation $\varepsilon = 1/2$ suffices for efficient deep exploration with logarithmic per-step computation costs. To highlight, Lemma 4.1 also validates our assumption about the *last-layer linear hypermodel* in the special case when the hidden layer feature mapping $\phi_w(\cdot)$ is a fixed one-hot mapping. We leave the exploration of general setups for $\phi_w(\cdot)$ to future work.

Understanding Equation (8). Now, we will argue that the HyperAgent-induced operator F_k^γ is essentially mimicking the behavior of the true Bellman operator $F_{\mathcal{M}}^\gamma$ conditioned on history $\mathcal{H}_k$, thus producing an approximate posterior over $Q^\star$. As shown in Lemma C.1, the stochastic Bellman operator F_k^γ is a contraction mapping and thus guarantees convergence of Equation (7). It differs from the empirical Bellman iteration $V_Q^\top \hat{P}_{k,sa}$ in two ways: (1) there is a slight regularization toward the prior mean $\mu_{0,sa}$, and (2) more importantly, HyperAgent adds noise $w_{k,sa} := \tilde{m}_{k,sa}^\top \boldsymbol{\xi}_k(s)$ to each iteration. For a common choice $\boldsymbol{\xi}_k(s) \sim P_\xi := N(0, I_M)$, the noise $w_{k,sa}$ is Gaussian distributed conditioned on $\tilde{m}_{k,sa}$. Importantly, as

shown by Lemma 4.1, the variance of the perturbation noise

$$\mathrm{Var}_{\boldsymbol{\xi}_k}\left(w_{k,sa}\right) = \|\tilde{m}_{k,sa}\|^2 \propto \frac{1}{N_{k,sa}+\beta}$$

coincides with the posterior variance of the true Bellman operator in Equation (10) up to a constant. Incorporating the Gaussian noise in the Bellman iteration would backpropagate the uncertainty estimates and approximate the posteriors associated with the optimal action-value $Q^\star$, which is essential to incentivize deep exploration behavior. This is because the injected noise w from later states with less visitation counts (thus larger variance) will be backpropagated to initial state by the `HyperAgent`. A simple illustration on this efficient deep exploration behavior is in Appendix D.2.

Regret bound. To rigorously justify the algorithmic insights and benefits, we provide theoretical results for `HyperAgent` with tabular representation and hyperparameters for `update` specified in Table 6. Denote the regret of a policy π_k over episode k by $\Delta_k := V^\star_{\mathcal{M}}(s_{k,0}) - V^{\pi_k}_{\mathcal{M}}(s_{k,0})$. Maximizing total reward is equivalent to minimizing the expected total regret: $\mathrm{Regret}(K, \mathrm{agent}) := \mathbb{E}\sum_{k=1}^{K}\Delta_k$.

Theorem 4.2. *Under Assumptions D.7 and D.8 with $\beta \geq 3$, if the Tabular* `HyperAgent` *is applied with planning horizon H, and parameters with $(M, \mu_0, \sigma, \sigma_0)$ satisfying*

$$M = O(1)\cdot\log(SAHK) = \tilde{O}(\log K),$$

$(\sigma^2/\sigma_0^2) = \beta$, $\sigma \geq \sqrt{6}H$ and $\min_{sa}\mu_{0,sa} \geq H$, then $\forall K \in \mathbb{N}$, $\mathrm{Regret}(K,$ `HyperAgent`*$)$ is upper bounded by*

$$18H^2\sqrt{\beta SAK\log_+(1+SAHK)}\log_+\left(1+K/SA\right),$$

where $\log_+(x) = \max(1, \log(x))$. The per-step computation of `HyperAgent` *is $O(S^2A + SAM)$.*

Remark 4.3. The time-inhomogeneous MDP is a common benchmark for regret analysis in the literature, e.g., (Azar et al., 2017; Osband et al., 2019). For notation unification with Table 1, as explained in Assumption D.7, $|\mathcal{S}| = SH, |\mathcal{A}| = A$ and $\tau = H$ almost surely. Assumption D.8 is common in the literature of Bayesian regret analysis (Osband et al., 2019; Osband & Van Roy, 2017; Lu & Van Roy, 2019). The regret bound $\tilde{O}(H^2\sqrt{SAK})$ of `HyperAgent` matches the best known Bayesian regret bound, such as those of RLSVI (Osband et al., 2019) and PSRL (Osband & Van Roy, 2017), while `HyperAgent` provides the computation and scalability benefits that RLSVI and PSRL do not have, as discussed in Section 1 and Table 1. On the other hand, most practically scalable algorithms, including recent BBF (Schwarzer et al., 2023), use ε-greedy, which is provably data-inefficient without sublinear regret guarantees in general (Dann et al., 2022). As shown in Table 1, a recent concurrent work claims LMC-LSVI (Ishfaq et al., 2024) is practical and provable but suffers $\tilde{O}(K)$ per-step computational complexity, which is not acceptable under bounded computation resource constraints. Instead, the per-step computation of `HyperAgent` is $\tilde{O}(\log K)$. This is due to the choice of M in Appendix D.1 when proving the key Lemma 4.1. These comparisons imply `HyperAgent` is the first provably scalable and efficient RL agent among practically useful ones. Analytical details are in Appendix D.3. Extension to frequentist regret without Assumption D.8 is direct either using `HyperAgent` or its variants of optimistic index sampling in Appendix C.2.

5. Empirical studies

This section assesses the efficiency and scalability of `HyperAgent` empirically. In DeepSea hard exploration problems, it is the *only and first* deep RL algorithm that successfully handling large state spaces up to 120×120 with optimal episodes complexity. In Atari benchmark suite, it excels in processing continuous state spaces with pixels and achieves human-level performance with comparably minimal total number of interactions. `HyperAgent` achieves the best performance in 8 hardest exploration compared to other approximate posterior sampling type deep RL algorithms. We provide reproduction details in Appendix B.

5.1. Computational results for deep exploration

We demonstrate the exploration effectiveness and scalability of `HyperAgent` utilizing DeepSea, a reward-sparse environment that demands deep exploration (Osband et al., 2020; 2019). Details for DeepSea are in Appendix B.3.

Comparative analysis. Based on the structure of DeepSea with size N, i.e., $N \times N$ states, the proficient agent can discern an optimal policy within $\Theta(N)$ episodes (Osband et al., 2019), since it learns to move right from one additional cell along the diagonal in each episode. We compare `HyperAgent` with several baselines: Ensemble+ (Osband et al., 2018; 2019), HyperDQN (Li et al., 2022), and ENNDQN (Osband et al., 2023a), which also claimed deep exploration ability. As depicted in Figure 3, `HyperAgent` outperforms other baselines, showcasing its exceptional data efficiency. To highlight, it is the *only and first* deep RL agent learning the optimal policy with optimal episodes complexity $\Theta(N)$. Moreover, `HyperAgent` offers the advantage of computation efficiency as its output layer (hypermodel) maintains *constant parameters* when scaling up the problem size. In contrast, ENNDQN requires increasing number of parameters as the problem size increases, due to the inclusion of the original state as part of the inputs (see Appendix C.3.2 for detailed discussions). For instance, in DeepSea($N = 20$), `HyperAgent` uses only 5% of the parameters required by ENNDQN.

Through more ablation studies on DeepSea in Appendix E, we offer a comprehensive understanding of `HyperAgent`,

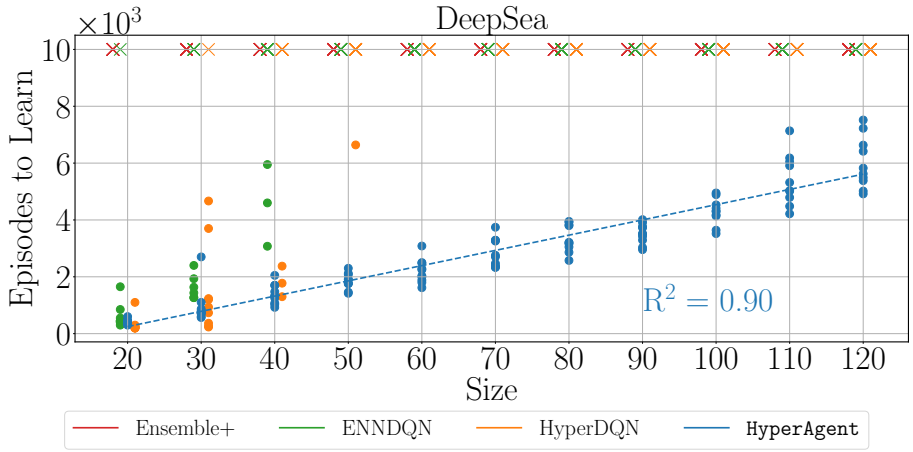


Figure 3. The metric Episodes to Learn$(N) := \text{avg}\{K|\bar{R}_K \geq 0.99\}$ measures the episodes needed to learn the optimal policy in DeepSea of size N, where $\bar{R}_K$ is the return achieved by the agent after K episodes of interaction, averaged over 100 evaluations. The crossmark ✗ denotes the algorithm's failure to solve the problem within 10^4 episodes. We conduct experiments on each algorithm with 10 different initial random seeds, presenting each result as a distinct point in the figure. The dashed line for `HyperAgent`, based on linear regression with an R^2 of 0.90, illustrates linear scaling in episode complexity, represented by $\Theta(N)$.

including validation for theoretical insights in Section 4, index sampling schemes and sample-average approximation in Section 3, and comparison with structures within the hypermodel framework in Section 2.1.

5.2. Results on Atari benchmark

Baselines. We further assess the data and computation efficiency on the Arcade Learning Environment (Bellemare et al., 2013) using IQM (Agarwal et al., 2021) as the evaluation criterion. An IQM score of 1.0 indicates that the algorithm performs on par with humans. We examine `HyperAgent` with several baselines: DDQN† (van Hasselt et al., 2016), Ensemble+ (Osband et al., 2018; 2019), Rainbow (Hessel et al., 2018), DER (van Hasselt et al., 2019), HyperDQN (Li et al., 2022), BBF (Schwarzer et al., 2023), and EfficientZero (Ye et al., 2021). Following the established practice in widely accepted research (Łukasz Kaiser et al., 2020; van Hasselt et al., 2019; Ye et al., 2021), the results are compared on 26 Atari games.

Overall results. Figure 1 illustrates the correlation between model parameters and training data for achieving human-level performance. `HyperAgent` attains human-level performance with minimal parameters and relatively modest training data, surpassing other methods. Notably, neither DER nor HyperDQN can achieve human-level performance within 2M training data (refer to Appendix F).

Ablation study. Table 2 displays the comprehensive results of `HyperAgent` across 26 Atari games. To demonstrate the superior performance of `HyperAgent` stemming from our principled algorithm design rather than the fine-tuning of hyper-parameters, we developed our version of DDQN, referred to as DDQN(ours). This implementation mirrors the hyper-parameters and network structure (except for the last layer) of `HyperAgent`. The comparative result with vanilla DDQN† (Hessel et al., 2018) indicates that (1) DDQN(ours) outperforms DDQN† due to hyperparameters adjustments, and (2) `HyperAgent` exhibits superior performance compared to DDQN(ours), owing to the inclusion of an additional hypermodel, index sampling schemes, and incremental mechanisms that facilitate deep exploration. It is worth noting that we also applied an identical set of hyper-parameters across all 55 Atari games (refer to Appendix F), where `HyperAgent` achieves top performance in 31 out of 55 games, underscoring its robustness and scalability.

Method	IQM	Median	Mean
DDQN†	0.13 (0.11, 0.15)	0.12 (0.07, 0.14)	0.49 (0.43, 0.55)
DDQN(ours)	0.70 (0.69, 0.71)	0.55 (0.54, 0.58)	0.97 (0.95, 1.00)
`HyperAgent`	1.22 (1.15, 1.30)	1.07 (1.03, 1.14)	1.97 (1.89, 2.07)

Table 2. Performance profiles of `HyperAgent` across 26 Atari games with 2M training data. The data in parentheses represent the 95% confidence interval.

Exploration on Atari. Through a comparison with algorithms related to approximate posterior sampling on the 8 most challenging exploration Atari games (Bellemare et al., 2016), as shown in Figure 4, `HyperAgent` can achieve the best performance in 7 out of 8 games, demonstrating its ability to efficiently track the approximate posteriors over $Q^\star$ and perform deep exploration.

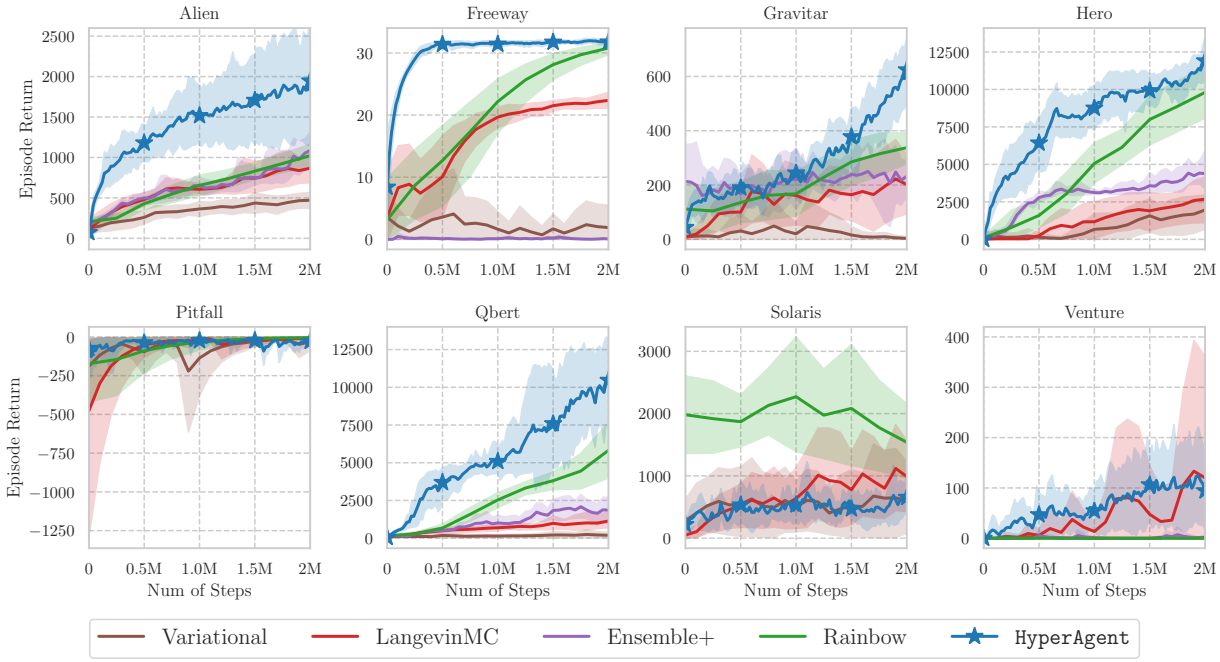


Figure 4. Comparison of HyperAgent on the 8 hardest exploration Atari games with Variational approximation (SANE Aravindan & Lee, 2021), LangevinMC (AdamLMCDQN Ishfaq et al., 2024, an extension of LMC-LSVI in deep RL), Ensemble+ (Osband et al., 2018; 2019, Ensemble sampling with a randomized prior function) and Rainbow (Hessel et al., 2018).

6. Conclusion and future directions

We present a reinforcement learning (RL) algorithm, HyperAgent, that simplifies, accelerates, and scales the learning process across complex environments. With a hypermodel, index sampling, and incremental updates, HyperAgent efficiently tracks the approximate posterior distribution associated with the optimal value function $Q^\star$ and performs the greedy policy over randomly sampled $Q^\star$ from this distribution to facilitate deep exploration. It achieves significant efficiency in data and computation over baselines. This is demonstrated through superior performance in challenging benchmarks like the DeepSea and Atari suites with minimal computational resources. HyperAgent's algorithmic simplicity, practical efficiency, and theoretical underpinnings—highlighted by the incremental posterior approximation argument with a novel reduction to sequential random projection (Li, 2024a)—establish HyperAgent as a solution that effectively bridges the gap between theoretical rigor and practical application in reinforcement learning, setting new standards for future RL algorithm design.

Future directions in both practical and theoretical domains are highlighted here. On the practical side, the hypermodel's compatibility with any feedforward neural network architecture offers seamless integration into a wide array of deep reinforcement learning frameworks, including actor-critic structures and transformer-based large models. This flexibility enhances its utility across various applications, such as foundation models, large language models (LLMs), and vision-language models (VLMs). Exploring these integrations could yield significant advancements. Theoretically, the prospect of extending our analysis to include linear, generalized linear, and neural function approximations with stochastic gradient descent (SGD) updates opens up a promising field for future studies. This exploration could deepen our understanding of the underlying mechanisms and improve the model's efficacy and applicability in complex scenarios, further bridging the gap.

The study of scalable posterior inference and uncertainty estimation for both exploration and alignment is of great importance. First, utilizing hypermodel for uncertainty-aware reward modeling is promising for mitigating reward hacking in offline alignment problem and facilitate active feedback query in online alignment problem. Second, it is possible to derive both more efficient alignment algorithm via HyperAgent, performing approximate posterior sampling over $Q^\star$. Extending large foundation models to solve multi-stage sequential decision tasks is also promising.

Acknowledgements

The author would like to thank David Janz for reviewing the manuscript and providing feedback on writing. The work of Y. Li was supported by the Internal Project Fund from Shenzhen Research Institute of Big Data under Grants J00220240001. The work of Z.-Q. Luo was supported by the Guangdong Major Project of Basic and Applied Basic Research (No.2023B0303000001), the Guangdong Provincial Key Laboratory of Big Data Computing, and the National Key Research and Development Project under grant 2022YFA1003900.

Impact statement

HyperAgent represents a major advancement in reinforcement learning (RL). Its applications span gaming, autonomous vehicles, robotics, healthcare, financial trading, energy production, and more. With its computation- and data-efficient design, businesses can use HyperAgent for real-time decision-making, optimizing efficiency and outcomes under resource constraints.

Education and research in machine learning (ML) will benefit from HyperAgent. Its simplicity allows easy implementation, facilitating learning and experimentation for researchers and students, and accelerating advancements in the field. This tool could become vital in academic research and corporate R&D, driving discoveries and breakthroughs in AI. The ease of implementation can democratize access to RL algorithms, fostering innovation and growth.

Smaller organizations and startups, often limited by computing resources, could leverage HyperAgent's scalability and performance, creating a level playing field and encouraging creative innovation.

However, ethical considerations are crucial. The handling of large-scale interaction data requires robust policies to ensure user privacy and data protection. Efficiently managing large data sets heightens privacy concerns. Autonomous decision-making with RL must be monitored to prevent harmful behavior. Ethical implementation and continuous monitoring are essential to ensure fairness, safety, and security.

References

Agarwal, A., Jin, Y., and Zhang, T. VOQL: Towards Optimal Regret in Model-free RL with Nonlinear Function Approximation. In Neu, G. and Rosasco, L. (eds.), *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195 of *Proceedings of Machine Learning Research*, pp. 987–1063. PMLR, 12–15 Jul 2023. URL https://proceedings.mlr.press/v195/agarwal23a.html.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/f514cec81cb148559cf475e7426eed5e-Paper.pdf.

Aravindan, S. and Lee, W. S. State-Aware Variational Thompson Sampling for Deep Q-Networks. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 124–132, 2021.

Azar, M. G., Osband, I., and Munos, R. Minimax Regret Bounds for Reinforcement Learning. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 263–272. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/azar17a.html.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying Count-Based Exploration and Intrinsic Motivation. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/afda332245e2af431fb7b672a68b659d-Paper.pdf.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 449–458. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/bellemare17a.html.

Bertsekas, D. P. and Tsitsiklis, J. N. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition, 1996. ISBN 1886529108.

Dann, C., Mansour, Y., Mohri, M., Sekhari, A., and Sridharan, K. Guarantees for Epsilon-Greedy Reinforcement Learning with Function Approximation. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato,

S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 4666–4689. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/dann22a.html.

Doucet, A. A note on efficient conditional simulation of Gaussian distributions. *Departments of Computer Science and Statistics, University of British Columbia*, 1020, 2010.

Du, S., Kakade, S., Lee, J., Lovett, S., Mahajan, G., Sun, W., and Wang, R. Bilinear Classes: A Structural Framework for Provable Generalization in RL. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2826–2836. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/du21a.html.

Dwaracherla, V. and Van Roy, B. Langevin DQN, 2021.

Dwaracherla, V., Lu, X., Ibrahimi, M., Osband, I., Wen, Z., and Van Roy, B. Hypermodels for Exploration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryx6WgStPB.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research*, 6(18):503–556, 2005. URL http://jmlr.org/papers/v6/ernst05a.html.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. Noisy Networks For Exploration. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rywHCPkAW.

Foster, D. J., Kakade, S. M., Qian, J., and Rakhlin, A. The Statistical Complexity of Interactive Decision Making, 2023.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering Diverse Domains through World Models, 2024.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Hoffman, Y. and Ribak, E. Constrained realizations of Gaussian fields-A simple algorithm. *Astrophysical Journal, Part 2-Letters (ISSN 0004-637X), vol. 380, Oct. 10, 1991, p. L5-L8.*, 380:L5–L8, 1991.

Ishfaq, H., Cui, Q., Nguyen, V., Ayoub, A., Yang, Z., Wang, Z., Precup, D., and Yang, L. Randomized Exploration in Reinforcement Learning with General Value Function Approximation. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4607–4616. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/ishfaq21a.html.

Ishfaq, H., Lan, Q., Xu, P., Mahmood, A. R., Precup, D., Anandkumar, A., and Azizzadenesheli, K. Provable and Practical: Efficient Exploration in Reinforcement Learning via Langevin Monte Carlo. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=nfIAEJFiBZ.

Jaksch, T., Ortner, R., and Auer, P. Near-optimal Regret Bounds for Reinforcement Learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

Jiang, N., Krishnamurthy, A., Agarwal, A., Langford, J., and Schapire, R. E. Contextual Decision Processes with low Bellman rank are PAC-Learnable. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1704–1713. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/jiang17c.html.

Jin, C., Yang, Z., Wang, Z., and Jordan, M. I. Provably efficient reinforcement learning with linear function approximation. In Abernethy, J. and Agarwal, S. (eds.), *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pp. 2137–2143. PMLR, 09–12 Jul 2020. URL https://proceedings.mlr.press/v125/jin20a.html.

Jin, C., Liu, Q., and Miryoosefi, S. Bellman Eluder Dimension: New Rich Classes of RL Problems, and Sample-Efficient Algorithms. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 13406–13418. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/6f5e4e86a87220e5d361ad82f1ebc335-Paper.pdf.

Johnson, W. B. and Lindenstrauss, J. Extensions of Lipschitz mappings into a Hilbert space. In *Conference on Modern Analysis and Probability*, volume 26, pp. 189–206. American Mathematical Society, 1984.

Journel, A. G. and Huijbregts, C. J. Mining geostatistics. 1976.

Kakade, S. M. *On the Sample Complexity of Reinforcement Learning*. University of London, University College London (United Kingdom), 2003.

Kearns, M. and Singh, S. Near-Optimal Reinforcement Learning in Polynomial Time. *Machine Learning*, 49(2):209–232, 2002. doi: 10.1023/A:1017984413808. URL https://doi.org/10.1023/A:1017984413808.

Lai, T. L. and Robbins, H. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1): 4–22, 1985.

Laskin, M., Srinivas, A., and Abbeel, P. CURL: Contrastive unsupervised representations for reinforcement learning. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5639–5650. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/laskin20a.html.

Li, Y. Probability Tools for Sequential Random Projection, 2024a. URL https://arxiv.org/abs/2402.14026.

Li, Y. Simple, unified analysis of Johnson-Lindenstrauss with applications, 2024b. URL https://arxiv.org/abs/2402.10232.

Li, Z., Li, Y., Zhang, Y., Zhang, T., and Luo, Z.-Q. HyperDQN: A Randomized Exploration Method for Deep Reinforcement Learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=X0nrKAXu7g-.

Liang, H. and Luo, Z.-Q. Bridging Distributional and Risk-sensitive Reinforcement Learning with Provable Regret Bounds, 2024.

Liu, H. and Abbeel, P. APS: Active Pretraining with Successor Features. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6736–6747. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/liu21b.html.

Liu, Z., Lu, M., Xiong, W., Zhong, H., Hu, H., Zhang, S., Zheng, S., Yang, Z., and Wang, Z. Maximize to Explore: One Objective Function Fusing Estimation, Planning, and Exploration. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=A57UMlUJdc.

Lu, X. and Van Roy, B. Information-Theoretic Confidence Bounds for Reinforcement Learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/411ae1bf081d1674ca6091f8c59a266f-Paper.pdf.

Lu, X., Van Roy, B., Dwaracherla, V., Ibrahimi, M., Osband, I., and Wen, Z. Reinforcement Learning, Bit by Bit. *Foundations and Trends® in Machine Learning*, 16(6):733–865, 2023. ISSN 1935-8237. doi: 10.1561/2200000097. URL http://dx.doi.org/10.1561/2200000097.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, 2015. doi: 10.1038/nature14236. URL https://doi.org/10.1038/nature14236.

Nikishin, E., Schwarzer, M., D'Oro, P., Bacon, P.-L., and Courville, A. The primacy bias in deep reinforcement learning. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16828–16847. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/nikishin22a.html.

Osband, I. and Van Roy, B. Bootstrapped Thompson Sampling and Deep Exploration, 2015.

Osband, I. and Van Roy, B. Why is Posterior Sampling Better than Optimism for Reinforcement Learning? In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2701–2710. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/osband17a.html.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep Exploration via Bootstrapped DQN. *Advances in neural information processing systems*, 29, 2016a.

Osband, I., Van Roy, B., and Wen, Z. Generalization and Exploration via Randomized Value Functions. In *International Conference on Machine Learning*, pp. 2377–2386. PMLR, 2016b.

Osband, I., Aslanides, J., and Cassirer, A. Randomized prior functions for deep reinforcement learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/5a7b238ba0f6502e5d6be14424b20ded-Paper.pdf.

Osband, I., Van Roy, B., Russo, D. J., and Wen, Z. Deep Exploration via Randomized Value Functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019. URL http://jmlr.org/papers/v20/18-339.html.

Osband, I., Doron, Y., Hessel, M., Aslanides, J., Sezener, E., Saraiva, A., McKinney, K., Lattimore, T., Szepesvari, C., Singh, S., Van Roy, B., Sutton, R., Silver, D., and van Hasselt, H. Behaviour Suite for Reinforcement Learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygf-kSYwH.

Osband, I., Wen, Z., Asghari, S. M., Dwaracherla, V., Ibrahimi, M., Lu, X., and Van Roy, B. Approximate Thompson Sampling via Epistemic Neural Networks. In *The 39th Conference on Uncertainty in Artificial Intelligence*, 2023a. URL https://openreview.net/forum?id=xampQmrqD8U.

Osband, I., Wen, Z., Asghari, S. M., Dwaracherla, V., Ibrahimi, M., Lu, X., and Van Roy, B. Epistemic Neural Networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b. URL https://openreview.net/forum?id=dZqcC1qCmB.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. Parameter Space Noise for Exploration. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ByBAl2eAZ.

Qin, C., Wen, Z., Lu, X., and Van Roy, B. An Analysis of Ensemble Sampling. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=c6ibx0yl-aG.

Quan, J. and Ostrovski, G. DQN Zoo: Reference implementations of DQN-based agents, 2020. URL http://github.com/deepmind/dqn_zoo.

Russo, D. and Van Roy, B. Learning to optimize via information-directed sampling. *Operations Research*, 66(1):230–252, 2018. doi: 10.1287/opre.2017.1663. URL https://doi.org/10.1287/opre.2017.1663.

Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., and Wen, Z. A Tutorial on Thompson Sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018. ISSN 1935-8237. doi: 10.1561/2200000070. URL http://dx.doi.org/10.1561/2200000070.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020. doi: 10.1038/s41586-020-03051-4. URL https://doi.org/10.1038/s41586-020-03051-4.

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uCQfPZwRaUu.

Schwarzer, M., Ceron, J. S. O., Courville, A., Bellemare, M. G., Agarwal, R., and Castro, P. S. Bigger, Better, Faster: Human-level Atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380. PMLR, 2023.

Strehl, A. L. *Probably Approximately Correct (PAC) exploration in reinforcement learning*. PhD thesis, Rutgers University-Graduate School-New Brunswick, 2007.

Strens, M. A Bayesian Framework for Reinforcement Learning. In *International Conference on Machine Learning*, pp. 943–950, 2000.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Thompson, W. R. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25(3/4):285–294, 1933. ISSN 00063444. URL http://www.jstor.org/stable/2332286.

Thrun, S. Efficient Exploration In Reinforcement Learning. Technical Report CMU-CS-92-102, Carnegie Mellon University, Pittsburgh, PA, January 1992.

Tiapkin, D., Belomestny, D., Moulines, É., Naumov, A., Samsonov, S., Tang, Y., Valko, M., and Ménard, P. From Dirichlet to Rubin: Optimistic exploration in RL without bonuses. In *International Conference on Machine Learning*, pp. 21380–21431. PMLR, 2022.

van Hasselt, H., Guez, A., and Silver, D. Deep Reinforcement Learning with Double Q-Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Mar. 2016. doi: 10.1609/aaai.v30i1.10295. URL https://ojs.aaai.org/index.php/AAAI/article/view/10295.

van Hasselt, H. P., Hessel, M., and Aslanides, J. When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32, 2019.

Wang, R., Salakhutdinov, R. R., and Yang, L. Reinforcement Learning with General Value Function Approximation: Provably Efficient Approach via Bounded Eluder Dimension. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6123–6135. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/440924c5948e05070663f88e69e8242b-Paper.pdf.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling Network Architectures for Deep Reinforcement Learning. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1995–2003, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/wangf16.html.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8:279–292, 1992.

Welling, M. and Teh, Y. W. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In Getoor, L. and Scheffer, T. (eds.), *ICML*, pp. 681–688. Omnipress, 2011. URL http://dblp.uni-trier.de/db/conf/icml/icml2011.html#WellingT11.

Wen, Z. *Efficient Reinforcement Learning with Value Function Generalization*. PhD thesis, Stanford University, Stanford, CA, USA, 2014. AAI28121065.

Wilson, J., Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. Efficiently sampling functions from Gaussian process posteriors. In *International Conference on Machine Learning*, pp. 10292–10302. PMLR, 2020.

Xu, P., Zheng, H., Mazumdar, E. V., Azizzadenesheli, K., and Anandkumar, A. Langevin Monte Carlo for Contextual Bandits. In *International Conference on Machine Learning*, pp. 24830–24850. PMLR, 2022.

Ye, W., Liu, S., Kurutach, T., Abbeel, P., and Gao, Y. Mastering Atari Games with Limited Data. *Advances in Neural Information Processing Systems*, 34:25476–25488, 2021.

Łukasz Kaiser, Babaeizadeh, M., Miłos, P., Osiński, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G., and Michalewski, H. Model Based Reinforcement Learning for Atari. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1xCPJHtDB.

A. Additional discussion on related works

Our work represents sustained and focused efforts towards developing principled RL algorithms that are practically efficient with function approximation in complex environments.

Discussion on the algorithmic simplicity and deployment efficiency. To address data efficiency, recent deep RL algorithms have incorporated increasingly complex heuristic and algorithmic components, such as DDQN (van Hasselt et al., 2016), Rainbow (Hessel et al., 2018), EfficientZero (Ye et al., 2021), and BBF (Schwarzer et al., 2023). Furthermore, their practical efficiency often falls short due to high per-step computational costs, exemplified by BBF's use of larger networks and more complex components that require careful tuning and may challenge deployment in real-world settings. Several works (Hessel et al., 2018; van Hasselt et al., 2019; Schwarzer et al., 2023) employ a combination of techniques, including dueling networks (Wang et al., 2016), reset strategy (Nikishin et al., 2022), distributional RL (Bellemare et al., 2017), and others, to achieve data efficiency. Some have demonstrated remarkable performance on Atari games. However, integrating multiple techniques makes the algorithms complicated and challenging to apply to various scenarios. It requires careful selection of hyperparameters for each technique. For example, the reset frequency in the reset strategy needs meticulous consideration. Furthermore, combining multiple techniques results in a significant computational cost. For instance, BBF (Schwarzer et al., 2023) designs a larger network with 15 convolutional layers, which has 20 times more parameters than our method. Model-based RL (Łukasz Kaiser et al., 2020; Schrittwieser et al., 2020; Ye et al., 2021; Hafner et al., 2024) is a widely used approach for achieving data efficiency. However, the performance of these methods is contingent upon the accuracy of the learned predictive model. Furthermore, the learning of predictive models can incur higher computational costs, and employing tree-based search methods with predictive models may not offer sufficient exploration. Other methods (Schwarzer et al., 2021; Laskin et al., 2020; Liu & Abbeel, 2021) achieve data efficiency by enhancing representation learning. While these approaches perform well in environments with image-based states, they show poorer performance in environments characterized by simple structure yet requiring deep exploration, as seen in DeepSea.

Algorithm	Components
DDQN	Incremental SGD with experience replay and target network
Rainbow	(DDQN) + Prioritized replay, Dueling networks, Distributional RL, Noisy Nets.
BBF	(DDQN) + Prioritized replay, Dueling networks, Distributional RL, Self-Prediction, Harder resets, Larger network, Annealing hyper-parameters.
HyperAgent	(DDQN) + **hypermodel**

Table 3. The extra techniques used in different algorithms, e.g. DDQN (van Hasselt et al., 2016), Rainbow (Hessel et al., 2018), BBF (Schwarzer et al., 2023) and HyperAgent.

Other Principled Exploration Approaches. Exploration strategies such as "Optimism in the Face of Uncertainty" (OFU) (Lai & Robbins, 1985) and Information-Directed Sampling (IDS) (Russo & Van Roy, 2018) also play crucial roles. OFU, efficient in tabular settings, encompasses strategies from explicit exploration in unknown states (E^3, Kearns & Singh (2002)) to bonus-based and bonus-free optimistic exploration (Jaksch et al., 2010; Tiapkin et al., 2022; Liang & Luo, 2024). However, OFU also encounter computational hurdles in RL with general function approximation, leading to either intractability or unsustainable resource demands as data accumulates (Jiang et al., 2017; Jin et al., 2021; Du et al., 2021; Foster et al., 2023; Liu et al., 2023; Wang et al., 2020; Agarwal et al., 2023). IDS, while statistically advantageous and tractable in multi-armed and linear bandits, lacks feasible solutions for RL problems in tabular settings (Russo & Van Roy, 2018).

Ensemble-based methods. Osband & Van Roy (2015); Osband et al. (2016a) initiated the bootstrapped ensemble methods, an incremental version of randomized value functions (Wen, 2014), on bandit and deep RL, maintaining an ensemble of point estimates, each being incremental updated. This algorithm design methodology avoid refit a potentially complex model from scratch in the online interactive decision problems. Bayes-UCBVI (Tiapkin et al., 2022) was extended to Incre-Bayes-UCBVI (Tiapkin et al., 2022) using the exact same idea as Algorithm 5 in (Osband & Van Roy, 2015) and then extended to Bayes-UCBDQN (Tiapkin et al., 2022) following BootDQN (Osband et al., 2016a). As reported by the author, Bayes-UCBDQN shares very similar performance as BootDQN (Osband et al., 2016a) but requires addition algorithmic module on artificially generated pseudo transitions and pseduo targets, which is environment-dependent and challenging to

tune in practice as mentioned in appendix G.3 of (Tiapkin et al., 2022). Ensemble+ (Osband et al., 2018; 2019) introduces the randomized prior function for controlling the exploration behavior in the initial stages, somewhat similar to optimistic initialization in tabular RL algorithm design, facilitate the deep exploration and data efficiency. This additive prior design principle is further employed in a line of works (Dwaracherla et al., 2020; Li et al., 2022; Osband et al., 2023b;a). For a practical implmentation of LSVI-PHE (Ishfaq et al., 2021), it utilizes the optimistic sampling (OS) with ensemble methods as a heuristic combination: it maintains an ensemble of M value networks $\{Q_i(s,a), i = 1, \ldots, M\}$ and take greedy action according to the maximum value function over M values $Q(s,a) = \max_{i\in[M]} Q_i(s,a)$ for action selection and Q-target computation. As will be discussed in Appendix C.2, we propose another index sampling scheme called optimistic index sampling (OIS). OIS, OS and quantile-based Incre-Bayes-UCBVI are related in a high level, all using multiple randomized value functions to form a optimistic value function with high probability, thus leading to OFU-based principle for deep exploration. Critical distinction exists, compared with ensemble-based OS and Incre-Bayes-UCBVI, OIS is computationally much more friendly[2] due to our continuous reference distribution P_ξ for sampling as many indices as possible to construct randomized value functions.

Theoretical analysis of ensemble based sampling is rare and difficult. As pointed out by the first correct analysis of ensemble sampling for linear bandit problem (Qin et al., 2022), the first analysis of ensemble sampling in 2017 has technical flaws. The results of Qin et al. (2022) show Ensemble sampling, although achieving sublinear regret in (d-dimensional, T-steps) linear bandit problems, requires $\tilde{O}(T)$ per-step computation, which is unsatisfied for a scalable agent with bounded resource. Because of the potential challenges, there is currently no theoretical analysis available for ensemble-based randomized value functions across any class of RL problems.

Langevin Monte-Carlo. Langecin Monte-Carlo (LMC), staring from SGLD (Welling & Teh, 2011), has huge influence in Bayesian deep learning and approximate Bayesian inference. However, as discussed in many literature (Osband et al., 2023b), the computational costs of LMC-based inference are prohibitive in large scale deep learning systems. Recent advances show the application of LMC-based posterior inference for sequence decision making, such as LMCTS (Xu et al., 2022) for contextual bandits as well as LangevinDQN (Dwaracherla & Van Roy, 2021) and LMC-LSVI (Ishfaq et al., 2024) for reinforcement learning. As we will discuss in the following, these LMC-based TS schemes still suffer scalability issues as the per-step computational complexity would grow unbounded with the increasingly large amount of interaction data.

LMCTS (Xu et al., 2022) provides the first regret bound of LMC based TS scheme in (d-dimensional, T-steps) linear bandit problem, showing $\tilde{O}(d^{3/2}\sqrt{T})$ regret bound with $\kappa_t \log(3\sqrt{2dT\log(T^3)})$ inner-loop iteration complexity within time step t. As discussed in (Xu et al., 2022), the conditional number $\kappa_t = O(t)$ in general. For a single iteration in time step t, LMC requires $O(d^2)$ computation for a gradient calculation of loss function: $\nabla L_t(\theta) = 2(V_t\theta - b_t)$ using notations in (Xu et al., 2022); and $O(d)$ computation on noise generation and parameters update (line 5 and 6 in Algorithm 1 (Xu et al., 2022)). Additional dA computation comes from greedy action selection among action set $\mathcal{A}$ by first computing rewards with inner product and selecting the maximum. Therefore, the per-step computation complexity of LMCTS is $\tilde{O}(d^2T + dA)$, which scales polynomially with increasing number of interactions T. LMCTS is not provably scalable under resource constraints.

LMC-LSVI (Ishfaq et al., 2024) applies similar methodologies and analytical tools as LMCTS (Xu et al., 2022), providing $\tilde{O}(d^{3/2}H^{3/2}\sqrt{T})$ regret in the linear MDP (d-dimensional feature mappings and H-horizons and K episodes where $T = KH$). The inner-loop iteration complexity of LMC within one time step (k,h) of episode k is $2\kappa_k \log(4HKd)$. Similarly, $\kappa_k = O(k)$ in general. (1) In the general feature case: For a single iteration in time step (k,h), LMC requires $O(d^2)$ computation cost for the gradient calculation as from equation (6) of (Ishfaq et al., 2024) and $O(d)$ computation cost for noise generation and parameter update. The per-step computational complexity caused by LMC inner-loops is $O(d^2\kappa_k \log(4HKd)) = O(d^2 k \log(HKd))$. Additional per-step computation cost in episode k is $\tilde{O}(d^2Ak)$, coming from LSVI as discussed in (Jin et al., 2020). Therefore, the per-step computation complexity is $\tilde{O}(d^2K\log(HKd) + d^2AK)$, scaling polynomially with increasing number of episodes K. (2) When consider tabular representation where the feature is one-hot vector with $d = SA$, the per-step computational complexity caused by LMC is $O(SAK\log(SAHK))$ as the covariance matrix is diagonal in the tabular setting. The computation cost by LSVI is now $O(S^2A)$ with no dependence on K as we can perform incremental counting and bottom-up dynamic programming in tabular setting. The per-step computational complexity of LMC-LSVI is $O(SAK\log(SAHK) + S^2A)$, still scaling polynomially with increasing number of episodes K. Thus, LMC-LSVI is not provably scalable under resource constraints.

Ishfaq et al. (2024) also extends their LMC-LSVI to deep RL setting, with a combination of Adam optimization techniques,

[2]See detailed descriptions and ablation studies in Appendix C.2.

resulting the AdamLMCDQN. It introduces additional hyper-parameters, such as the bias term and temperature, to tune as shown in Algorithm 2 of (Ishfaq et al., 2024). As discussed in (Ishfaq et al., 2024), AdamLMCDQN is sensitive to the bias term and tuned over a set of hyper-parameters for different Atari environments, showing its deployment difficulty. As shown in Figure 4, our HyperAgent, using a single set of hyper-parameters for all Atari environments, performs much better than AdamLMCDQN (LangevinMC) in all 8 hardest exploration Atari environments.

Heuristics on noise injection. For example, Noisy-Net (Fortunato et al., 2018) learns noisy parameters using gradient descent, whereas (Plappert et al., 2018) added constant Gaussian noise to the parameters of the neural network. SANE (Aravindan & Lee, 2021) is a variational Thompson sampling approximation for DQNs which uses a deep network whose parameters are perturbed by a learned variational noise distribution. Noisy-Net can be regarded as an approximation to the SANE. While Langevin Monte-Carlo may seem simliar to Noisy-Net (Fortunato et al., 2018; Plappert et al., 2018) or SANE (Aravindan & Lee, 2021) due to their random perturbations of neural network weights in state-action value and target value computation, as well as action selection. Critical differences exist, as noisy networks are not ensured to approximate the posterior distribution (Fortunato et al., 2018) and do not achieve deep exploration (Osband et al., 2018). SANE (Aravindan & Lee, 2021) also lacks rigorous guarantees on posterior approximation and deep exploration.

B. Reproducibility

In support of other researchers interested in utilizing our work and authenticating our findings, we offer the implementation of HyperAgent at the link https://github.com/szrlee/HyperAgent. This repository includes all the necessary code to replicate our experimental results and instructions on usage. Following this, we will present the evaluation protocol employed in our experiments and the reproducibility of the compared baselines.

B.1. Evaluation protocol

Protocol on DeepSea. We replicate all experiments on DeeSea using 10 different random seeds. In each experiment run, we set the maximum episode to 10000 and evaluate the agent 100 times for every 1000 interactions to obtain the average return. The experiment can stop early when the average return reaches 0.99, at which point we record the number of interactions used by the agent. We then collect 10 data points (the number of interactions) for a specific problem size N, which are then utilized to generate a scatter plot in Figures 3 and 8 to 15. In Figure 3, we perform linear regression on HyperAgent's data to establish the dashed line and calculate the R^2 value indicating the goodness of fit. In Figures 8 to 15, we calculate the average over 10 data points to construct the polyline.

Protocol on Atari. For the experiments on Atari, our training and evaluation protocol follows the baseline works (Mnih et al., 2015; van Hasselt et al., 2016; Łukasz Kaiser et al., 2020; Li et al., 2022) During the training process, we assess the agent 10 times after every 20,000 interactions to calculate the average return at each checkpoint. This allows us to obtain 20 data points for each game at every checkpoint, as we repeat the process with 20 different random seeds. We then compute the mean and 90% confidence interval range for plotting the learning curve in Figures 4, 20 and 21.

We calculate the best score for each game using the following steps: (1) for each Atari game, the algorithm is performed with 20 different initial random seeds; (2) The program with one particular random seed will produce one best model (the checkpoint with highest average return), leading to 20 different models for each Atari game; (3) We then evaluate all 20 models, each for 200 times; (4) We calculate the average score from these 200 evaluations as the score for each model associated with each seed. (5) Finally, we calculate and report the average score across 20 seeds as the final score for each Atari game. We follow the aforementioned five-step protocol to determine the score for all 55 Atari games outlined in Tables 7 and 9. We then utilize the 20 scores for each of the 26 Atari games to compute the Interquartile Mean (IQM), median, and mean score, along with the 95% confidence interval[3] for our algorithms, as depicted in Tables 2 and 8.

B.2. Reproducibility of baselines

Experiments on DeepSea. We present our HyperModel and ENNDQN implementation and utilize the official HyperDQN implementation[4] to replicate results. We kindly request the implementation of Ensemble+ from the author of Li et al. (2022),

[3]Using the evaluation code from the paper (Agarwal et al., 2021) available at https://github.com/google-research/rliable.

[4]https://github.com/liziniu/HyperDQN

and they employ the repository on behavior suite[5] for reproduction. We perform ablation studies in DeepSea benchmarks for a comprehensive understanding of HyperAgent in Appendix E.

Experiments on Atari. We provide our version of DDQN(ours) and replicated the results using a well-known repository[6] for DER. We obtained the result data for DDQN[†] and Rainbow from DQN Zoo[7]. As they were based on 200M frames, we extracted the initial 20M steps from these results to compare them with HyperAgent. We acquired the official implementation of BBF[8] and EfficientZero[9] from their official repositories, and reached out to the author of Li et al. (2022) for the results data of Ensemble+ with an ensemble size[10] of 10. For the experiments about exploration on Atari (refer to Figure 4 and Figure 21), we utilized the official implementation to replicate the results of SANE[11]. and we obtained the raw result data of Ensemble+[12], AdamLMCDQN and LangevinAdam from official repository of LMC-LSVI[13]. In addition to the results shown in the main article, we also provide fine-grained studies on Atari suite in Appendix F.

B.3. Environment Settings

In this section, we describe the environments used in experiments. We firstly use the DeepSea (Osband et al., 2020; 2019) to demonstrate the exploration efficiency and scalability of HyperAgent. DeepSea (see Figure 5) is a reward-sparse environment that demands deep exploration (Osband et al., 2019). The environment under consideration has a discrete action space consisting of two actions: moving left or right. During each run of the experiment, the action for moving right is randomly sampled from Bernoulli distribution for each row. Specifically, the action variable takes binary values of 1 or 0 for moving right, and the action map is different for each run of the experiment. The agent receives a reward of 0 for moving left, and a penalty of $-(0.01/N)$ for moving right, where N denotes the size of DeepSea. The agent will earn a reward of 1 upon reaching the lower-right corner. The optimal policy for the agent is to learn to move continuously towards the right. The sparse rewards and states presented in this environment effectively showcase the exploration efficiency of HyperAgent without any additional complexity.

For the experiments on the Atari games, we utilized the standard wrapper provided by OpenAI gym. Specifically, we terminated each environment after a maximum of 108K steps without using sticky actions. For further details on the settings used for the Atari games, please refer to Table 4.

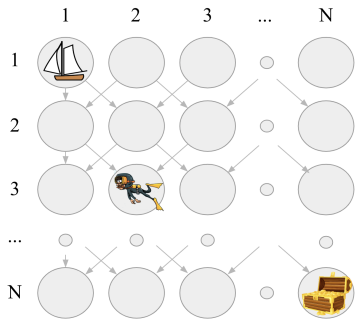


Figure 5. Illustration for DeepSea.

Hyper-parameters	Setting
Grey scaling	True
Sticky action	False
Observation down-sampling	(84, 84)
Frames stacked	4
Action repetitions	4
Reward clipping	[-1, 1]
Terminal on loss of life	True
Max frames per episode	108K

Table 4. Detailed settings for Atari games

[5] https://github.com/google-deepmind/bsuite
[6] https://github.com/Kaixhin/Rainbow
[7] https://github.com/google-deepmind/dqn_zoo
[8] https://github.com/google-research/google-research/tree/master/bigger_better_faster
[9] https://github.com/YeWR/EfficientZero
[10] For guidance on selecting the ensemble size, refer to Appendix C.4 in (Li et al., 2022).
[11] https://github.com/NUS-LID/SANE
[12] It shares the same implementation as Li et al. (2022).
[13] https://github.com/hmishfaq/lmc-lsvi

C. HyperAgent details

In this section, we describe more details of the proposed HyperAgent. First, we describe the general treatment for the incremental update function (in line 11 of HyperAgent) in the following Algorithm 2 of Appendix C.1. Then, we provide the details of index sampling schemes in Appendix C.2. Next, in Appendix C.3, we provide the implementation details of HyperAgent with deep neural network (DNN) function approximation. We want to emphasize that all experiments done in this article is using **Option** (1) with DNN value function approximation. In Appendix C.4, we describe the closed-form update rule (**Option** (2)) when the tabular representation of the value function is exploited. Note that the tabular version of HyperAgent is only for the clarity of analysis and understanding.

C.1. Incremental update mechanism of HyperAgent

Algorithm 2 update

Input: buffer D, $\theta, \theta^-, \boldsymbol{\xi}^-$, agent step t, train step j
if $t \mod \text{training\_freq} = 0$ **then**
  repeat
    Obtain θ by optimizing the loss $L^{\gamma,\sigma,\beta}(\theta; \theta^-, \boldsymbol{\xi}^-, D)$ in Equation (4):
    – Option (1) with gradient descent w.r.t. the mini-batch sampled loss Equation (5); (HyperAgent)
    – Option (2) with closed-form update in Equations (6) to (8). (Tabular HyperAgent)
    Increment $j \leftarrow j + 1$
    if $(j \mod \text{target\_update\_freq}) = 0$ **then**
      $\theta^- \leftarrow \theta$
    end if
  until $(j \mod \text{sample\_update\_ratio} \times \text{training\_freq}) = 0$
end if
Return: θ, θ^-, j.

Notice that in update, there are three important hyper-parameters (target_update_freq, sample_update_ratio, training_freq), which we will specify in Table 5 the hyper-paramters for practical implementation of HyperAgent with DNN function approximation for all experimental studies; and in Table 6 the hyper-parameters only for regret analysis in finite-horizon tabular RL with fixed horizon H. To highlight, We have not seen this level of unification of algorithmic update rules between practice and theoretical analysis in literature!

Hyper-parameters	Atari Setting	DeepSea Setting
weight decay β	0.01	0
discount factor γ	0.99	0.99
learning rate	0.001	0.001
mini-batch size $\lvert\tilde{D}\rvert$	32	128
index dim M	4	4
# Indices $\lvert\tilde{\Xi}\rvert$ for approximation	20	20
Perturbation std. σ	0.01	0.0001
n-step target	5	1
target_update_freq in update	5	4
sample_update_ratio in update	1	1
training_freq in update	1	1
hidden units	256	64
min replay size for sampling	2,000 steps	128 steps
memory size	500,000 steps	1000000 steps

Table 5. Hyper-parameters of HyperAgent. Other hyper-parameters used for Atari suite are the same as Rainbow (Hessel et al., 2018). Note that we utilize a **single configuration** for all 55 games from Atari suite and a **single configuration** for DeepSea with varying sizes.

For the approximation of expectation in Equation (4), we sample multiple indices for each transition tuple in the mini-batch

and compute the empirical average, as described in Equation (5). Recall that $|\tilde{\Xi}|$ is the number of indices for each state and we set $|\tilde{\Xi}| = 20$ as default setting. We have demonstrated how the number of indices $|\tilde{\Xi}|$ impacts our method in Appendix E.2.

C.2. Index sampling schemes of HyperAgent

Let us review the loss function. For a transition tuple $d = (s, a, r, s', \mathbf{z}) \in D$ and given index ξ, the perturbed temporal difference (TD) loss $\ell^{\gamma,\sigma}(\theta; \theta^-, \boldsymbol{\xi}^-, \xi, d)$ is

$$\left(r + \sigma \xi^\top \mathbf{z} + \gamma \max_{a' \in \mathcal{A}} f_{\theta^-}(s', a', \boldsymbol{\xi}^-(s')) - f_\theta(s, a, \xi)\right)^2, \tag{11}$$

where θ^- is the target parameters, and σ is a hyperparameter to control the std of algorithmic perturbation.

We have two options for index sampling schemes for $\boldsymbol{\xi}_k(s)$:

1. **State-dependent sampling.** As for implementation, especially for continuous or uncountable infinite state space: in the interaction, $\boldsymbol{\xi}_k(s)$ in the line 7 of HyperAgent is implemented as independently sampling $\xi \sim P_\xi$ for each encountered state; for the target computation in Equation (11), $\boldsymbol{\xi}_k(s')$ is implemented as independently sampling $\xi \sim P_\xi$ for each tuple $d = (s, a, r, s', \mathbf{z})$ in the every sampled mini-batch.
2. **State-independent sampling.** The implementation of state-independent $\boldsymbol{\xi}_k(s) = \xi_k$ is straightforward as we independently sample ξ_k in the beginning of each episode k and use the same ξ_k for each state s encountered in the interaction and for each target state s' in target computation.

In our implementation by default, HyperAgent employs the state-independent $\boldsymbol{\xi}$ for action selection and utilizes state-dependent $\boldsymbol{\xi}$ for Q-target computation. The ablation results in Appendix E.1 demonstrate that these distinct index sampling schemes for $\boldsymbol{\xi}$ yield nearly identical performance.

Optimistic index sampling. To make agent's behavior more optimistic with more aggresive deep exploration, in each episode k, we can sample N_{OIS} indices $\xi_{k,1}, \ldots, \xi_{k,N_{\text{OIS}}}$ and take the greedy action according to the associated hypermodel:

$$a_k = \arg\max_{a \in \mathcal{A}} \max_{n \in [N_{\text{OIS}}]} f_\theta(s_k, a, \xi_{k,n}), \tag{12}$$

which we call optimistic index sampling (OIS) action selection scheme.

In the hypermodel training part, for any transition tuple $d = (s, a, r, s', \mathbf{z})$, we also sample multiple indices $\xi_1^-, \ldots, \xi_{N_{\text{OIS}}}^-$ independently and modify the target computation in Equation (11) as

$$r + \sigma \xi^\top \mathbf{z} + \gamma \max_{a' \in \mathcal{A}} \max_{n \in [N_{\text{OIS}}]} f_{\theta^-}(s', a', \xi_n^-(s')). \tag{13}$$

This modification in target computation boosts the propagation of uncertainty estimates from future states to earlier states, which is beneficial for deep exploration. We call this variant HyperAgent w. OIS and compare it with HyperAgent in Appendix E.2. HyperAgent w, OIS can outperform HyperAgent, and the OIS method incurs minimal additional computation, as we have set $M = 4$ and $N_{\text{OIS}} = 5$ in empirical studies. Theoretically, leveraging this optimistic value estimation with OFU-based regret analysis, e.g. UCBVI-CH in (Azar et al., 2017), could lead a $O(H^2\sqrt{SAK})$ frequentist regret bound in finite-horizon time-inhomogeneous RL (Assumption D.7) **without** using Assumption D.8.

C.3. Function approximation with deep neural networks

Here we describe the implementation details of HyperAgent with deep neural networks and the main difference compared to baselines.

C.3.1. HYPERMODEL ARCHITECTURE IN HyperAgent

First, we develop a hypermodel for efficient approximate the posterior over the action-value function under neural network function approximation. As illustrated in Figure 2, we made assumptions that (1) **Base-model**: the action-value function is

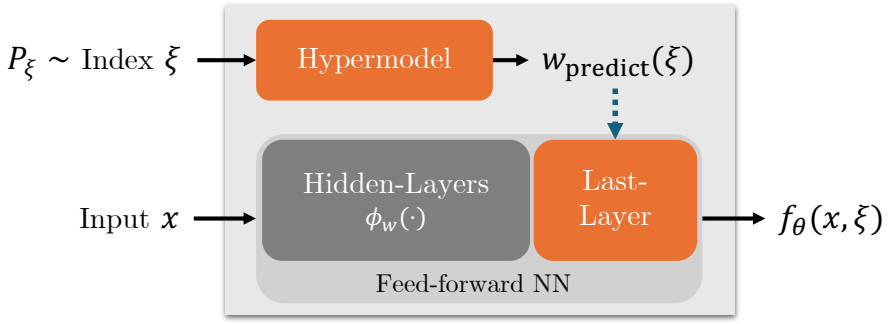


Figure 6. (Figure 2 restated.) Description of the last-layer linear hypermodel: we made an assumption that the injected randomness only from the linear layer is sufficient for uncertainty estimation of feed-forward neural networks.

linear in the feature space even when the feature is unknown and needs to be learned through the training of neural network hidden layers; and (2) **Last-layer linear hypermodel**: the degree of uncertainty for base-model can be represented by a linear hypermodel transforming the index distribution to the approximation posterior distribution over the last-layer; and can be used for efficient deep exploration.

The (1) base-model assumption is common in supervised learning and deep reinforcement learning, e.g. DDQN(Mnih et al., 2015; van Hasselt et al., 2016), BBF(Schwarzer et al., 2023).

As the explanation of the (2) last-layer linear hypermodel assumption: for example, in Figure 2, suppose the hidden layers in neural networks forms the nonlinear feature mapping $\phi_w(\cdot)$ with parameters w. Our last-layer linear hypermodel assumption is formulated in Equation (14), with trainable $\theta = \{\boldsymbol{A}, b, w\}$ and fixed parameters $\{\boldsymbol{A}_0, b_0, w_0\}$, taking the random index $\xi \in \mathbb{R}^M$ from reference distribution P_ξ as input and outputs the weights for last-layer.

$$
\begin{aligned}
f_\theta(x, \xi) &= \underbrace{\langle \boldsymbol{A}\xi + b, \phi_w(x) \rangle}_{\textbf{Learnable } f^L_\theta(x,\xi)} + \underbrace{\langle \boldsymbol{A}_0\xi + b_0, \phi_{w_0}(x) \rangle}_{\textbf{Fixed prior } f^P(x,\xi)} \\
&= \underbrace{\langle \boldsymbol{A}\xi, \phi_w(x) \rangle}_{\sigma^L_\theta(x,\xi)} + \underbrace{\langle \boldsymbol{A}_0\xi, \phi_{w_0}(x) \rangle}_{\sigma^P(x,\xi)} + \underbrace{\langle b, \phi_w(x) \rangle}_{\mu^L_\theta(x)} + \underbrace{\langle b_0, \phi_w(x) \rangle}_{\mu^P(x)}.
\end{aligned} \tag{14}
$$

It's worth to note that our hypermodel only outputs the weights but not bias for last-layer.

(2) As shown in Lemma 4.1, we validate that the linear hypermodel with incremental update can approximate the posterior of action-value function in the sequential decision processes. We conjecture that last layer linear hypermodel assumption is reasonable under neural network function approximation. Through our formulation in Equation (14), `HyperAgent` is supposed to accurately estimate the learnable mean $\mu^L_\theta(x)$, which relies solely on the original input x, and the variation prediction $\sigma^L_\theta(x, \xi)$, which is dependent on both the original input x and random index ξ. Since not being influenced by other components that may only depend on the random index ξ like HyperDQN (Li et al., 2022), we conjecture our last-layer linear hypermodel assumption in Equation (14) allows the hypermodel to capture uncertainty better. Another benefit last-layer linear hypermodel is that this structure will not result in much parameters and provide better expectation estimate.

The fixed prior model also offers prior bias and prior variation through the functions $\mu^P(x)$ and $\sigma^P(x, \xi)$. This prior function is *not* trainable so that it will not bring much computation, and designed to provide better exploration in the early stage of training. We use Xavier normal initialization for the entire network except for the prior model. For the initialization of prior model, we follow the method described in (Li et al., 2022; Dwaracherla et al., 2020). In this way, each row of prior function is sampled from the unit hypersphere, which guarantees that the output of prior function can follow a desired Gaussian distribution.

In the context of reinforcement learning, we define the action-value function with hypermodel and DNN approximation as following. For each action $a \in \mathcal{A}$, there is a set of trainable parameters $\{\mathbf{A}^{(a)}, b^{(a)}\}$ and fixed parameters $\{\mathbf{A}_0^{(a)}, b_0^{(a)}\}$, i.e., the trainable set of parameters $\theta = \{w, (\mathbf{A}^{(a)}, b^{(a)}) : a \in \mathcal{A}\}$ and the fixed one $\{w_0, (\mathbf{A}_0^{(a)}, b_0^{(a)}) : a \in \mathcal{A}\}$ with

action-value function

$$f_\theta(s,a,\xi) = \underbrace{\langle \mathbf{A}^{(a)}\xi + b^{(a)}, \phi_w(s)\rangle}_{\textbf{Learnable } f_\theta^L(s,a,\xi)} + \underbrace{\langle \mathbf{A}_0^{(a)}\xi + b_0^{(a)}, \phi_{w_0}(s)\rangle}_{\textbf{Fixed prior } f^P(s,a,\xi)}.$$

The last-layer linear hypermodel assumption is further supported by the empirical results Figures 3 and 4 where hypermodel with incremental updates enables efficient deep exploration in RL.

C.3.2. Difference compared to prior works

Several related work can be included in the hypermodel framework introduced in Section 2.1. We will discuss the structural and algorithmic differences under the unified framework in this sections. Furthermore, we performs ablation studies concerning these mentioned differences in Appendix E.3.

Difference with HyperModel (Dwaracherla et al., 2020). HyperModel (Dwaracherla et al., 2020) employs hypermodel to represent epistemic uncertainty and facilitate exploration in bandit problem. However, implementing hypermodel across entire based-model results in a significant number of parameters and optimization challenges. As a reuslts, applying HyperModel to tackle large-scale problems, like Atari games, can prove to be exceedingly difficult, as highlighted in Li et al. (2022). Additionally, HyperModel also encounters challenges in addressing the DeepSea problem due to its substantial state space, even when the size is relatively small, as demonstrated in Appendix E.3. In contrast, `HyperAgent` offers the advantage of computation efficiency as it only applies hypermodel to the last output layer of base-model, which maintains constant parameters when scaling up the problem. `HyperAgent` aslo demonstrates superior data efficiency compared to HyperModel, as evidenced in Figure 14.

Structural difference with Ensemble+ (Osband et al., 2018; 2019). Ensemble+ applies the bootstrapped ensemble method to the entire based-model, which maintains an ensemble of M value networks $\{Q_i(s,a), i = 1,\dots,M\}$. When combined with prior network, it has demonstrated effective exploration in chain environments with large sizes. Nevertheless, to achieve effective exploration, Ensemble+ demands a relatively large ensemble size M (Osband et al., 2018), which raises challenges analogous to those faced by the HyperModel. This involves managing numerous parameters and optimization issues. We also evaluated Ensemble+ with a larger $M = 16$, and it still proved ineffective in solving the DeepSea, as depicted in Figure 10, highlighting the superior data efficiency of `HyperAgent`.

Structural difference with hypermodel in HyperDQN (Li et al., 2022). HyperDQN shares a similar structure with `HyperAgent` and has demonstrated promising results in exploration. Nevertheless, it struggles to handle the DeepSea, which requires deep exploration, as depicted in Figure 3. We enhanced HyperDQN by simplifying the hypermodel, as demonstrated in Equation (14). `HyperAgent` estimates the mean μ exclusively from the original input x and estimates the variation σ using both the original input x and the random index $\boldsymbol{\xi}$. In the implementation of HyperDQN, there are two linear hypermodel $f_{\theta_1}(\xi)$ and $f_{\theta_2}(\xi)$.

$$\begin{aligned} f_{\theta_1}(\xi) &= \boldsymbol{A}_1\xi + b_1; & f_{\theta_2}(\xi) &= \boldsymbol{A}_2\xi + b_2. \\ f_{\theta_1^P}(\xi) &= \boldsymbol{A}_1^P\xi + b_1^P; & f_{\theta_2^P}(\xi) &= \boldsymbol{A}_2^P\xi + b_2^P. \end{aligned}$$

The hypermodel $f_{\theta_1}(\xi)$ outputs weights for last output layer of base-model, and hypermodel $f_{\theta_2}(\xi)$ outputs bias for last output layer of base-model. The functions $f_{\theta_1^P}(\xi)$ and $f_{\theta_2^P}(\xi)$ are the prior network corresponding to trainable linear hypermodel. These linear hypermodel contain trainable $\theta_1 = \{\boldsymbol{A}_1, b_1\}$, $\theta_2 = \{\boldsymbol{A}_2, b_2\}$ and fixed parameters $\theta_1^P = \{\boldsymbol{A}_1^P, b_1^P\}$, $\theta_2^P = \{\boldsymbol{A}_2^P, b_2^P\}$. Therefore, the implementation of HyperDQN can be formulated by

$$\begin{aligned} f^{\text{HyperDQN}}(x,\xi) &= \langle f_{\theta_1}(\xi), \phi_w(x)\rangle + f_{\theta_2}(\xi) + \langle f_{\theta_1^P}(\xi), \phi_w^P(x)\rangle + f_{\theta_2^P}(\xi) \\ &= \underbrace{\langle b_1, \phi_w(x)\rangle}_{\mu^L(x)} + \underbrace{\langle b_1^P, \phi_w^P(x)\rangle}_{\mu^P(x)} + \underbrace{\langle \boldsymbol{A}_1\xi, \phi_w(x)\rangle}_{\sigma_1^L(,\xi)} + \underbrace{\langle \boldsymbol{A}_1^P\xi, \phi_w^P(x)\rangle}_{\sigma_1^P(,\xi)} + \\ &\quad \underbrace{\boldsymbol{A}_2\xi}_{\sigma_2^L(\xi)} + \underbrace{\boldsymbol{A}_2^P\xi}_{\sigma_2^P(\xi)} + \underbrace{b_2}_{\mu_3^L} + \underbrace{b_2^P}_{\mu_3^P}. \end{aligned} \tag{15}$$

As demonstrated in Equation (15), HyperDQN utilizes the hypermodel to generate both weights and bias for the output layer, leading to redundant components, such as functions ($\sigma_2^L(\xi)$, $\sigma_2^P(\xi)$) that rely solely on the random index $\boldsymbol{\xi}$, or functions (μ_3^L, μ_3^P) that do not depend on any inputs. These components lack a clear semantic explanation. We also found that initializing the hypermodel with Xavier Normal can improve optimization. These modifications are some of the factors leading to HyperAgent outperforming HyperDQN on both DeepSea and Atari games, as demonstrated in Section 5.

Structural difference with epinet in ENNDQN (Osband et al., 2023b;a). ENNDQN (Osband et al., 2023a), leveraging the epinet (Osband et al., 2023b) structure, exhibits potential in capturing epistemic uncertainty and has showcased effectiveness across diverse tasks. A notable difference is that ENNDQN use "stop gradient" between feature layers and epinet. This indicates that the error feedback from epinet will not be back-propagated to the feature layers. Another difference is about epinet structure where the original input x, feature $\phi_w(x)$, and random index $\boldsymbol{\xi}$ are concatenated as the input of epinet. An ensemble prior function with size M is used for the output layer, but a separated prior feature network is not present. This network structure results in larger parameters when handling tasks at a large scale, creating notable computation and optimization challenges. For instance, in the case of DeepSea with a size of 20, the parameters of epinet are nearly 20 times larger than those of HyperAgent. This is due to the raw state input $x \in \mathbb{R}^{N^2}$ for DeepSea with size N, whose dimension N^2 is too large for the epinet to effectively process. As as results, epinet struggles with larger scale of the problem, as evidenced in Figure 14. In contrast, HyperAgent takes only a random index $\boldsymbol{\xi}$ and feature $\phi_w(x)$ as input for output layer, resulting in more efficient computation with fewer constant parameters.

Algorithmic difference with Ensemble+, HyperDQN and ENNDQN. For a transition tuple $d = (s, a, r, s', \mathbf{z}) \in D$ and given a single index ξ, with main hypermodel parameters θ and target hypermodel parameters θ^-, the loss in these works can be represented using our notion of hypermodel:

- The temporal difference (TD) loss for Ensemble+ (Osband et al., 2018; 2019) inherits the BootDQN (Osband et al., 2016a),

$$\ell^{\gamma}_{\text{Ensemble+}}(\theta; \theta^-, \xi, d) = (\mathbf{z}^\top \xi) \left(f_\theta^{\text{Ensemble+}}(s, a, \xi) - (r + \gamma \max_{a' \in \mathcal{A}} f_{\theta^-}^{\text{Ensemble+}}(s', a', \xi)) \right)^2, \tag{16}$$

  where $f^{\text{Ensemble+}}$ is the ensemble network structure, $\xi \sim P_\xi := \mathcal{U}(\{e_1, \ldots, e_M\})$ where e_i is the one-hot vector in $\mathbb{R}^M$ and $\mathbf{z} \sim P_\mathbf{z}$, where $\mathbf{z}_i$ sampled from $2 \cdot \text{Bernoulli}(0.5)$ independently across entries $i \in [M]$.

- The TD loss for HyperDQN (Li et al., 2022) is

$$\ell^{\gamma,\sigma}_{\text{HyperDQN}}(\theta; \theta^-, \xi, d) = \left(f_\theta^{\text{HyperDQN}}(s, a, \xi) - (r + \sigma \mathbf{z}^\top \xi \gamma + \max_{a' \in \mathcal{A}} f_{\theta^-}^{\text{HyperDQN}}(s', a', \xi)) \right)^2, \tag{17}$$

  where f^{HyperDQN} is the network structure of HyperDQN, $\xi \sim P_\xi := \mathcal{U}(\mathbb{S}^{M-1})$ and $\mathbf{z} \sim P_\mathbf{z} := \mathcal{U}(\mathbb{S}^{M-1})$.

- The TD loss for ENNDQN (Osband et al., 2023a) is

$$\ell^{\gamma}_{\text{ENNDQN}}(\theta; \theta^-, \xi, d) = \left(f_\theta^{\text{epinet}}(s, a, \xi) - (r + \gamma \max_{a' \in \mathcal{A}} f_{\theta^-}^{\text{epinet}}(s', a', \xi)) \right)^2, \tag{18}$$

  where f^{epinet} is the epinet structure used in ENNDQN, $\xi \sim P_\xi := N(0, I_M)$.

As discussed in this section, $f^{\text{Ensemble+}}, f^{\text{HyperDQN}}, f^{\text{epinet}}$ can all be represented within our hypermodel framework and our construction of hypermodel in HyperAgent has a few mentioned advantages. The key difference in these loss functions Equations (16) to (18) is that the index ξ used for target computation is the same one as in main network; while in Equation (11) of HyperAgent, we choose a index mapping $\boldsymbol{\xi}^-$ in the target computation that is independent of ξ used in main network.

The choice for independent $\boldsymbol{\xi}^-$ is **critical** for theoretical analysis as closed-form solution under setting with fixed feature mapping can be derived with Equation (11) of HyperAgent but can *not* be derived with Equations (16) to (18) of any previous related works. The empirical difference for this issue will be discussed in ablation studies in Appendix E.3. Another issue is that whether we need a additive perturbation. As stated in the incremental update Equation (6), the std of artificial perturbation σ is important for our posterior approximation argument Lemma 4.1. However, in some practical problems with deterministic transitions, do we really need this level of perturbations i.e., $\sigma = 0$ or $\sigma > 0$? If we need $\sigma > 0$, how large should it be? This issue would be address in Appendix E.2.

C.4. Tabular representations

To understand and analyze the behavior of HyperAgent, we specify the algorithm in the tabular setups. Notice that the closed-form iterative update rule derived here is general and can be applied for infinite-horizon and finite horizon problems.

Hypermodel in HyperAgent with tabular representation would be

$$f_\theta(s,a,\xi) = \underbrace{\mu_{sa} + m_{sa}^\top \xi}_{\textbf{Learnable } f_\theta^L(s,a,\xi)} + \underbrace{\mu_{0,sa} + \sigma_0 \mathbf{z}_{0,sa}^\top \xi}_{\textbf{Fixed prior } f^P(s,a,\xi)}$$

where $\theta = (\mu \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}, m \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times M})$ are the parameters to be learned, and $\mathbf{z}_{0,sa} \in \mathbb{R}^M$ is a independent random vector sampled from $P_\mathbf{z}$ and $\mu_{0,sa}, \sigma_0$ is a prior mean and prior variance for each $(s,a) \in \mathcal{S}\times\mathcal{A}$. The tabular representation is related to the last-layer linear hypermodel assumption in Equation (14) when the hidden layer maps each state s to a fixed one-hot feature $\phi_w(s) = \phi_{w_0}(s) = \mathbb{1}_s \in \mathbb{R}^{|\mathcal{S}|}$. The regularizer in Equation (4) then becomes $\beta\|\theta\|^2 = \beta \sum_{s,a} \left(\mu_{sa}^2 + \|m_{sa}\|^2\right)$.

Derivations of Equations (6) to (8) in Section 4. The derivation is mainly from the separability of optimization problem in tabular setup. Let $\theta_{sa} = (\mu_{sa}, m_{sa})$ be the optimization variable for specific $(s,a) \in \mathcal{S}\times\mathcal{A}$. As mentioned in Section 4, at the beginning of episode k, with $D = \mathcal{H}_k$ and target noise mapping $\boldsymbol{\xi}^- = \boldsymbol{\xi}_k$, we iterative solve Equation (4) by taking target parameters $\theta^- = \theta_k^{(i)}$ as previous solved iterate starting from $\theta_k^{(0)} = \theta_{k-1} = (\mu_{k-1}, m_{k-1})$ as the solution for previous episode $k-1$.

Let the optimal solution in (i)-th iteration be $\theta_k^{(i+1)} = \arg\min_\theta L^{\gamma,\sigma,\beta}(\theta; \theta^- = \theta_k^{(i)}, \boldsymbol{\xi}^- = \boldsymbol{\xi}_k, \mathcal{H}_k)$. In tabular setting, by the separability of the objective function in Equation (4), we have $\theta_{k,sa} = (\mu_{k,sa}, m_{k,sa}) = \arg\min_{\theta_{sa}} L_{sa}(\theta_{sa}; \theta^- = \theta_k^{(i)}, \boldsymbol{\xi}^- = \boldsymbol{\xi}_k, \mathcal{H}_k))$ where $\arg\min_{\theta_{sa}} L_{sa}(\theta_{sa}; \theta^-, \boldsymbol{\xi}^-, \mathcal{H}_k)$ is defined as

$$\begin{aligned}
&\arg\min_{\theta_{sa}} \mathbb{E}_{\xi\sim P_\xi}\left[\sum_{\ell=1}^{k-1}\sum_{t\in E_{\ell,sa}} \left(f_\theta(S_{\ell,t},A_{\ell,t},\xi) - (\sigma\xi^\top \mathbf{z}_{\ell,t+1} + y_{\ell,t+1}(\theta^-,\boldsymbol{\xi}^-))\right)^2\right] + \beta(\mu_{sa}^2 + \|m_{sa}\|^2)\\
=&\arg\min_{\theta_{sa}} \mathbb{E}_{\xi\sim P_\xi}\left[\sum_{\ell=1}^{k-1}\sum_{t\in E_{\ell,sa}} \left(f_\theta(s,a,\xi) - (\sigma\xi^\top \mathbf{z}_{\ell,t+1} + y_{\ell,t+1}(\theta^-,\boldsymbol{\xi}^-))\right)^2\right] + \beta(\mu_{sa}^2 + \|m_{sa}\|^2)\\
=&\arg\min_{(\mu_{sa},m_{sa})} \mathbb{E}_{\xi\sim P_\xi}\left[\sum_{\ell=1}^{k-1}\sum_{t\in E_{\ell,sa}} \left((\mu_{sa}+\mu_{0,sa}) + (m_{sa}+\sigma_0\mathbf{z}_{0,sa})^\top\xi - (\sigma\xi^\top \mathbf{z}_{\ell,t+1} + y_{\ell,t+1}(\theta^-,\boldsymbol{\xi}^-))\right)^2\right]\\
&\qquad + \beta(\mu_{sa}^2 + \|m_{sa}\|^2)
\end{aligned}$$

where the target value is

$$y_{\ell,t+1}(\theta^-,\boldsymbol{\xi}^-) = R_{\ell,t+1} + \gamma \max_{a'\in\mathcal{A}} f_{\theta^-}(S_{\ell,t+1}, a', \boldsymbol{\xi}^-(S_{\ell,t+1})).$$

With some calculations, the closed form solution of $\theta_k^{(i+1)} = (\mu_{k,sa}^{(i+1)}, m_{k,sa})$ is

$$\begin{aligned}
m_{k,sa} + \sigma_0\mathbf{z}_{0,sa} &= \frac{\sigma\sum_{\ell=1}^{k-1}\sum_{t\in E_{\ell,sa}} \mathbf{z}_{\ell,t+1} + \beta\sigma_0\mathbf{z}_{0,sa}}{N_{k,sa}+\beta}\\
&= \frac{(N_{k-1,sa}+\beta)(m_{k-1,sa}+\sigma_0\mathbf{z}_{0,sa}) + \sigma\sum_{t\in E_{k-1,sa}}\mathbf{z}_{k-1,t+1}}{N_{k,sa}+\beta},
\end{aligned}$$

which derives the incremental update in Equation (6), and

$$\mu_{k,sa}^{(i+1)} = \frac{\sum_{\ell=1}^{k-1}\sum_{t\in E_{\ell,sa}} y_{\ell,t+1}(\theta^- = \theta_k^{(i)}, \boldsymbol{\xi}^- = \boldsymbol{\xi}_k) + \beta\mu_{0,sa}}{N_{k,sa}+\beta}. \tag{19}$$

Recall some short notations: (1) V_Q is greedy value with respect to Q such that $V_Q(s) = \max_{a\in\mathcal{A}} Q(s,a)$ for all $s\in\mathcal{S}$; (2) $f_{\theta,\boldsymbol{\xi}}(s,a) = f_\theta(s,a,\boldsymbol{\xi}(s)), \forall (s,a)\in\mathcal{S}\times\mathcal{A}$. Recall the stochastic bellman operator F_k^γ induced by HyperAgent,

$$F_k^\gamma Q(s,a) := \frac{\beta\mu_{0,sa} + N_{k,sa}(r_{sa} + \gamma V_Q^\top \hat{P}_{k,sa})}{N_{k,sa}+\beta} + m_{k,sa}^\top \boldsymbol{\xi}_k(s), \quad \forall (s,a)\in\mathcal{S}\times\mathcal{A}.$$

With the following observation,

$$\sum_{\ell=1}^{k-1}\sum_{t\in E_{\ell,sa}} y_{\ell,t+1}(\theta^-,\boldsymbol{\xi}^-) = N_{k,sa}(r_{sa}+\gamma\sum_{s'\in\mathcal{S}}\hat{P}_{k,sa}(s')(\max_{a'\in\mathcal{A}} f_{\theta^-}(s',a',\boldsymbol{\xi}^-(s')))),$$

from Equation (19), we have for all pairs (s,a)

$$f_{\theta_k^{(i+1)}}(s,a,\boldsymbol{\xi}_k) = \mu_{k,sa}^{(i+1)} + m_{k,sa}^\top \boldsymbol{\xi}_k(s) = \frac{\beta\mu_{0,sa} + N_{k,sa}(r_{sa}+\gamma V_{f_{\theta_k^{(i)}},\boldsymbol{\xi}_k}^\top \hat{P}_{k,sa})}{N_{k,sa}+\beta} + m_{k,sa}^\top \boldsymbol{\xi}_k(s),$$

which is essentially bellman iteration under stochastic bellman operator induced by HyperAgent,

$$f_{\theta_k^{(i+1)},\boldsymbol{\xi}_k} = F_k^\gamma f_{\theta_k^{(i)},\boldsymbol{\xi}_k}.$$

Lemma C.1 (Contraction mapping)**.** *Let $B(\mathcal{S}\times\mathcal{A})$ be the space of bounded functions $Q:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$. Let ρ be the distance metric $\rho(Q,Q') = \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}}|Q(s,a)-Q'(s,a)|$. For all $k\in\mathbb{Z}_{++}$ the Bellman operator of* HyperAgent *$F_k^\gamma : B(\mathcal{S}\times\mathcal{A})\to B(\mathcal{S}\times\mathcal{A})$ is a contraction mapping with modulus $\gamma\in[0,1)$ in metric space $(B(\mathcal{S}\times\mathcal{A}),\rho)$.*

By Lemma C.1, since contraction mapping, the bellman operator of HyperAgent F_k^γ has a unique fixed point and the iterative process in Equation (7) can converge to a unique fixed point θ_k. Essentially, due the algorithmic randomness introduced in the iterative process, $f_{\theta_k,\boldsymbol{\xi}_k}$ is a randomized state-action value function.

Proof of Lemma C.1. By Blackwell's sufficient conditions, we need to show that F_k^γ satisfies the following two conditions:

1. Monotonicity: for all $Q,Q'\in B(\mathcal{S}\times\mathcal{A})$, if $Q(s,a)\le Q'(s,a)$ for all $(s,a)\in\mathcal{S}\times\mathcal{A}$, then

$$F_k^\gamma Q(s,a) \le F_k^\gamma Q'(s,a)$$

2. Discounting: for all $(s,a)\in\mathcal{S}\times\mathcal{A}$, all $c\ge 0$ and $Q\in B(\mathcal{S}\times\mathcal{A})$,

$$[F_k^\gamma(Q+c)](s,a) - [F_k^\gamma Q](s,a) = \left(\frac{N_{k,sa}}{N_{k,sa}+\beta}\right)\gamma c \le \gamma c$$

□

True Bellman Operator. For any MDP $\mathcal{M} = (\mathcal{S},\mathcal{A},r,P,\rho,s_{\text{terminal}})$, consider a function $Q\in B(\mathcal{S}\times\mathcal{A})$. The true Bellman operator, when applied to Q, is defined as follows:

$$F_{\mathcal{M}}^\gamma Q(s,a) = r_{sa} + \gamma V_Q^\top P_{sa}, \quad \forall (s,a)\in\mathcal{S}\times\mathcal{A}.$$

As will be introduced later, under Dirichlet prior Assumption D.8, given the randomness in P, $F_{\mathcal{M}}^\gamma$ is essentially stochastic in nature. When $F_{\mathcal{M}}^\gamma$ acts upon a state-action value function $Q\in B(\mathcal{S}\times\mathcal{A})$, the result is a randomized state-action value function.

D. Insight and Theoretical analysis of HyperAgent

D.1. Sequential posterior approximation argument in Lemma 4.1

We use short notation for $[n]=\{1,2,\dots,n\}$ and $\mathcal{T}=\{0,1,\dots,T\}=\{0\}\cup[T]$. Before digging into the details of proof of our key lemma, let us first introduce some useful probability tools developed for random projection recently.

Lemma D.1 (Distributional JL lemma (Johnson & Lindenstrauss, 1984))**.** *For any $0<\varepsilon,\delta<1/2$ and $d\ge 1$ there exists a distribution $\mathcal{D}_{\varepsilon,\delta}$ on $\mathbb{R}^{M\times d}$ for $M = O\left(\varepsilon^{-2}\log(1/\delta)\right)$ such that for any $\boldsymbol{x}\in\mathbb{R}^d$*

$$\mathop{\mathbb{P}}_{\Pi\sim\mathcal{D}_{\varepsilon,\delta}}\left(\|\Pi\boldsymbol{x}\|_2^2 \notin \left[(1-\varepsilon)\|\boldsymbol{x}\|_2^2, (1+\varepsilon)\|\boldsymbol{x}\|_2^2\right]\right) < \delta$$

Theorem D.2 (Sequential random projection in adaptive process (Li, 2024a)). *Let $\varepsilon \in (0,1)$ be fixed and $(\mathcal{F}_t)_{t\geq 0}$ be a filtration. Let $\mathbf{z}_0 \in \mathbb{R}^M$ be an $\mathcal{F}_0$-measurable random vector satisfies $\mathbb{E}\left[\|\mathbf{z}_0\|^2\right] = 1$ and $|\|\mathbf{z}_0\|^2 - 1| \leq (\varepsilon/2)$. Let $(\mathbf{z}_t)_{t\geq 1} \subset \mathbb{R}^M$ be a stochastic process adapted to filtration $(\mathcal{F}_t)_{t\geq 1}$ such that it is $\sqrt{c_0/M}$-sub-Gaussian and each $\mathbf{z}_t$ is unit-norm. Let $(x_t)_{t\geq 1} \subset \mathbb{R}$ be a stochastic process adapted to filtration $(\mathcal{F}_{t-1})_{t\geq 1}$ such that it is c_x-bounded. Here, c_0 and c_x are absolute constants. For any fixed $x_0 \in \mathbb{R}$, if the following condition is satisfied*

$$M \geq \frac{16c_0(1+\varepsilon)}{\varepsilon^2}\left(\log\left(\frac{1}{\delta}\right) + \log\left(1 + \frac{c_x T}{x_0^2}\right)\right), \tag{20}$$

we have, with probability at least $1-\delta$

$$\forall t \in \mathcal{T}, \quad (1-\varepsilon)\left(\sum_{i=0}^{t} x_i^2\right) \leq \|\sum_{i=0}^{t} x_i \mathbf{z}_i\|^2 \leq (1+\varepsilon)\left(\sum_{i=0}^{t} x_i^2\right). \tag{21}$$

Remark D.3. Li (2024a) claims this is a "sequential random projection" argument because one can relate Theorem D.2 to the traditional random projection (Lemma D.1) setting where $\Pi_t = (\mathbf{z}_0, \ldots, \mathbf{z}_t) \in \mathbb{R}^{M\times t+1}$ is a random projection matrix and $\boldsymbol{x}_t = (x_0, \ldots, x_t)^\top \in \mathbb{R}^{t+1}$ is the vector to be projected. The argument in Equation (21) translates to

$$\forall t \in \mathcal{T}, \quad (1-\varepsilon)\|\boldsymbol{x}_t\|^2 \leq \|\Pi_t \boldsymbol{x}_t\|^2 \leq (1+\varepsilon)\|\boldsymbol{x}_t\|^2. \tag{22}$$

When assuming independence between $\boldsymbol{x}_t$ and Π_t for all $t \in \mathcal{T}$, by simply applying union bound over time index $t \in \mathcal{T}$ with existing JL analysis, we can derive that the required dimension $M = O(\varepsilon^{-2}\log(T/\delta))$ is of the same order in Equation (20). **However**, as discussed in (Li, 2024a), existing JL analytical techniques are *not* able to handle the sequential dependence in our setup as $\boldsymbol{x}_t$ is statistically dependent with Π_t for $t \in \mathcal{T}$. In short, the fundamental difficulties for adapting existing JL techniques in our setup are **(1)** when conditioned on x_t, the random variables $(\mathbf{z}_s)_{s<t}$ **loss** their independence; **(2)** there is **no** characterization on the conditional distribution $P_{(\mathbf{z}_s)_{s<t}|x_t}$.

Proof of Lemma 4.1. **Step 1: Prior approximation.** We first show the prior approximation by the fixed prior model, i.e. the event $\mathcal{G}_{1,sa}(\varepsilon/2) = \{|\|\sigma_0 \mathbf{z}_{0,sa}\|^2 - \sigma_0^2| \leq \frac{\varepsilon}{2}\sigma_0^2\}$ holds for any (s,a). This is obvious true even for $\varepsilon = 0$ due to the fact $\|\mathbf{z}_{0,sa}\| = 1$ for all $(s,a) \in \mathcal{S}\times\mathcal{A}$ as by the choice of the perturbation distribution $P_{\mathbf{z}} := \mathcal{U}(\mathbb{S}^{M-1})$.

Step 2: Posterior approximation. Recall that $\beta = \sigma^2/\sigma_0^2$. To handle the posterior approximation, we first define a sequence of indicator variables

$$x_{\ell,t} = \mathbb{1}_{t\in E_{\ell,sa}},$$

where $E_{\ell,sa}$ is the collection of time steps in episode ℓ encountering state-action pair (s,a). We also define auxiliary notations $\mathbf{z}_0 := \mathbf{z}_{0,sa}$ and $x_0 = \sqrt{\beta}$. Immediately, we could rewrite Equation (6) as

$$\frac{(N_{k,sa}+\beta)}{\sigma}\tilde{m}_{k,sa} = x_0\mathbf{z}_0 + \sum_{\ell=1}^{k-1}\sum_{t\in E_\ell} x_{\ell,t}\mathbf{z}_{\ell,t+1} \tag{23}$$

The reorganization in Equation (23) is essential to reduce Lemma 4.1 to the following argument:

Remark D.4 (Reduction). In the following, we are going to prove with probability $1-\delta$, the Equation (24) holds for all $(s,a) \in \mathcal{S}\times\mathcal{A}$ and $k \in [K]$ simultaneously:

$$(1-\varepsilon)(x_0^2 + \sum_{\ell=1}^{k-1}\sum_{t\in E_\ell} x_{t,\ell}^2) \leq \|x_0\mathbf{z}_0 + \sum_{\ell=1}^{k-1}\sum_{t\in E_\ell} x_{\ell,t}\mathbf{z}_{\ell,t+1}\|^2 \leq (1+\varepsilon)(x_0^2 + \sum_{\ell=1}^{k-1}\sum_{t\in E_\ell} x_{\ell,t}^2) \tag{24}$$

Recall the notations E_ℓ for the collection of time steps in episode ℓ and $E_{\ell,sa}$ for the collection of time steps in episode ℓ. Importantly, the sequential dependence structure in `HyperAgent` when interacting with environment is that

- $x_{\ell,t} := \mathbb{1}_{t\in E_{\ell,sa}}$ is **dependent** on the environmental and algorithmic randomness in all previous time steps:

$$\mathbf{z}_0, (x_{1,t'}, \mathbf{z}_{1,t'+1})_{t'\in E_1}, (x_{2,t'}, \mathbf{z}_{2,t'+1})_{t'\in E_2}, \ldots, (x_{\ell,t'}, \mathbf{z}_{\ell,t'+1})_{t'<t};$$

- $\mathbf{z}_{\ell,t+1}$ is **independent** of the environmental and algorithmic randomness in all previous time steps:

$$\mathbf{z}_0, (x_{1,t'}, \mathbf{z}_{1,t'+1})_{t' \in E_1}, (x_{2,t'}, \mathbf{z}_{2,t'+1})_{t' \in E_2} \ldots, (x_{\ell,t'+1}, \mathbf{z}_{\ell,t'+1})_{t'<t}, x_{\ell,t},$$

The difficulty of posterior approximation comes from the above dependence structure as we can not directly use argument conditioning on the entire history $\mathcal{H}_\ell$ at once. This is because the conditional distributions of $(\mathbf{z}_{\ell',t'})$ given $\mathcal{H}_\ell$ are changed from the unconditional one, without clear characterization. Besides, the random variables $((\mathbf{z}_{\ell',t'})$ are not conditionally independent given the history $\mathcal{H}_\ell$.

These difficulties calls for the innovation on fundamental tools in probability with martingale analysis, as shown in Theorem D.2 and Remark D.3.

Prior approximation guarantees the initial condition $\|\mathbf{z}_0\|^2 = 1$ for applying Theorem D.2. Observe that for all (k, s, a), we have $x_{\ell,t} := \mathbb{1}_{t \in E_{\ell,sa}} \leq 1$ bounded and $\sum_{\ell=1}^{k-1} \sum_{t \in E_\ell} x_{\ell,t}^2 = \sum_{\ell=1}^{k-1} \sum_{t \in E_\ell} \mathbb{1}_{t \in E_{\ell,sa}} = N_{k,sa}$. Also, as proved in (Li, 2024b;a), at each time step (ℓ, t), the perturbation random vector $\mathbf{z}_{\ell,t} \sim P_{\mathbf{z}} := \mathcal{U}(\mathbb{S}^{M-1})$ is the $\frac{1}{\sqrt{M}}$-sub-Gaussian random variable in $\mathbb{R}^M$. Now, we are ready to apply Theorem D.2 to the RHS of Equation (23) with sequence $(\mathbf{z}_{\ell,t})$ and $(x_{\ell,t})$. This yields the results $\mathbb{P}(\cap_{k=1}^{K+1} \mathcal{G}_{k,sa}(\varepsilon)) \geq 1 - \delta$ if

$$M \geq \frac{16(1+\varepsilon)}{\varepsilon^2} \left(\log\left(\frac{1}{\delta}\right) + \log\left(1 + \frac{T}{\beta}\right) \right),$$

where $\sum_{k=1}^{K} |E_k| = T$ almost surely. Then, by union bound over the set $\mathcal{S} \times \mathcal{A}$, we conclude that if

$$M \geq \frac{16(1+\varepsilon)}{\varepsilon^2} \left(\log\left(\frac{|\mathcal{S}||\mathcal{A}|}{\delta}\right) + \log\left(1 + \frac{T}{\beta}\right) \right), \tag{25}$$

then $\mathbb{P}\left(\cap_{(s,a) \in \mathcal{S} \times \mathcal{A}} \cap_{k=1}^{K+1} \mathcal{G}_{k,sa}(\varepsilon)\right) \geq 1 - \delta$. □

Remark D.5. According to the proof, the above sequential posterior approximation argument Lemma 4.1 holds for any tabular MDP. It does *not* rely on any assumptions made latter for regret analysis. If the tabular MDP additionally satisfies Assumption D.7, the same result in Lemma 4.1 holds when

$$M \geq M(\varepsilon) := \frac{16(1+\varepsilon)}{\varepsilon^2} \left(\log\left(\frac{|\mathcal{S}||\mathcal{A}|}{\delta}\right) + \log\left(1 + \frac{K}{\beta}\right) \right), \tag{26}$$

where $|\mathcal{S}| = SH$. The difference between Equation (26) and Equation (25) is in the $\log(1 + K/\beta)$ term. This difference is due to the fact we apply Theorem D.2 for random variables only in a single stage t across episode $\ell = 1, \ldots, K$ under Assumption D.7 since the visitation counts $N_{k,(t,x),a}$ only takes the historical data in stage t into considerations for all $(t, x) \in \mathcal{S}_t$, $a \in \mathcal{A}$ and $t \in \{0, \ldots, H-1\}$.

D.2. Insight: How does HyperAgent drives efficient deep exploration?

In this section, we highlight the key components of HyperAgent that enable efficient deep exploration. We consider a simple example (adapted from (Osband et al., 2019)) to understand the HyperAgent's learning rule in Equations (4) and (5) and the role of hypermodel, and how they together drive efficient deep exploration.

Example D.6. Consider a fixed horizon MDP $\mathcal{M}$ with four states $\mathcal{S} = \{1, 2, 3, 4\}$, two actions $\mathcal{A} = \{up, down\}$ and a horizon of $H = 6$. Let us consider the scenario when the agent is at the beginning of k-th episode. Let $\mathcal{H}_k$ be the history of all transitions observed prior to episode k, and let $\mathcal{H}_{k,sa} = \left((\hat{s}, \hat{a}, r, s') \in \mathcal{H}^k : (\hat{s}, \hat{a}) = (s, a)\right)$ contain the transitions from state-action pair (s, a) encountered before episode k. Suppose $|\mathcal{H}_{4,down}^k| = 1$, while for every other pair $(s, a) \neq (4, down)$, $|\mathcal{D}_{s,a}|$ is very large, virtually infinite. Hence, we are highly certain about the expected immediate rewards and transition probabilities except for $(4, down)$.

From Equations (6) to (8), HyperAgent produces a sequence of action-value functions $Q^{(i)} := f_{\theta_k^{(i)}}$ for $i = 0, 1, 2, \ldots, 6$ as shown in Figure 7 by iteratively solving Equation (4) starting from $Q^{(0)} = \mathbf{0}$,

$$Q^{(i+1)}(s, a) = F_k^\gamma Q^{(i)}(s, a) = \frac{\beta \mu_{0,sa} + N_{k,sa}(r_{sa} + \gamma V_{Q^{(i)}}^\top \hat{P}_{k,sa})}{N_{k,sa} + \beta} + \tilde{m}_{k,sa}^\top \boldsymbol{\xi}_k(s),$$

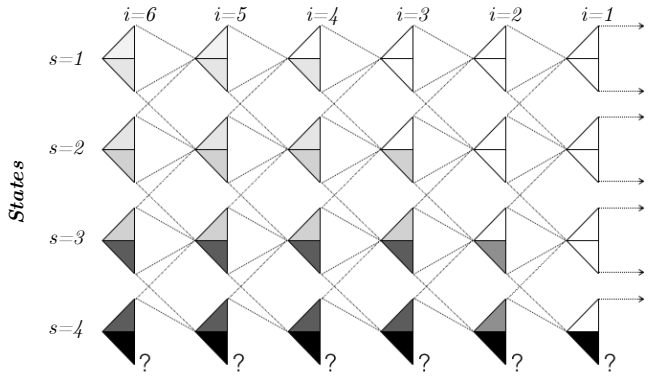


Figure 7. Example to illustrate how HyperAgent achieves deep exploration. We can see the propagation of uncertainty from later time period to earlier time period in the figure. Darker shade indicates higher degree of uncertainty.

In Figure 7, each triangle in row s and column t contains two smaller triangles that are associated with action-values of up and $down$ actions at state s. The shade on the smaller triangle shows the uncertainty estimates in the $Q^{(i)}(s,a)$, specifically the variance. The dotted lines show plausible transitions, except at $(4, down)$. Since we are uncertain about $(4, down)$, any transition is plausible.

As shown in the Figure 7, when $i = 1$, since data size at $(s,a) \neq (4, \text{down})$ tends to infinity, $\|\tilde{m}_{k,sa}\|^2 \approx 0$ from our Lemma 4.1, thus almost zero variance and white colored at $(s,a) \neq (4, \text{down})$; while $(s,a) = (4, \text{down})$ exhibits large variance, thus dark share, due to the lack of data $|\mathcal{H}_{k,(4,\text{down})}| = 1$ and Lemma 4.1. By performing bellman update, the noise term $\tilde{m}_{k,sa}^\top \boldsymbol{\xi}_k(s)$ at $(s,a) = (4, \text{down})$ is back-propagated to other states consecutively by the iterative process $i = 2, 3, 4, 5$. This is because other state-action pairs may lead to the transition into $(4, \text{down})$, thus being influenced by the variance introduced by the estimation of $\mathcal{Q}^{(1)}(4, \text{down})$. As traced in Figure 7, it's clear that the propagation of noise affects the value estimates in a pattern depicted by the figure's shading.

This dynamic is crucial for fostering deep exploration. In essence, a sample $\mathcal{Q}^{(6)}(s,a)$ with high variance may appear excessively optimistic in certain episodes, incentivizing the agent to explore that action. Such exploration is justified by the agent's uncertainty about the true optimal value $Q^\star(s,a)$ across the planning horizon. This incentive extends beyond just the immediate reward and transition uncertainty. As depicted in Figure 7, the spread of uncertainty through the system creates incentives for the agent to seek out information, potentially over several time steps, to make a informative observation. This process underlines the core principle of deep exploration.

D.3. Provable scalability and efficiency of HyperAgent

The design goal of RL algorithm is to maximize the expected total reward up to episode K: $\mathbb{E}\left[\sum_{k=1}^{K}\sum_{t=1}^{\tau_k} R_{k,t} \mid \mathcal{M}, \pi_{\text{agent}}\right]$, which is equivalent to $\mathbb{E}\left[\sum_{k=1}^{K} V_{\mathcal{M}}^{\pi_k}(s_{k,0}) \mid \mathcal{M}, \pi_{\text{agent}}\right]$. Note that the expectations are over the randomness from stochastic transitions $P(\cdot \mid \cdot)$ under the given MDP $\mathcal{M}$, and the algorithmic randomization introduced by agent. The expectation in the former one is also over the random termination time τ_k.

Next, we give some specific assumptions under which we simplify the exposition on the analysis.

D.3.1. Assumptions

Assumption D.7 (Finite-horizon time-inhomogeneous MDPs)**.** We consider a problem class that can be formulated as a special case of the general formulation in Section 2. Assume the state space factorizes as $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_{H-1}$ where the state always advances from some state $s_t \in \mathcal{S}_t$ to $s_{t+1} \in \mathcal{S}_{t+1}$ and the process terminates with probability 1 in period H, i.e.,

$$\sum_{s' \in \mathcal{S}_{t+1}} P_{sa}(s') = 1 \quad \forall t \in \{0, \ldots, H-2\}, s \in \mathcal{S}_t, a \in \mathcal{A}$$

and

$$\sum_{s' \in \mathcal{S}} P_{sa}(s') = 0 \quad \forall s \in \mathcal{S}_{H-1}, a \in \mathcal{A}.$$

For notational convenience, we assume each set $\mathcal{S}_0, \ldots, \mathcal{S}_{H-1}$ contains an equal number of S elements. Each state $s \in \mathcal{S}_t$ can be written as a pair $s = (t, x)$ where $t \in \{0, \ldots, H-1\}$ and $x \in \mathcal{X} = \{1, \ldots, S\}$. That is, $|\mathcal{S}| = SH$.

Our notation can be specialized to this time-inhomogenous problem, writing transition probabilities as $P_{t,xa}(x') \equiv P_{(t,x),a}((t+1, x'))$. For consistency, we also use different notation for the optimal value function, writing

$$V^{\pi}_{\mathcal{M},t}(x) \equiv V^{\pi}_{\mathcal{M}}((t, x))$$

and define $V^{*}_{\mathcal{M},t}(x) := \max_{\pi} V^{\pi}_{\mathcal{M},t}(x)$. Similarly, we can define the state-action value function under the MDP at timestep $t \in \{0, \ldots, H-1\}$ by

$$Q^{*}_{\mathcal{M},t}(x, a) = \mathbb{E}\left[r_{t+1} + V^{*}_{\mathcal{M},t+1}(x_{t+1}) \mid \mathcal{M}, x_t = x, a_t = a\right] \quad \forall x \in \mathcal{X}, a \in \mathcal{A}.$$

This is the expected reward accrued by taking action a in state x and proceeding optimally thereafter. Equivalently, the process applying true bellman operator $F_{\mathcal{M},t}Q^{*}_{\mathcal{M},t+1}(\cdot,\cdot) := F^{\gamma}_{\mathcal{M}}Q^{*}_{\mathcal{M}}((t+1,\cdot),\cdot)$ when $\gamma = 1$, where $F^{\gamma}_{\mathcal{M}}$ is defined in Equation (9) yields a series of optimal state-action value functions, fulfilling $Q^{*}_{\mathcal{M},H} = 0$ and the Bellman equation $Q^{*}_{\mathcal{M},t} = F_{\mathcal{M},t}Q^{*}_{\mathcal{M},t+1}$ for $t < H$.

Hyper-parameters	Finite MDP with Horizon H
reference distribution P_{ξ}	$N(0, I_M)$
perturbation distribution $P_{\mathbf{z}}$	$\mathcal{U}(\mathbb{S}^{M-1})$
level of perturbation σ^2	$6H^2$
prior variance σ_0^2	$6H^2/\beta$
prior regularization β	β in Assumption D.8
index dimension M	$M(1/2)$ in Equation (26)
discount factor γ	1
target_update_freq	1
sample_update_ratio	1
training_freq	H

Table 6. Hyper-parameters of our Tabular-HyperAgent and corresponding update rules.

The agent designer's prior beliefs over MDPs M is formalized with mild assumptions.

Assumption D.8 (Independent Dirichlet prior for outcomes)**.** For each $(s, a) \in \mathcal{S} \times \mathcal{A}$, the outcome distribution is drawn from a Dirichlet prior

$$P_{sa} \sim \text{Dirichlet}(\alpha_{0,sa})$$

for $\alpha_{0,sa} \in \mathbb{R}^{\mathcal{S}}_{+}$ and each P_{sa} is drawn independently across (s, a). Assume there is $\beta \geq 3$ such that $\mathbf{1}^{\top}\alpha_{0,sa} = \beta$ for all (s, a).

D.3.2. Bayesian analysis

Denote the short notation $[n] = \{1, \ldots, n\}$. Let us define a filtration $(\mathcal{Z}_k)_{k \geq 1}$ on the algorithmic random perturbation that facilitates the analysis

$$\mathcal{Z}_k = \sigma((\mathbf{z}_{0,sa})_{(s,a) \in \mathcal{S} \times \mathcal{A}}, (\mathbf{z}_{\ell,t} : \ell \in [k-1], t \in E_{\ell})).$$

Specifically from this definition, $\tilde{m}_{k,sa}$ is $(\mathcal{H}_k, \mathcal{Z}_k)$-measurable. As derived in Equation (8), the perturbation $\tilde{m}^{\top}_{k,s,a}\boldsymbol{\xi}_k(s)$ injected to the Bellman update is conditionally Gaussian distributed

$$\tilde{m}^{\top}_{k,s,a}\boldsymbol{\xi}_k(s) \mid \mathcal{H}_k, \mathcal{Z}_k \sim N(0, \|\tilde{m}_{k,s,a}\|^2), \tag{27}$$

due to the fact that for all $s \in \mathcal{S}$, $\boldsymbol{\xi}_k(s)$ is independent of $\mathcal{H}_k, \mathcal{Z}_k$ and is Normal distributed.

Meanwhile, using notation $s = (t, z)$ in the time-inhomogeneous setting where $|E_\ell| = H$ for all ℓ, these injected perturbation $\tilde{m}_{k,(t,x),a}^\top \boldsymbol{\xi}_k((t,x))$ are conditionally independent across $(t,x) \in ([H-1] \cup \{0\}) \times \mathcal{X}$ given $\mathcal{H}_t, \mathcal{Z}_t$. From Lemma 4.1 using $M(1/2)$ in Equation (26), the noise level of $\tilde{m}_{k,(t,x),a}^\top \boldsymbol{\xi}_k((t,x))$ is a $(1/2)$-approximation of the noise level of $\omega_k(t,x,a)$ through out the entire learning process for all tuples $(k,(t,x),a)$. Therefore, the role of the injected perturbation $\tilde{m}_{k,(t,x),a}^\top \boldsymbol{\xi}_k((t,x))$ in the Bellman update of `HyperAgent` is essentially the same as the Gaussian noise $\omega_k(t,x,a)$ injected in the Bellman udpate of RLSVI.

Basically, as long as the sequential approximation argument in Lemma 4.1 is established, the regret analysis follows analysis of RLSVI in the Section 6 of (Osband et al., 2019). Still, we want to emphasize a key argument that enables efficient deep exploration is the stochastic optimism of `HyperAgent` by a selection on $\sigma^2 = 6H^2$ which is a double of the number of σ^2 selected in (Osband et al., 2019). The rest of the analysis follows Section 6.4 (Optimism and regret decompositions) and Section 6.6 (Analysis of on-policy Bellman error) in (Osband et al., 2019).

Definition D.9 (Stochastic optimism)**.** A random variable X is stochastically optimistic with respect to another random variable Y, written $X \geq_{SO} Y$, if for all convex increasing functions $u : \mathbb{R} \to \mathbb{R}$

$$\mathbb{E}[u(X)] \geq \mathbb{E}[u(Y)].$$

We show that `HyperAgent` is stochastic optimistic in the sense that it overestimates the value of each state-action pairs in expectation. This is formalized in the following proposition.

Proposition D.10. *If Assumptions D.7 and D.8 hold and tabular* `HyperAgent` *is applied with planning horizon H and parameters parameters $(M, \mu_0, \sigma, \sigma_0)$ satisfying $M = M(1/2)$ defined in Equation* (26)*, $(\sigma^2/\sigma_0^2) = \beta$, $\sigma \geq \sqrt{6}H$ and $\min_{s,a} \mu_{0,s,a} \geq H$,*

$$f_{\theta_k}(s, a, \boldsymbol{\xi}_k(s)) \mid (\mathcal{H}_k, \mathcal{Z}_k) \succeq_{SO} Q^*_{\mathcal{M}}(s,a) \mid (\mathcal{H}_k, \mathcal{Z}_k), \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A} \tag{28}$$

holds for all episode $k \in \{0, \ldots, K\}$ simultaneously with probability at least $(1-\delta)$.

The following Lemma D.11 about the stochastic optimistic operators is also built upon our sequential posterior approximation argument in Lemma 4.1. As long as Lemma D.11 is established, the proof of Proposition D.10 follows the proof of Corollary 2 in (Osband et al., 2019).

Lemma D.11 (δ-approximate stochastically optimistic operators)**.** *If Assumption D.8 holds and* `HyperAgent` *is applied with $(M, \mu_0, \sigma, \sigma_0)$ satisfying $M = M(1/2)$ defined in Equation* (25) *or Equation* (26)*, $\sigma^2/\sigma_0^2 = \beta$, $\sigma \geq \sqrt{6}\gamma \operatorname{Span}(V_Q)$ and $\min_{s,a} \mu_0(s,a) \geq \gamma \max_{s \in \mathcal{S}} V_Q(s) + 1$,*

$$F_k^\gamma Q(s,a) \mid (\mathcal{H}_k, \mathcal{Z}_k) \succeq_{SO} F_{\mathcal{M}}^\gamma Q(s,a) \mid (\mathcal{H}_k, \mathcal{Z}_k), \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{29}$$

holds for all episode $k \in \{0, \cdots, K\}$ simultaneously with probability at least $(1-\delta)$.

Remark D.12. The Lemma D.11 with $M(1/2)$ in Equation (25) does not require time-inhomogeneity Assumption D.7 and holds with any fixed $\gamma \geq 0$. In the time-inhomogeneous problem, one can set $M(1/2)$ in Equation (26), $\gamma = 1$ and $\operatorname{Span}(V_Q) \leq \max_s V_Q \leq H - 1$ as V_Q is from the case that $\{t = 1, \ldots, H-1\}$.

Proof of Lemma D.11. Recall from Equation (9), we have

$$F_{\mathcal{M}}^\gamma Q(s,a) = r_{sa} + \gamma V_Q^\top P_{sa} = (r_{sa}\mathbf{1} + \gamma V_Q)^\top P_{sa},$$

where $\mathbf{1}$ is the all one vector in $\mathbb{R}^{|\mathcal{S}|}$. Notice that by Assumption D.8, for each pair (s,a), the transition vector $P_{sa} \perp\!\!\!\perp \mathcal{Z}_k \mid \mathcal{H}_k$ and the conditional distribution is

$$\mathbb{P}(P_{sa} \in \cdot \mid (\mathcal{H}_k, \mathcal{Z}_k)) = \mathbb{P}(P_{sa} \in \cdot \mid \mathcal{H}_k) = \operatorname{Dirichlet}(\alpha_{k,sa}),$$

where $\alpha_{k,sa} = \alpha_{0,sa} + N_{k,sa}\hat{P}_{k,sa} \in \mathbb{R}^S$. Define the posterior mean of the transition vector as

$$\bar{P}_{k,sa} := \mathbb{E}\left[P_{sa} \mid \mathcal{H}_k\right] = \frac{\alpha_{0,sa} + N_{k,sa}\hat{P}_{k,sa}}{\beta + N_{k,sa}} \in \mathbb{R}^{|\mathcal{S}|}, \tag{30}$$

which can be interpreted as the empirical transition $\hat{P}_{s,a}$ but with a slight regularization towards the prior mean $\mu_{0,sa}$ by weights $\beta = \mathbf{1}^\top \alpha_{0,sa}$.

As mentioned in Section 4 and Equation (27), from Equation (8)

$$F_k^\gamma Q(s,a) \mid (\mathcal{H}_k, \mathcal{Z}_k) \sim N(\mu_{k,sa}, \|\tilde{m}_{k,sa}\|^2)$$

where the conditional mean $\mu_{k,sa}$ is

$$\mu_{k,sa} = (r_{sa}\mathbf{1} + \gamma V_Q)^\top \bar{P}_{k,sa} + \frac{\beta\mu_{0,sa} - \beta r_{sa} - \gamma V_Q^\top \alpha_{0,sa}}{\beta + N_{k,sa}}. \tag{31}$$

From our established Lemma 4.1, when $M = M(\varepsilon)$, the joint event $\cap_{(k,s,a)\in[K]\times\mathcal{S}\times\mathcal{A}} G_{k,sa}(\varepsilon)$ holds with probability $1-\delta$. Specifically, $G_{k,sa}(1/2) \in \sigma(\mathcal{H}_k, \mathcal{Z}_k)$ implies that

$$\|\tilde{m}_{k,sa}\|^2 \geq (1/2)\sigma^2/(\beta + N_{k,sa}).$$

Under the above established event, the result follows from Lemma D.13 if we establish on conditional variance $\|\tilde{m}_{k,sa}\|^2 \geq 6(\mathbf{1}^\top\alpha_{k,sa})^{-1}\,\mathrm{Span}(r_{sa}\mathbf{1} + \gamma V_Q)^2$ and on conditional mean $\mu_{k,sa} \geq (\mathbf{1}^\top\alpha_{k,sa})^{-1}(r_{sa}\mathbf{1} + \gamma V_Q)^\top\alpha_{k,sa} = (r_{sa}\mathbf{1} + \gamma V_Q)^\top\bar{P}_{k,sa}$ since $\mathbf{1}^\top\alpha_{0,sa} = \beta$ for all (s,a). That is, for conditional variance

$$\frac{3\cdot\mathrm{Span}(r_{sa}\mathbf{1} + \gamma V_Q)^2}{(\mathbf{1}^\top\alpha_{k,sa})} \leq \frac{3\gamma^2\cdot\mathrm{Span}(V_Q)^2}{\beta + N_{k,sa}} \overset{(1)}{\leq} \frac{(1/2)\sigma^2}{\beta + N_{k,sa}} \overset{(2)}{\leq} \|\tilde{m}_{k,sa}\|^2,$$

where (1) is from the condition of σ in Lemma D.11 and (2) holds true under the event $G_{k,sa}(1/2)$. Next we have the conditional mean $\mu_{k,sa}$ dominating the posterior mean of true bellman update $\mathbb{E}\left[F_{\mathcal{M}}^\gamma Q \mid \mathcal{H}_k\right] = (r_{sa}\mathbf{1} + \gamma V_Q)^\top\bar{P}_{k,sa}$,

$$\begin{aligned}
\mu_{k,sa} - (r_{sa}\mathbf{1} + \gamma V_Q)^\top\bar{P}_{k,sa} &= \frac{\beta\mu_{0,sa} - \beta r_{sa} - \gamma V_Q^\top\alpha_{0,sa}}{\beta + N_{k,sa}} \\
&\geq \frac{\beta\mu_{0,sa} - \beta(\gamma\max_{s\in\mathcal{S}} V_Q(s) + 1)}{\beta + N_{k,sa}} \\
&\geq 0
\end{aligned}$$

because of the condition in Lemma D.11 that $\min_{sa}\mu_{0,sa} \geq \gamma\max_s V_Q(s) + 1$. □

Lemma D.13 (Gaussian vs Dirichlet optimism, Lemma 4 in (Osband et al., 2019))**.** *Let $Y = p^\top v$ for $v \in \mathbb{R}^n$ fixed and $p \sim \mathrm{Dirichlet}(\alpha)$ with $\alpha \in \mathbb{R}_+^n$ and $\sum_{i=1}^n \alpha_i \geq 3$. Let $X \sim N\left(\mu, \sigma^2\right)$ with $\mu \geq (\sum_{i=1}^n \alpha_i)^{-1}\sum_{i=1}^n \alpha_i V_i, \sigma^2 \geq 3\left(\sum_{i=1}^n \alpha_i\right)^{-1}\mathrm{Span}(V)^2$, then $X \succeq_{SO} Y$. Thus, $p^\top v$ is sub-Gaussian with parameter σ^2.*

E. Understanding HyperAgent via comprehensive studies on DeepSea

In this section, we present insightful experiments to demonstrate the impact of critical components on HyperAgent. Specifically, we employ DeepSea to validate the theoretical insights of our method, discuss the sampling schemes for Q-target computation and action selection, introduce methods for achieving accurate posterior approximation, and compare with additional baselines within the hypermodel framework as discussed in Appendix C.3.2.

E.1. Validating theoretical insights through experimentation

Empirical validation of theoretical insights on the index dimension M. According to the theoretical analysis in Lemma 4.1, increasing the index dimension M can enhance the approximation of posterior covariance under closed-form solution of Equation (4), where its expectation is precisely computed, thereby facilitating deep exploration.

In this experiment, we set the number of indices $|\tilde{\Xi}|$ of Equation (5) to 20 for all index dimensions $\{1, 2, 4, 8\}$. Let us first discuss the case $M = 1$, it can be regarded as an incremental updated RLSVI point estimate without resampling the perturbation and re-solving the least-square for the entire history. As evidenced by the results in Figure 8, the incremental version of RLSVI cannot fully solve DeepSea tasks starting from size of 30, indicating dimension $M > 1$ is necessary and increasing a bit on M benefits deep exploration, possibly from more accurate posterior approximation by larger M.

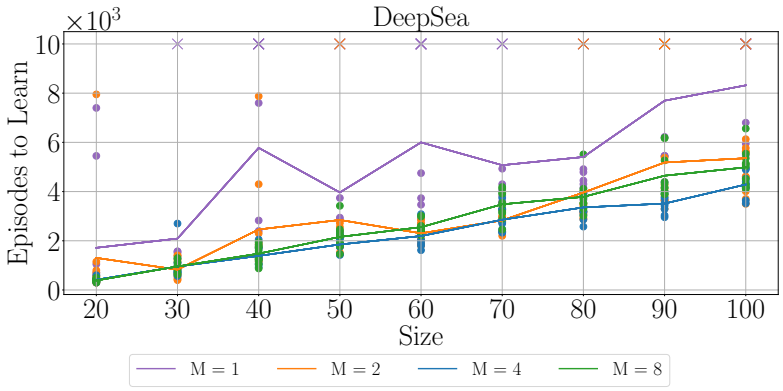


Figure 8. Ablation results on the impact of index dimension M. HyperAgent exhibits improved performance when M exceeds 1, aligning with our theoretical analysis.

Ablation experiment on different sampling schemes for ξ. We have two sampling schemes for ξ in the computation of the Q-target and action selection: state-independent ξ sampling and state-dependent ξ sampling, see detailed description in Appendix C.2. In our implementation by default, HyperAgent employs the state-independent ξ for action selection, where we use the same index $\xi_k(S) = \xi_k$ for each state within episode k. Whereas, HyperAgent utilizes state-dependent ξ for Q-target computation, where we sample independent $\xi \sim P_\xi$ when computing the Q-target for each transition tuple in mini-batches. To systematically compare the index sampling schemes on action selection and Q-target computation, two variants are created.

- HyperAgent *with state-independent* ξ: applying state-independent ξ sampling to both Q-target computation and action selection.
- HyperAgent *with state-dependent* ξ: applying state-dependent ξ sampling to both Q-target computation and action selection.

As illustrated in Figure 9, these distinct index sampling schemes for ξ exhibit nearly identical performance.

Comparative results on optimistic index sampling. Introduced in C.2, optimistic index sampling (OIS) for action selection and Q-target computation is another index sampling scheme in HyperAgent framework, naming HyperAgent w. OIS. As shown in Figure 10(a), HyperAgent w. OIS outperforms HyperAgent by leveraging the OIS method to

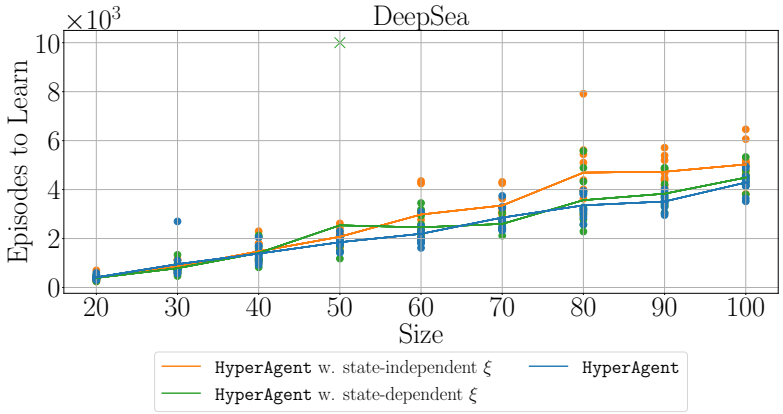


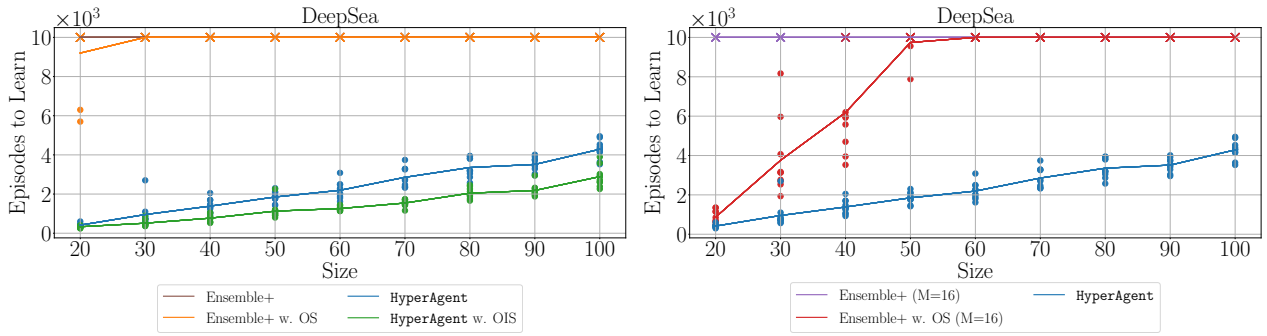
Figure 9. Ablation results for the sampling schemes used in Q-target computation. Both sampling schemes achieve similar performance.

generate more optimistic estimates, thereby enhancing exploration. The OIS method incurs minimal additional computation, as we have set the dimension M to 4.

The empirical implementation of LSVI-PHE (Ishfaq et al., 2021) also utilizes the similar optimistic sampling (OS) method with ensemble networks as described in Appendix A. Since no official implementation of LSVI-PHE is available, we apply the OIS to the Ensemble+ (Osband et al., 2018; 2019), denoted as Ensemble+ w. OS, representing the LSVI-PHE. A critical difference between LSVI-PHE and `HyperAgent` w. OIS is that LSVI-PHE performs maximum value over M separately trained ensemble models, while `HyperAgent` w. OIS has the computation advantage that we can sample as many as indices from reference distribution P_ξ to form the optimistic estimation of value function even when the index dimension M is small.

When $M = 4$, both Ensemble+ and Ensemble+ w. OS demonstrates subpar performance. According to Osband et al. (2018) and the observation in Figure 10, a larger ensemble size ($M = 16$) can lead to improved performance. When $M = 16$, both Ensemble+ w. OS (LSVI-PHE) has a bit better performance compared with Ensemble+. The result of Ensemble+ w. OS ($M = 16$) and `HyperAgent` w. OIS demonstrates enhanced deep exploration via optimistic value estimation, which is aligned with our theoretical insights.

In another dimension of comparison, As depicted in Figure 10(b), Ensemble+ w. OS ($M = 16$) still fails to solve the large-size DeepSea. This further showcases the exploration efficiency of `HyperAgent`.



(a) We set $M = 4$ for all algorithms. `HyperAgent` w. OIS achieves the best performance.

(b) We set $M = 16$ for algorithms with ensemble network, and keep $M = 4$ for `HyperAgent`.

Figure 10. Results on different action selection schemes. The OIS method can achieve better performance due to the optimistic estimates.

E.2. Ablation studies on implementation settings and hyper-parameters

Sampling-based approximation. HyperAgent by default uses Gaussian reference distribution $P_\xi = N(0, I_M)$, which requires sampling-based approximation Equations (4) and (5). Therefore, it is imperative to set the number of indices $|\tilde{\Xi}|$ of Equation (5) to be sufficiently large for a good approximation given index dimension M. $|\tilde{\Xi}| = 20$ falls short of delivering satisfactory results when $M = 16$ as illustrated in Figure 11. For this scenario, increasing the number of indices $|\tilde{\Xi}|$ can alleviate performance degradation as depicted in Figure 12, facilitating deep exploration empirically. To achieve accurate approximation, $|\tilde{\Xi}|$ may need to grow exponentially with M, but this comes at the cost of increased computation. To balance accuracy and computation, we have chosen $M = 4$ and $|\tilde{\Xi}| = 20$ as the default hyper-parameters, which already demonstrate superior performance in Figure 3.

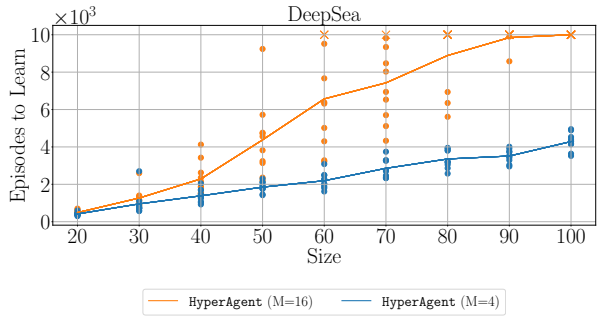


Figure 11. Results of HyperAgent with large index dimension. In both settings, we set $|\tilde{\Xi}| = 20$, which fails to achieve satisfactory performance for $M = 16$.

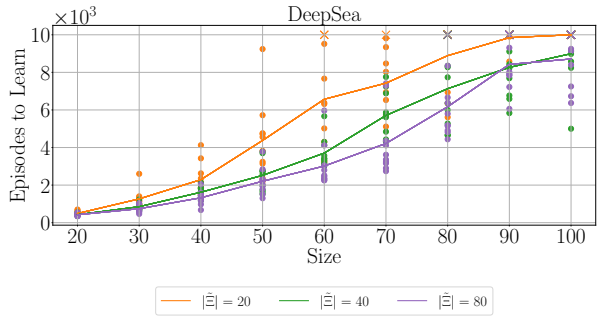


Figure 12. Results of HyperAgent with index dimension $M = 16$. We can increase the number of indices $|\tilde{\Xi}|$ to alleviate performance degradation,

Ablation results for other hyper-parameters. In addition, we have also investigated the effect of the σ of Equation (11) on our method, as shown in Figure 13. HyperAgent is not sensitive to this hyper-parameter, and we have selected $\sigma = 0.0001$ as the default hyper-parameter.

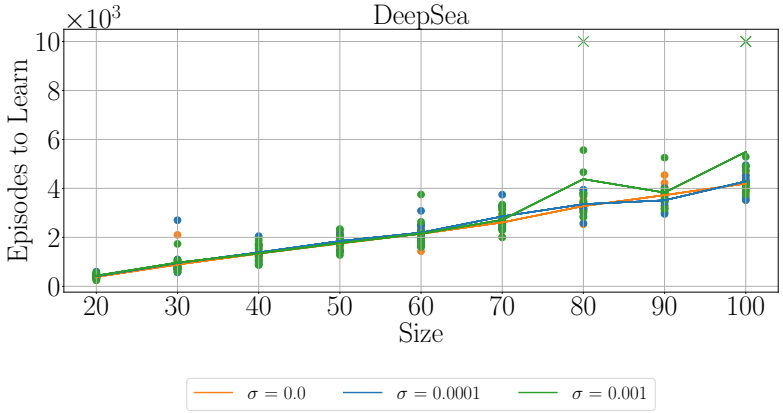


Figure 13. Ablation results on the $\sigma = 0.0001$ of Equation (11). We set $\sigma = 0.0001$ as the default setting.

E.3. Additional results with algorithms using hypermodel framework

Experiment investigating different network structures within the hypermodel framework. We perform an ablation experiment examining various network structures, (1) HyperModel (Dwaracherla et al., 2020) (2) epinet (Osband et al., 2023b), outlined in Appendix C.3.2 with the same hyper-parameters, same algorithmic update rule, same target computation rule and same action selection rule.

The comparison results are presented in Figure 14. HyperModel is unable to solve DeepSea, even with a size of 20, while ENNDQN cannot solve DeepSea when the size exceeds 30. Overall, HyperAgent demonstrates superb efficiency and

scalability, as it efficiently solves DeepSea instances of size 100^2 that previous literature has never achieved.

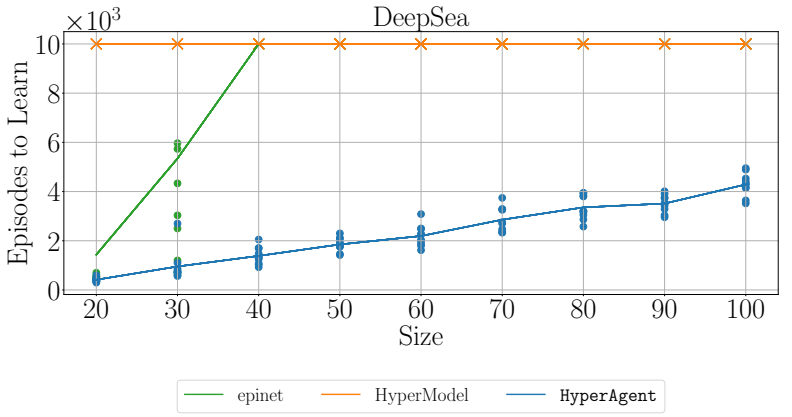


Figure 14. Results from various network structures using the hypermodel framework. `HyperAgent` demonstrates superior data and computational efficiency compared to other methods.

Ablation experiment on Q-target computation. Several methods, including HyperDQN, ENNDQN, and Ensemble+, employ the injected index to achieve posterior approximation and utilize the same index to compute the Q-target as described in Appendix C.3.2. They apply the same index for both main Q-value and target Q-value computation, which we refer to as *dependent Q-target computation*. In contrast, `HyperAgent` employs independent Q-target computation, where it independently samples different indices to compute the target Q-value, a strategy supported by theoretical analysis. We contrast these two Q-target computation methods within the `HyperAgent` framework and introduce the `HyperAgent` w. dependent Q-target. As depicted in Figure 15, `HyperAgent` w. dependent Q-target demonstrates improved results, prompting further analysis of the underlying reasons in future research.

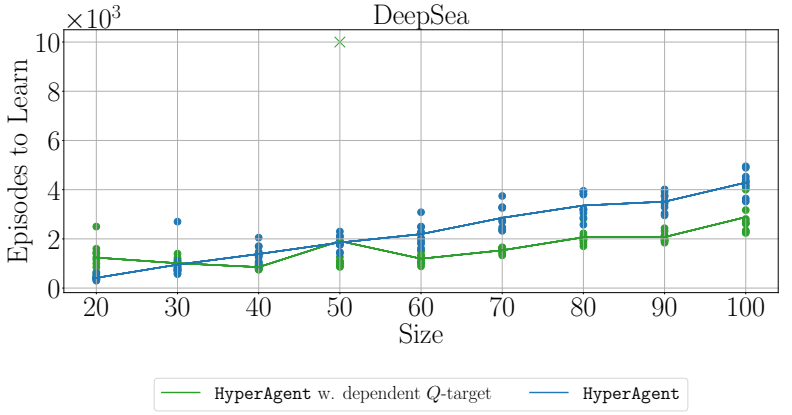


Figure 15. Ablation results on different Q-target computation methods. The method of *dependent Q-target computation* can achieve better results inspiring future research.

F. Additional Results on Atari

We demonstrated the efficiency of HyperAgent in handling data and computation in Section 5.2. Here, we present comprehensive results for each environment to further establish the superiority of our approach.

Detailed results on each game. We adhere to the evaluation protocol described in Appendix B.1 to obtain the best score achieved in each game with 2M steps of HyperAgent, as presented in Table 7. The sources of the compared baseline results can also be found in Appendix B.2.

Game	Random	Human	DDQN†	DER	Rainbow	HyperDQN	HyperAgent
Alien	227.8	7127.7	722.7	1642.2	1167.1	862.0	**1830.2**
Amidar	5.8	1719.5	61.4	476.0	374.0	140.0	**800.4**
Assault	222.4	742.0	815.3	488.3	2725.2	494.2	**3276.2**
Asterix	210.0	8503.3	2471.1	1305.3	3213.3	713.3	2370.2
BankHeist	14.2	753.1	7.4	460.5	411.1	272.7	430.3
BattleZone	2360.0	37187.5	3925.0	19202.5	19379.7	11266.7	**29399.0**
Boxing	0.1	12.1	26.7	1.7	69.9	6.8	**74.0**
Breakout	1.7	30.5	2.0	6.5	137.3	11.9	54.8
ChopperCommand	811.0	7387.8	354.6	1488.9	1769.4	846.7	**2957.2**
CrazyClimber	10780.5	35829.	53166.5	36311.1	110215.8	42586.7	**121855.8**
DemonAttack	152.1	1971.0	1030.8	955.3	45961.3	2197.7	**5852.0**
Freeway	0.0	29.6	5.1	32.8	32.4	30.9	32.2
Frostbite	65.2	4334.7	358.3	3628.3	3648.7	724.7	**4583.9**
Gopher	257.6	2412.5	569.8	742.1	4938.0	1880.0	**7365.8**
Hero	1027.0	30826.4	2772.9	15409.4	11202.3	9140.3	12324.7
Jamesbond	29.0	302.8	15.0	462.1	773.1	386.7	**951.6**
Kangaroo	52.0	3035.0	134.9	8852.3	6456.1	3393.3	8517.1
Krull	1598.0	2665.5	6583.3	3786.7	8328.5	5488.7	8222.6
KungFuMaster	258.5	22736.3	12497.2	15457.0	25257.8	12940.0	23821.2
MsPacman	307.3	6951.6	1912.3	2333.7	1861.1	1305.3	**3182.3**
Pong	-20.7	14.6	-15.4	20.6	5.1	20.5	20.5
PrivateEye	24.9	69571.3	37.8	900.9	100.0	64.5	171.9
Qbert	163.9	13455.0	1319.4	12345.5	7885.3	5793.3	12021.9
RoadRunner	11.5	7845.0	3693.5	14663.0	33851.0	7000.0	28789.4
Seaquest	68.4	42054.7	367.6	662.0	1524.7	370.7	**2732.4**
UpNDown	533.4	11693.2	3422.8	6806.3	39187.1	4080.7	19719.2

Table 7. The averaged score over 200 evaluation episodes for the best policy in hindsight (after 2M steps) for 26 Atari games. The performance of the random policy and the human expert is from dqn_zoo Quan & Ostrovski (2020).

We also present the relative improvement of HyperAgent in comparison to other baselines for each game, which is determined by the given following equation as per (Wang et al., 2016):

$$\text{relative improvement} = \frac{\text{proposed} - \text{baseline}}{\max(\text{human}, \text{baseline}) - \text{human}}.$$

Our classification of environments into three groups, namely "hard exploration (dense reward)", "hard exploration (sparse reward)" and "easy exploration", is based on the taxonomy proposed by (Bellemare et al., 2016). The overall results are illustrated in Figures 16 to 19.

HyperAgent algorithm exhibits significant improvement compared to DDQN†, DER, and HyperDQN in environments with "easy exploration", and overall it performs better in all environments. This indicates that HyperAgent has better generalization and exploration abilities. On the other hand, when compared to Rainbow, our algorithm performs better in environments which are in the group of "hard exploration (dense reward)", demonstrating its superior exploration

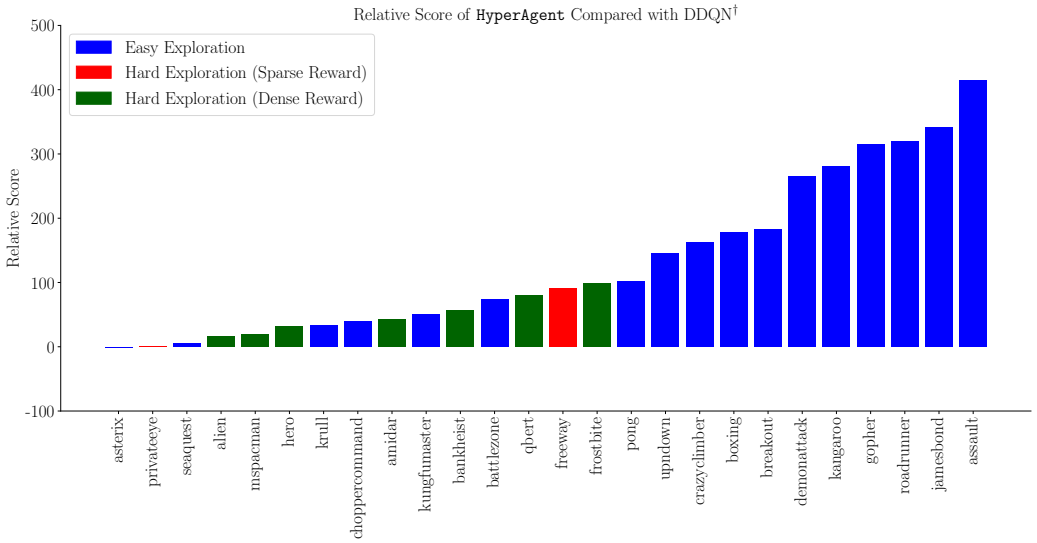


Figure 16. Relative improvement of HyperAgent compared with DDQN†

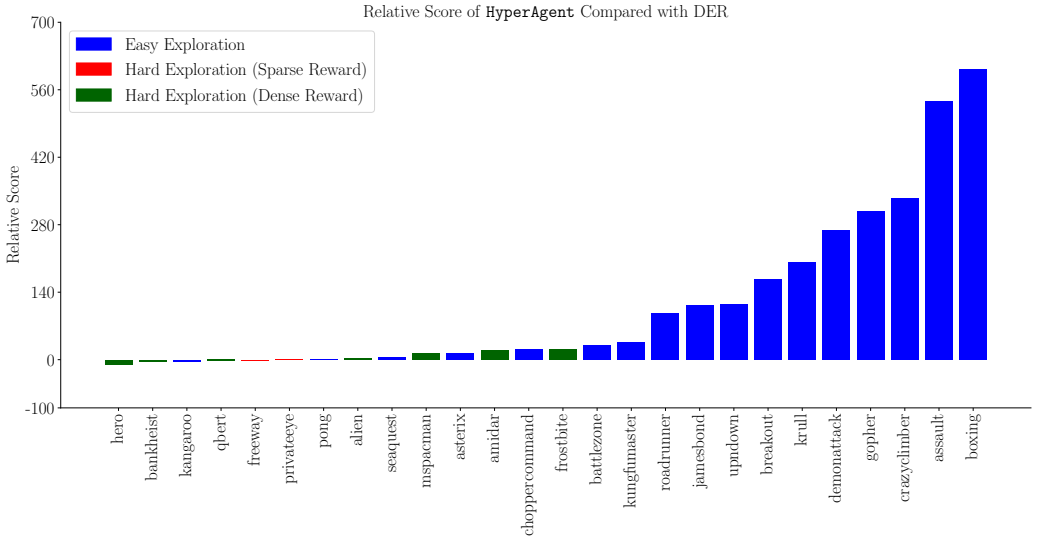


Figure 17. Relative improvement of HyperAgent compared with DER

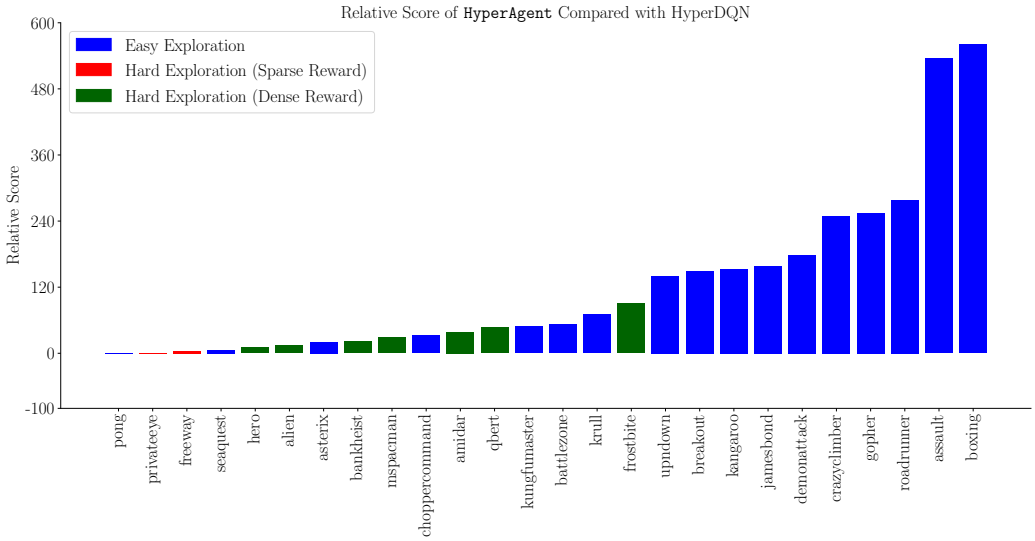


Figure 18. Relative improvement of HyperAgent compared with HyperDQN

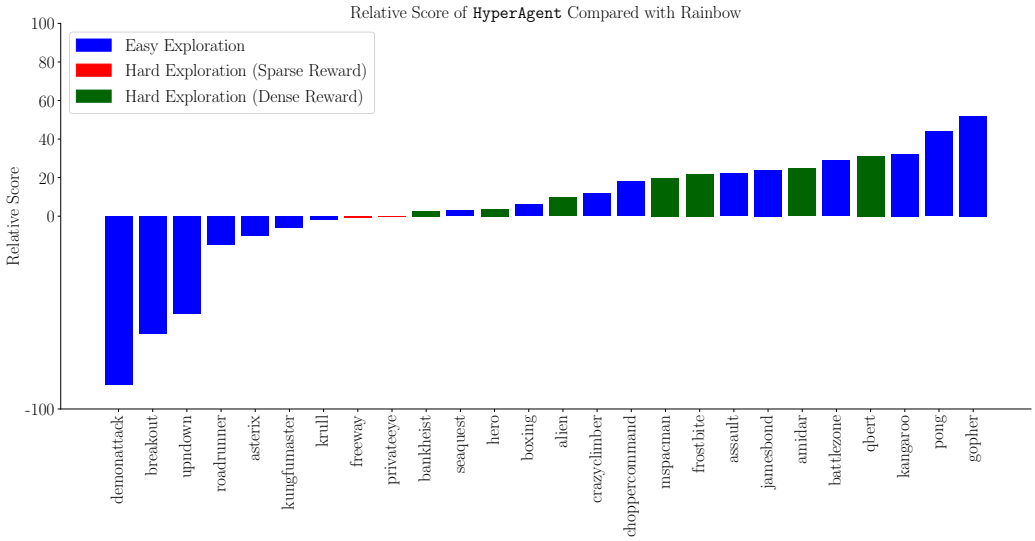


Figure 19. Relative improvement of HyperAgent compared with Rainbow

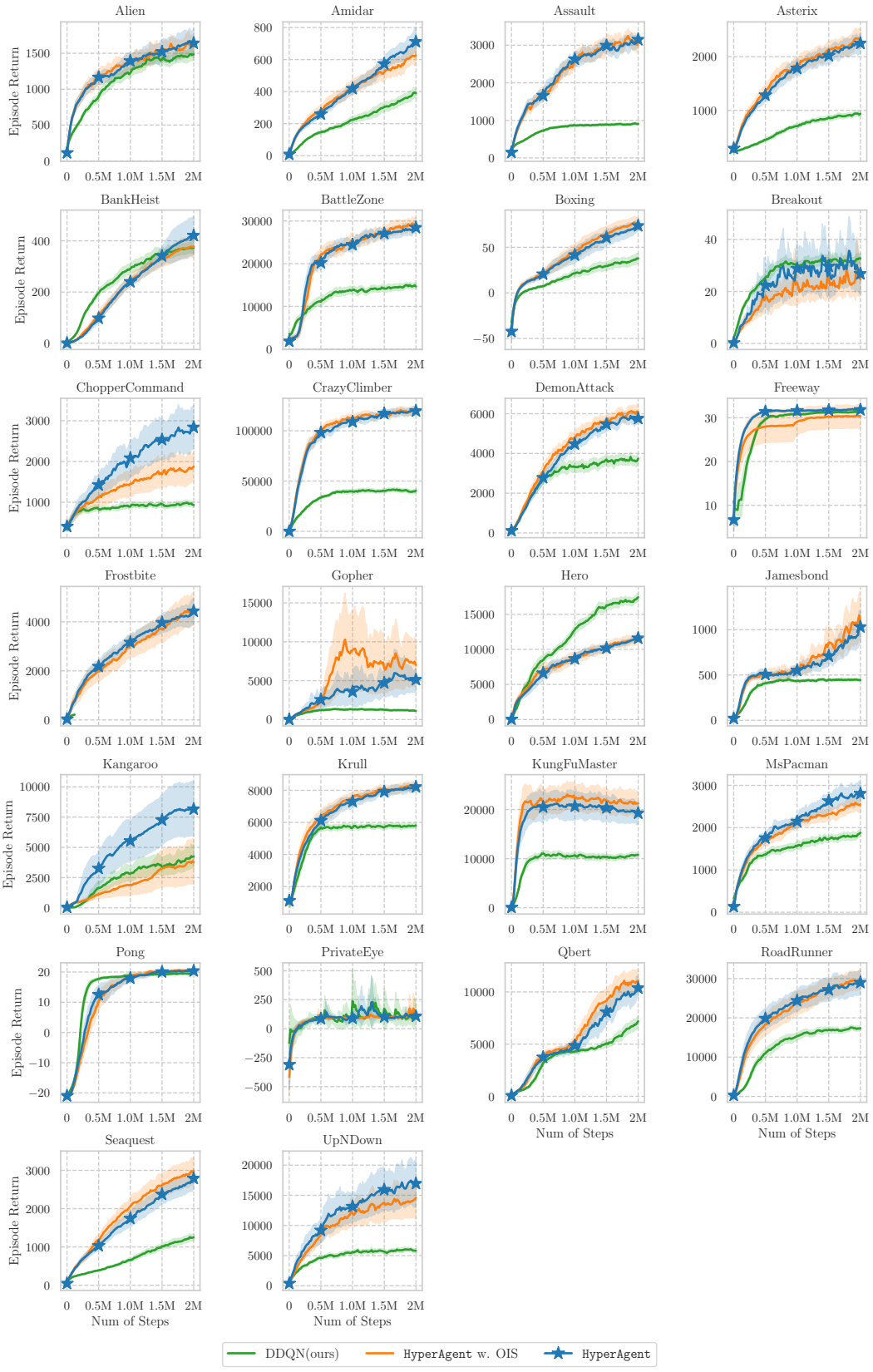


Figure 20. Learning curve for each game.

capabilities. In the case of Freeway, which belongs to the "hard exploration (sparse reward)" group, both HyperAgent and Rainbow achieve similar optimal scores (as shown in Table 7). However, HyperAgent demonstrates faster convergence, as evidenced in Figure 4. Overall, HyperAgent showcases better generalization and exploration efficiency than other baselines.

We also evaluate the OIS method across the 26 Atari games, as illustrated in Table 8. Our findings indicate that the OIS method does not generate significant differences in complex networks like Convolutional layers.

Method	IQM	Median	Mean
HyperAgent	1.22 (1.15, 1.30)	1.07 (1.03, 1.14)	1.97 (1.89, 2.07)
HyperAgent w. OIS	1.15 (1.09, 1.22)	1.12 (1.02, 1.18)	2.02 (1.91, 2.16)

Table 8. Comparable results achieved using the OIS method in Atari games. The data in parentheses represent the 95% confidence interval.

Furthermore, we present the learning curve for each game in Figure 20. It is evident that HyperAgent has demonstrated superior performance compared to DDQN(ours), attributed to the integration of hypermodel that enhances exploration. Moreover, it is worth highlighting that the learning curve of HyperAgent continues to rise in specific environments, indicating that it can achieve even better performance with additional training.

Additional results about exploration on Atari. To further demonstrate the exploration efficiency of HyperAgent, we compare it with additional baselines, including AdamLMCDQN (Ishfaq et al., 2024), LangevinAdam (Dwaracherla & Van Roy, 2021), HyperDQN (Li et al., 2022) and our variant HyperAgent w. OIS. As depicted in Figure 21, both HyperDQN and HyperAgent demonstrate improved results, indicating that applying hypermodel can lead to better posterior approximation and consequently enhance exploration.

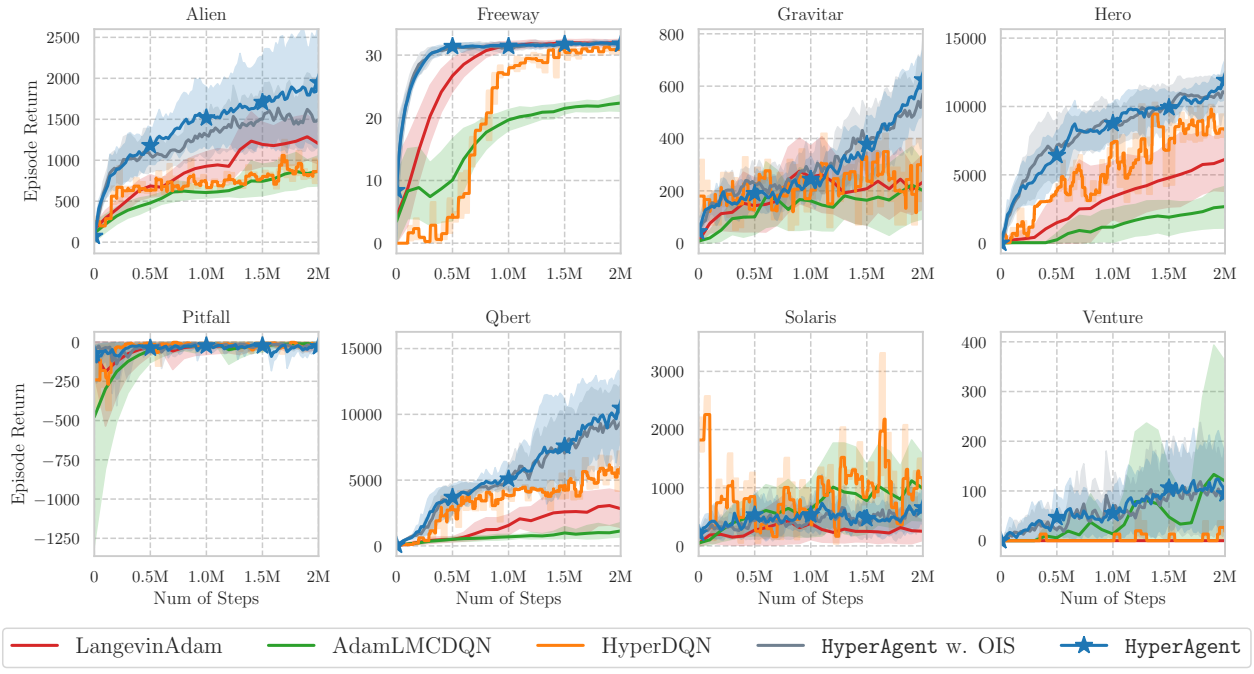


Figure 21. Comparative results on 8 hardest exploration games with more baselines.

Additional results on other 29 Atari games. To further showcase the robustness and scalability of HyperAgent, we conduct experiments on all 55 Atari games using identical settings (see Table 4). We present the best score achieved in these 29 environments using 2M steps in Table 9, with 5 seeds repeated for each environment. HyperAgent outperforms others in over half of these 29 environments, delivering top performance in 31 out of 55 Atari games.

Game	Random	Human	DDQN†	Rainbow	HyperDQN	HyperAgent
Asteroids	210.0	47388.7	520.8	969.4	1044.7	**1176.6**
Atlantis	12850.0	29028.1	5771.3	528642.7	370160.0	**793851.9**
BeamRider	363.9	16926.5	464.4	10408.0	981.9	6749.2
Berzerk	123.7	2630.4	566.6	822.7	336.7	**840.5**
Bowling	23.1	160.7	12.6	27.7	14.8	**61.3**
Centipede	2090.9	12017.0	4343.2	5352.1	2071.5	4121.9
Defender	2874.5	18688.9	2978.4	25457.1	4063.3	21422.7
DoubleDunk	-18.6	-16.4	-21.2	-2.2	-17.7	**-1.7**
Enduro	0.0	860.5	338.7	1496.6	201.0	1223.0
FishingDerby	-91.7	-38.7	-78.1	-22.4	-74.3	**-0.7**
Gravitar	173.0	3351.4	2.6	372.0	300.0	**629.5**
IceHockey	-11.2	0.9	-12.4	-7.0	-11.6	**-3.8**
NameThisGame	2292.3	8049.0	5870.3	9547.2	3628.0	5916.4
Phoenix	761.4	7242.6	3806.2	6325.0	3270.0	4941.9
Pitfall	-229.4	6463.7	-55.5	-0.1	-13.0	-11.2
Riverraid	1338.5	17118.0	3406.2	5627.9	4233.3	**6896.3**
Robotank	2.2	11.9	7.8	22.4	3.1	**37.2**
Skiing	-17098.1	-4336.9	-22960.7	-16884.8	-29975.0	**-11654.4**
Solaris	1236.3	12326.7	390.2	1185.9	1173.3	941.3
SpaceInvaders	148.0	1668.7	356.1	742.2	425.3	**762.2**
StarGunner	664.0	10250.0	346.3	11142.3	1113.3	6135.7
Tennis	-23.8	-8.3	-10.1	0.0	-17.0	-1.4
TimePilot	3568.0	5229.2	2204.6	3763.8	3106.7	**6006.1**
Tutankham	11.4	167.6	108.6	104.1	93.0	**116.2**
Venture	0.0	1187.5	21.5	0.0	26.7	**163.9**
VideoPinball	16256.9	17667.9	10557.5	49982.2	24859.2	29674.8
WizardOfWor	563.5	4756.5	275.7	3770.9	1393.3	**4019.5**
YarsRevenge	3092.9	54576.9	10485.5	10195.6	4263.1	**28805.5**
Zaxxon	32.5	9173.3	2.1	7344.1	3093.3	**7688.1**

Table 9. The evaluated score of other 29 games from ALE suite.