Model-based Lifelong Reinforcement Learning with Bayesian Exploration

Haotian Fu, Shangqun Yu, Michael Littman, George Konidaris
Department of Computer Science, Brown University
{hfu7,syu68,mlittman,gdk}@cs.brown.edu

Abstract

We propose a model-based lifelong reinforcement-learning approach that estimates a hierarchical Bayesian posterior distilling the common structure shared across different tasks. The learned posterior combined with a sample-based Bayesian exploration procedure increases the sample efficiency of learning across a family of related tasks. We first derive an analysis of the relationship between the sample complexity and the initialization quality of the posterior in the finite MDP setting. We next scale the approach to continuous-state domains by introducing a Variational Bayesian Lifelong Reinforcement Learning algorithm that can be combined with recent model-based deep RL methods, and that exhibits backward transfer. Experimental results on several challenging domains show that our algorithms achieve both better forward and backward transfer performance than state-of-the-art lifelong RL methods[1].

1 Introduction

Reinforcement learning (RL) [42; 26] has been successfully applied to solve challenging individual tasks such as learning robotic control [11] and playing Go [38]. However, the typical RL setting assumes that the agent solves exactly one task, which it has the opportunity to interact with repeatedly. In many real-world settings, an agent instead experiences a collection of distinct tasks that arrive sequentially throughout its operational lifetime; learning each new task from scratch is inefficient, but treating them all as a single task will fail. Therefore, recent research has focused on algorithms that enable agents to learn across multiple, sequentially posed tasks, leveraging knowledge from previous tasks to accelerate the learning of new tasks. This problem setting is known as *lifelong reinforcement learning* [7; 50; 25]. The key questions in lifelong RL research are: How can an algorithm exploit knowledge gained from past tasks to quickly adapt to new tasks (forward transfer), and how can data from new tasks help the agent perform better on previously learned tasks (backward transfer)?

We propose to address these problems by extracting the common structure existing in previously encountered tasks so that the agent can quickly learn the dynamics specific to the new tasks. We consider lifelong RL problems that can be modeled as hidden-parameter MDPs or *HiP-MDPs* [10; 27], where variations among the true task dynamics can be described by a set of hidden parameters. Our algorithm goes further than previous work in both lifelong learning and HiP-MDPs by **1)** Separately modeling epistemic and aleatory uncertainty over different levels of abstraction across the collection of tasks: the uncertainty captured by a world-model distribution describing the probability distribution over tasks, and the uncertainty captured by a task-specific model of the (stochastic) dynamics within a single task. To enable more accurate sequential knowledge transfer, we separate the learning process for these two quantities and maintain a hierarchical Bayesian posterior that approximates them. **2)** Performing Bayesian exploration enabled by the hierarchical posterior: The method lets the agent act optimistically according to models sampled from the posterior, and thus increases sample efficiency.

[1] Code repository available at `https://github.com/Minusadd/VBLRL`.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Specifically, we propose a model-based lifelong RL approach with Bayesian exploration that estimates a Bayesian world-model posterior that distills the common structure of previous tasks, and then uses this between-task posterior as a within-task prior to learn a task-specific model in each subsequent task. The learned hierarchical posterior model combined with sample-based Bayesian exploration procedures can increase the sample efficiency of learning. We first derive an explicit performance bound that shows that the task-specific model requires fewer samples to become accurate as the world-model posterior approaches the true underlying world-model distribution for the discrete case. We further develop Variational Bayesian exploration for Lifelong RL (VBLRL), a more scalable version that uses variational inference to approximate the distribution and leverages Bayesian Neural Networks (BNNs) [17; 3] to build the hierarchical Bayesian posterior. VBLRL provides a novel way to separately estimate different kinds of uncertainties in the HiP-MDP setting. Based on the same framework, we also propose a backward transfer version of VBLRL that is able to provide improvements for previously encountered tasks. Our experimental results on a set of challenging domains show that our algorithms achieve better both forward and backward transfer performance than state-of-the-art lifelong RL algorithms when given only limited interactions with each task.

2 Background

RL is the problem of maximizing the long-term expected reward of an agent interacting with an environment [42]. We usually model the environment as a Markov Decision Process or *MDP* [36], described by a five tuple: $\langle S, A, R, T, \gamma \rangle$, where S is a finite set of states; A is a finite set of actions; $R : S \times A \mapsto [0, 1]$ is a reward function, with a lower and upper bound of 0 and 1; $T : S \times A \mapsto \Pr(S)$ is a transition function, with $T(s'|s, a)$ denoting the probability of arriving in state $s' \in S$ after executing action $a \in A$ in state s; and $\gamma \in [0, 1)$ is a discount factor, expressing the agent's preference for delayed over immediate rewards.

An MDP is a suitable model for the task facing a single agent. In the lifelong RL setting, the agent instead faces a series of tasks $m_1, ..., m_n$, each of which can be modeled as an MDP: $\langle S^{(i)}, A^{(i)}, R^{(i)}, T^{(i)}, \gamma^{(i)} \rangle$. For lifelong RL problems, the performance of a specific algorithm is usually evaluated based on both forward transfer and backward transfer results [30]:

- *Forward transfer*: the influence that learning task t has on the performance in future task $k \succ t$.
- *Backward transfer*: the influence that learning task t has on the performance in earlier tasks $k \prec t$.

A key question in the lifelong setting is how the series of task MDPs are related; we model the collection of tasks as a HiP-MDP, where a family of tasks is generated by varying a latent task parameter ω drawn for each task according to the world-model distribution P_Ω. Each setting of ω specifies a unique MDP, but the agent neither observes ω nor has access to the function that generates the task family. The dynamics $T(s'|s, a; \omega_i)$ and reward function $R(r|s, a; \omega_i)$ for task i then depend on $\omega_i \in \Omega$, which is fixed for the duration of the task. The tasks are i.i.d. sampled from a fixed distribution and arrive one at a time.

3 Related work

The first category of lifelong RL algorithms learns a single model that encourages transfer across tasks by modifying objective functions. EWC [28] imposes a quadratic penalty that pulls each weight back towards its old values by an amount proportional to its importance for performance on previously-learned tasks to avoid forgetting. There are several extensions of this work based on the core idea of modifying the form of the penalty [29; 53; 33]. Another category of lifelong RL methods uses multiple models with shared parameters and task-specific parameters to avoid or alleviate the catastrophic problem [5; 24; 31]. The drawback of this method is that it is hard to incorporate the knowledge learned from previous tasks during initial training on a new task [31]. Nagabandi et al. [32] introduce a model-based continual learning framework based on MAML, but they focus on discovering when new tasks were encountered without access to task indicators.

Published HiP-MDP methods use Gaussian Processes [10] or Bayesian neural networks [27] to find a single model that works for all tasks, which may trigger catastrophic forgetting [5; 24]. Meta-RL [47; 12] and multi-task RL [35; 43] settings also attempt to accelerate learning by transferring knowledge from different tasks. Some work employs the MAML framework with Bayesian methods

to learn a stochastic distribution over initial parameters [51; 16; 13]. Other work uses the collected trajectories to infer the hidden parameter, which is taken as an additional input when computing the policy [37; 56; 14]. Our method, however, focuses on problems where the tasks arrive sequentially instead of having a large number of tasks available at the beginning of training. This sequential setting makes it hard to accurately infer the hidden parameters, but opens the door for algorithms that support backward transfer.

Some prior work uses Bayesian methods in RL to quantify uncertainty over initial MDP models [15; 1; 18]. Several algorithms start from the idea of sampling from a posterior over MDPs for Bayesian RL, maintaining Bayesian posteriors and sampling one complete MDP [41; 49] or multiple MDPs [2]. Instead of focusing on single-task RL, our algorithm aims to find a posterior over the common structure among multiple tasks. Wilson et al. [49] uses a hierarchical Bayesian infinite mixture model to learn a strong prior that allows the agent to rapidly infer the characteristics of a new environment based on previous tasks. However, it only infers the category label of a new MDP and only works in discrete settings.

4 Model-based Lifelong Reinforcement Learning

Our approach is built upon two main intuitions: First, transferring the transition model instead of policy/value function leads to more efficient usage of the data when "finetuning" on a new task. As we show empirically in Section 5.1, although some model-free lifelong RL algorithms perform better than the proposed model-based method in single-task cases, in lifelong RL setting of the same task type the model-based method is still able to achieve comparable/better performance with only half the amount of data. Secondly, with a model that is able to capture different levels of the uncertainty within HiP-MDPs, an agent can employ sample-based Bayesian exploration to further improve sample-efficiency.

The model underlying our approach is a hierarchical Bayesian posterior over task MDPs controlled by the hidden parameter ω. Intuitively, we maintain probability distributions that separately capture two categories of uncertainty within lifelong learning tasks: The *world-model posterior* $P(\omega)$ captures the epistemic uncertainty of the world-model distribution over all future and past tasks $m_1, \cdots, m_n$ controlled by the hidden parameter $\omega_1, \cdots, \omega_n \sim P_\Omega$. As the learner is exposed to more and more tasks, this posterior should converge to the world-model distribution P_Ω. The *task-specific posterior* $P(\omega_i)$ captures the epistemic uncertainty of the current task m_i (Throughout the paper we will often write i for simplicity.). As the learner is exposed to more and more transitions within the task, this posterior should approach the true distribution corresponding to ω_i, i.e. peaking at the true ω_i for this specific task i, leaving only the aleatoric uncertainty of transitions within the task, which is independent of other tasks. Each time the agent encounters a new task, we initialize the task-specific model using the world-model posterior and further train it with data collected only from the new task. One of our key insights here is that the sample complexity of learning a new task will decrease as the initial prior of task-specific model approaches the true underlying distribution of the transition function. Thus, the agent can learn new tasks faster by exploiting knowledge common to previous tasks, thereby exhibiting positive forward transfer.

Specifically, we model the task-specific posterior via the transition dynamics using $p(s^{t+1}, r^t | s^t, a^t; \omega_i)$. The task-specific posterior, given a new state–action pair from task i, can be rewritten via Bayes' rule:

$$P(\omega_i | D_i^t, a^t, s^{t+1}, r^t) = \frac{P(\omega_i | D_i^t) P(s^{t+1}, r^t | D_i^t, a^t; \omega_i)}{P(s^{t+1}, r^t | D_i^t, a^t)}, \tag{1}$$

where $D_i^t = \{s^1, a^1, \cdots, s^t\}$ is the agent's history of task i until time step t. The world-model posterior, given the new data from task i, can be rewritten as:

$$P(\omega | D_{1:i}) = \frac{P(\omega | D_{1:i-1}) P(D_i | D_{1:i-1}; \omega)}{P(D_i | D_{1:i-1})}, \tag{2}$$

where $D_{1:i}$ denotes the agent's history with all the experienced tasks $1 \sim i$ until current task i. In particular, each time when the agent faces a new task i and has not started updating its task specific posterior yet (that is, $D_i^t = \varnothing$), we first use the world-model posterior to initialize the task-specific prior: $P(\omega_i | D_i^t) = P(\omega | D_{1:i})$. The world-model distribution aims to approximate the underlying

P_Ω. The task-specific distribution aims to approximate the distribution that peaks at the true ω_i for this specific task i.

In Section 4.1, we derive a sample complexity bound in the finite MDP case which explicitly show how the distance between the distribution of a task's true transition model and our task-specific model's prior initialized by the parameters of the world-model posterior will affect the learning efficiency of a new task. Then, in Section 4.2 and 4.3 we extend our high-level idea to a scalable version that can be combined with recent model-based RL approaches and exhibits positive forward & backward transfer.

4.1 Sample Complexity Analysis

In this subsection, we consider the finite MDP setting and use a standard Bayesian exploration algorithm BOSS [2] as a single-task baseline. Note that BOSS creates optimism in the face of uncertainty as the agent can choose actions based on the highest performing transition of the K models sampled, which drives exploration. We included explanations of the finite MDP version of our algorithm BLRL (Bayesian Lifelong RL) based on BOSS in appendix C and simple experiments on Gridworlds in appendix L.

BLRL uses the world-model posterior $P(\omega|D_{1:i-1})$ learned from previous $i-1$ tasks to initialize (copy distribution) the task-specific prior of $P(\omega_i)$ of new task i, aiming to decrease the number of samples needed to learn an accurate task-specific posterior. Our analysis focuses on how the properties of the Bayesian prior affect the sample complexity of learning each specific task.

Let $\pi(\omega_i)$ denote the prior distribution on the parameter space Γ. We consider a set of transition-probability densities $p(\cdot|\omega_i) = p(s^{t+1}, r^t|s^t, a^t, \omega_i)$ indexed by ω_i, and the true underlying density q. We also denote the model family $\{p(\cdot|\omega_i) : \omega_i \in \Gamma\}$ by the same symbol Γ.

Lemma 4.1. *The task-specific posterior in Equation 1 can be regarded as the probability density $g(\omega_i)$ with respect to π that attains the infimum of:*

$$R_n(g) = \mathbb{E}_\pi g(\omega_i) \sum_{t=1}^{T} \ln \frac{q(s^{t+1}, r^t|D_i^t, a^t)}{p(s^{t+1}, r^t|D_i^t, a^t; \omega_i)} + D_{KL}(gd\pi||d\pi). \tag{3}$$

$\inf R_n(g)$ controls the complexity of the density estimation process for $g(\omega_i)$. Intuitively, Lemma 4.1 converts the Bayesian posterior into an information theoretical minimizer that allows us to further investigate the relationship between the properties of the Bayesian prior and the risk/complexity of attaining the posterior.

Proposition 4.2. *Define the **prior-mass radius** of the transition-probability densities as:*

$$d_\pi = \inf\{d : d \geq -\ln \pi(\{p \in \Gamma : D_{KL}(q||p) \leq d\})\}. \tag{4}$$

Intuitively, this quantity measures the distance between the Bayesian prior of the task-specific model and the true underlying task-specific distribution. Then, we adopt the same settings in Zhang [55]: $\forall \rho \in (0,1)$ and $\eta \geq 1$, let

$$\varepsilon_n = (1 + \frac{1}{n})\eta d_\pi + (\eta - \rho)\varepsilon_{upper,n}((\eta - 1)/(\eta - \rho)), \tag{5}$$

*where $\varepsilon_{upper,n}$ is the **critical upper-bracketing radius** [55]. The decay rate of $\varepsilon_{upper,n}$ controls the consistency of the Bayesian posterior distribution [2]. Let $\rho = \frac{1}{2}$, we have for all states and actions, $h \geq 0$ and $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\pi_n\Big(\Big\{p \in \Gamma : ||p - q||_1^2/2 \geq \frac{2\varepsilon_n + (4\eta - 2)h}{\delta/4}\Big\}\Big|X\Big) \leq \frac{1}{1 + e^{nh}}, \tag{6}$$

where $\pi_n(\omega_i)$ denotes the posterior distribution over ω after collecting n samples, X denotes the n collected samples. It functions the same as D_i^t in Equation 1.

Similar to BOSS, for a new MDP $m^* \sim M$ with hidden parameters ω_{m^*}, we can define the Bayesian concentration sample complexity for the task-specific posterior: $f(s, a, \epsilon_0, \delta_0, \rho_0)$, as the minimum number u such that, if u IID transitions from (s, a) are observed, then, with probability at least $1 - \delta_0$,

$$Pr_{m \sim posterior}(||T(\cdot|s, a, \omega_m) - T(\cdot|s, a, \omega_{m^*})||_1 < \epsilon_0) \geq 1 - \rho_0. \tag{7}$$

Intuitively, the inequality means that for a model with the hidden parameter sampled from the learned posterior distribution, the probability that that it is within ϵ_0 far away from the true model is larger than $1 - \rho_0$.

Lemma 4.3. *Assume the posterior of each task is consistent (that is, $\varepsilon_{upper,n} = o(1)$) and set $\eta = 2$, then the Bayesian concentration sample complexity for the task-specific posterior $f(s, a, \epsilon, \delta, \rho) = O\left(\frac{d_\pi + \ln \frac{1}{\rho}}{\epsilon^2 \delta - d_\pi}\right)$.*

Proof (sketch). This bound can be derived by directly combining Lemma 4.2 and Equation 7. □

The above lemma suggests an upper bound of the Bayesian concentration sample complexity using the prior-mass radius. We can further combine this result with PAC-MDP theory [39] and derive the sample complexity of the algorithm for each new task.

Proposition 4.4. *For each new task, set the sample size $K = \Theta(\frac{S^2 A}{\delta} \ln \frac{SA}{\delta})$ and the parameters $\epsilon_0 = \epsilon(1-\gamma)^2, \delta_0 = \frac{\delta}{SA}, \rho_0 = \frac{\delta}{S^2 A^2 K}$, then, with probability at least $1-4\delta$, $V^t(s^t) \geq V^*(s^t) - 4\epsilon_0$ in all but $\tilde{O}(\frac{S^2 A^2 d_\pi}{\delta \epsilon^3 (1-\gamma)^6})$ steps, where $\tilde{O}(\cdot)$ suppresses logarithmic dependence.*

Proof (sketch). The central part of the proof is Proposition 4.2 (detailed proof in appendix), and the re- maining parts are exactly the same as those for BOSS. In general, the proof is based on the PAC-MDP theorem [40] combined with the new bound for the Bayesian concentration sample complexity we derived in Lemma 2. For each new task, the main difference between BLRL and BOSS is that we use the world-model posterior to initialize the task-specific posterior, which results in a new sample complexity bound with d_π . □

The result formalizes how the sample complexity of the lifelong RL algorithm will change with respect to the initialization quality of the posterior: if we put a larger prior mass at a density close to the true q such that d_π is small, the sample efficiency of the algorithm will increase accordingly. In other words, the sample complexity of our algorithm drops proportionally to d_π, which is the distance between the Bayesian prior of the task-specific model initialized by the parameters of the world-model posterior and the true underlying task-specific distribution. We provide a illustrative example in the appendix P.

4.2 Variational Bayesian Lifelong RL

The intuition from the last section is that, if we initialize the task-specific distribution with a prior that is close to the true distribution, sample complexity will decrease accordingly. To scale our approach, we must find a efficient way to explicitly approximate these distributions. We propose a practical approximate algorithm, VBLRL, that uses neural networks and variational inference [21].

We choose Bayesian neural networks (BNN) to approximate the posterior. The intuition is that, in the context of stochastic outputs, BNNs naturally approximate the hierarchical Bayesian model since they also maintain a learnable distribution over their weights and biases [17; 23]. We use the uncertainty embedded in the weights and biases of networks to capture the epistemic uncertainty introduced by hidden parameters of different tasks, while we also set the outputs of the neural networks to be stochastic to capture the aleatory uncertainty within each specific task. In our case, the BNN weights and biases distribution $q(\omega; \phi)$ (a distribution over ω but parameterized by ϕ) can be modeled as fully factorized Gaussian distributions [3]:

$$q(\omega; \phi) = \prod_{j=1}^{|\Omega|} \mathcal{N}(\omega_j | \mu_j, \sigma_j^2), \tag{8}$$

where $\phi = \{\mu, \sigma\}$, and μ is the Gaussian's mean vector while σ is the covariance matrix diagonal.

We maintain a world-model BNN across all the tasks and a task-specific BNN for each task. The input for all the BNNs is a state–action pair, and the output are the mean and variance of the prediction for reward and next state. Then, the posterior distribution over the model parameters can be computed leveraging variational lower bounds [22; 23]:

$$\phi_t = \arg\min_{\phi} \Big[D_{KL}[q(\omega; \phi) || p(\omega)] - \mathbb{E}_{\omega \sim q(\cdot; \phi)} [\log p(s^{t+1}, r^t | D^t, a^t; \omega)] \Big], \tag{9}$$

where $p(\omega)$ represents the fixed prior distribution of ω, also recall that $D^t = \{s^1, a^1, r^1, \cdots, s^t\}$. In VBLRL, $p(\omega)$ for the task-specific distribution is initialized by the world-model distribution, while $p(\omega)$ of the world-model distribution is simply set to a Gaussian. (That could be improved with more informed task family knowledge.) The second term on the right hand side can be approximated through $\frac{1}{N}\sum_{i=1}^{N} \log p(s^{t+1}, r^t | D^t, a^t; \omega_i)$ with N samples from $\omega_i \sim q(\phi)$. This optimization can be performed in parallel for each s, keeping ϕ_{t-1} fixed. Details about our BNN structure can be found in appendix B.

Once we have the world-model as well as the task-specific posterior model, we can do posterior sampling to drive Bayesian exploration. Osband et al. [34] and Rakelly et al. [37] already show the benefit of posterior sampling for improving sample efficiency in single-task RL and meta-RL settings. Here, instead of sampling from the posterior of value functions or latent context representations, we directly sample from the posterior of the transition model and choose the optimal action based on the transition samples. Then we use the collected samples to update the posterior and do sampling again. The agent thus continually do Bayesian exploration and acts more and more optimally as the posterior distribution narrows. We provide the framework of VBLRL in Figure 1. The outer loop (yellow rectangle) stands for the loop of training world-model BNN across tasks (noted at the bottom-right corner), the inner loop (cyan rectangle) stands for the training of the task-specific BNN within each specific task.We employ our posterior knowledge models in the context of a model-based RL method. When encountering a new task, VBLRL first uses the model parameters (that is, $\{\mu, \sigma\}$ of weights and biases of BNN) from the general knowledge model to initialize the task-specific posterior network (directly copy parameters). We sample transitions from the task-specific posterior and select actions based on the generated transitions. The detailed algorithm is summarized in Algorithm 1. Specifically, for planning, at each step we begin by creating P particles from the current state $s^p_{\tau=t} = s_t \forall p$. Then, we sample N candidate action sequences $a_{t:t+T}$ from a learnable distribution. These two steps are the same as PETS [8]. Then we propagate the state–action pairs using the learned task-specific model $p_{m_i}(\cdot|s, a)$ (BNN) and use the cross entropy method [4] to update the sampling distribution to make the sampled action sequences close to previous action sequences that achieved high reward. We further calculate the cumulative reward estimated (via the learned model) for previously sampled sequences and select the current action based on the mean of that distribution. By sampling from the task-specific posterior at each step, the agent explores in a temporally extended and diverse manner.

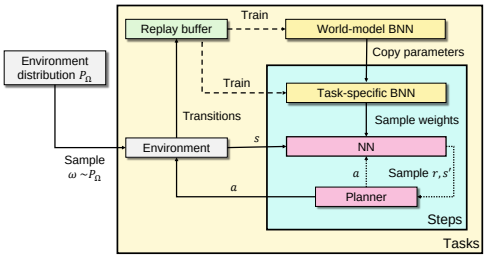

Figure 1: Variational Bayesian Lifelong RL (VBLRL). "NN" is the network using fixed parameters sampled from the weight distribution of the task-specific BNN. When the agent does CEM planning, we let it propagate the state particles using several NNs that use different network parameters, all sampled from the BNN to achieve randomness in transitions.

For each specific task, we use the task-specific BNN to plan during forward learning. They are updated with only the data from the current task thus avoid catastrophic forgetting in the forward transfer process. The world model is updated using the data from **all** the visited tasks. The intuition is to guide the two posteriors to separately learn two categories of uncertainty within lifelong learning tasks. The world-model posterior captures the epistemic uncertainty of the general knowledge distribution (shared across all tasks controlled by the hidden parameters) via the internal variance of world-model BNN. As the learner is exposed to more and more tasks, the posterior should converge to P_Ω. The task-specific posterior captures the epistemic uncertainty of the current task m_i, which comes from the aleatory uncertainty of the world model when generating ω_i for a new task, via the internal variance of task-specific BNN. We provide additional illustration in Figure 8. In general, we used the world-model BNN to encode P_Ω over a family of tasks, and the task-specific BNN to encode $P(\omega_i)$, which should peak at the specific true ω_i sampled for the current task.

Note that in single-task cases, other CEM-based algorithms like PETS usually maintain a set of neural networks using the same training data, and sample action sequences from each of the neural nets to achieve randomness in transitions. However, in lifelong RL settings, it is unrealistic to maintain

(≥ 30) models for each task encountered. Our usage of BNNs avoids such problems as we only have to train one neural network using the same data for each task, and we can sample an unlimited number of different action sequences to cover more possibilities as needed. In PETS, the epistemic uncertainty is estimated via the variance of the output mean of different neural networks, while, in VBLRL, it is estimated via the variance of the weights and biases distribution of the BNN.

4.3 Backward Transfer of Variational Bayesian Lifelong RL

In our lifelong RL setting, the agent interacts with each task for only a limited number of episodes and the task-specific model stops learning when the next task is initiated. As a result, there may exist portions of the transition dynamics in which model uncertainty remains high. However, as the world-model posterior continues to train on new tasks, it gathers more experience in the whole state space and can provide improved guesses concerning the "unknown" transition dynamics, even for previously encountered tasks.

Intuitively, the performance of an agent on one task has the potential to be further improved (positive backward transfer) if there exists a sufficiently large set of state–action transition pairs of which the task-specific model's predictions are not confident due to lack of data. In our algorithm, the aleatory uncertainty (irreducible chance in the outcome) is measured by the output variance of the prediction $\{\sigma_{r_\tau^p}, \sigma_{s_\tau^p}\}$, and the epistemic uncertainty (due to lack of experience) corresponds to the uncertainty of the output mean and variance (see Definition 1 below). Thus, a straightforward method to improve a previously learned task-specific model is to find the predictions it needs to make that have high epistemic uncertainty, and replace them with the predictions from the world-model posterior, which has lower epistemic uncertainty. If we only consider reward prediction, the quantity for measuring whether a task-specific/world model is sufficiently confident is as follows.

Definition 4.5. For a given state–action pair (s, a), we define the confidence level c of the predictions (reward) $r_\tau^p(s, a)$ from a task-specific/world model as:

$$c = -\frac{\sum_{p=1}^{P}(\mu_{r_\tau^p} - \overline{\mu}_{r_\tau^p})^2}{P-1} - \alpha * \frac{\sum_{p=1}^{P}(\sigma_{r_\tau^p} - \overline{\sigma}_{r_\tau^p})^2}{P-1}, \tag{10}$$

where P is the number of particles and α is a hyperparameter controlling the scale of the second term. A similar definition applies to the task-specific/world model's next-state prediction. Intuitively, c measures the uncertainty of the output mean and variance for each dynamic prediction. During CEM planning, we propagate each pair of P state particles with different network parameters thanks to the usage of BNN, resulting in P different $\{\mu_{r_\tau^p}, \sigma_{r_\tau^p}\}$. Thus we can further calculate the confidence level of current predictions based on these P pairs of output mean and variance.

We show the detailed backward transfer algorithm in Algorithm 2. Compared with the forward training version, the agent will also calculate the confidence level of the task-specific and general-knowledge model for each particular transition and compare the value of them when predicting the state particles. If the task-specific model is not confident enough for this state–action pair (i.e. $c_{m_i} < c_{wm}$), we will use the world-model to do the predictions instead.

5 Experiments

5.1 OpenAI Gym MuJoCo Domains

We evaluated the performance of VBLRL on HiP-MDP versions of several continuous control tasks from the Mujoco physics simulator [45], *HalfCheetah-gravity, HalfCheetah-bodyparts, Hopper-gravity, Hopper-bodyparts, Walker-gravity, Walker-bodyparts*, all of which are lifelong-RL benchmarks used in prior work [31]. For each of six different domains, the task-specific hidden parameters correspond to different gravity values or different sizes and masses of the simulated body parts. Details can be found in the appendix. Compared with prior work, we substantially reduced the number of iterations that the agent can sample and train on: 100 iterations for each task and a horizon of 100 (Halfcheetah) or 400 (Hopper & Walker) for each iteration. We used such settings to increase the difficulty of lifelong learning and test the sample efficiency of lifelong RL algorithms, given that it is hard for a single-task training algorithm to obtain a good policy within such limited number of interactions with the environments. However, we show in appendix I and J that, as the agent is

exposed to more and more tasks, our lifelong learning agent is able to achieve similar performance to that of more fully trained single-task agents while requiring far fewer per-task samples.

	VBLRL	T-HiP-MDP	LPG-FTW ($2\times$ samples)	EWC ($2\times$ samples)	Single-task MBRL
CG-Start	$\mathbf{160.68 \pm 48.80}$	126.95 ± 31.41	-81.59 ± 9.18	-3426.76 ± 827.99	-83.96 ± 60.10
CG-Train	$\mathbf{226.72 \pm 26.53}$	170.20 ± 39.92	-29.49 ± 11.03	-3440.66 ± 1007.50	-40.47 ± 10.68
CG-Back	$\mathbf{231.79 \pm 23.49}$	97.84 ± 22.04	-29.95 ± 11.64	-6672.33 ± 3748.63	/
CB-Start	$\mathbf{110.74 \pm 41.96}$	78.95 ± 18.43	-263.94 ± 40.80	-5016.93 ± 1708.10	-101.02 ± 39.11
CB-Train	$\mathbf{173.97 \pm 78.26}$	87.20 ± 9.42	-217.86 ± 42.82	-5454.52 ± 2145.82	-58.93 ± 33.24
CB-Back	$\mathbf{181.60 \pm 67.50}$	116.03 ± 17.35	-116.41 ± 65.64	-13889.31 ± 6851.05	/
HG-Start	268.11 ± 33.29	230.21 ± 30.75	305.63 ± 34.55	$\mathbf{306.79 \pm 29.14}$	18.62 ± 1.51
HG-Train	332.89 ± 23.68	285.87 ± 41.48	$\mathbf{352.10 \pm 25.25}$	345.47 ± 40.26	20.46 ± 1.77
HG-Back	$\mathbf{360.94 \pm 7.71}$	312.11 ± 55.45	347.07 ± 44.09	301.06 ± 114.11	/
HB-Start	193.27 ± 8.03	181.94 ± 12.97	$\mathbf{256.13 \pm 69.68}$	133.95 ± 27.61	19.12 ± 1.75
HB-Train	$\mathbf{296.34 \pm 11.15}$	227.50 ± 41.78	285.62 ± 78.18	139.01 ± 46.24	21.25 ± 2.67
HB-Back	$\mathbf{316.74 \pm 20.72}$	213.93 ± 75.95	281.99 ± 74.96	-384.27 ± 199.27	/
WG-Start	$\mathbf{248.97 \pm 19.95}$	217.94 ± 39.99	140.75 ± 67.64	163.61 ± 86.55	4.32 ± 0.23
WG-Train	$\mathbf{333.65 \pm 26.73}$	268.97 ± 36.26	166.00 ± 45.04	181.76 ± 97.42	210.41 ± 39.34
WG-Back	$\mathbf{364.18 \pm 49.60}$	173.33 ± 58.43	168.39 ± 94.19	216.09 ± 62.67	/
WB-Start	$\mathbf{222.88 \pm 34.49}$	220.07 ± 20.31	164.79 ± 10.31	164.00 ± 34.31	4.64 ± 0.23
WB-Train	$\mathbf{311.41 \pm 16.30}$	266.97 ± 26.72	165.75 ± 2.39	206.97 ± 41.40	196.42 ± 22.04
WB-Back	$\mathbf{317.82 \pm 13.90}$	229.10 ± 16.60	195.01 ± 49.25	152.94 ± 70.59	/

Table 1: Results on OpenAI Gym Mujoco domains. *CG* denotes **Cheetah-Gravity**, *CB* denotes **Cheetah-Bodyparts**, *HG* denotes **Hopper-Gravity**, *HB* denotes **Hopper-Bodyparts**, *WG* denotes **Walker-Gravity**, *WB* denotes **Walker-Bodyparts**.

We compared VBLRL against: 1. state-of-the-art lifelong RL method LPG-FTW [31], 2. EWC [28], which is a single-model lifelong RL algorithm that achieves comparable performance with LPG-FTW as shown in the latter paper, 3. single-task Model-Based RL (using the same BNN structure and planning procedures), which stands for training the agent from scratch for each new task and do not use the world model to initialize the task-specific model, 4. T-HiP-MDP [27], which is a model-based lifelong RL baseline. For a fair comparison, we further replaced the DDQN algorithm [46] used in T-HiP-MDP with CEM planning, and let the transition model also predict the reward for each state–action pair. This modified baseline is similar to the single-model version of VBLRL (i.e., only using the world-model posterior across all the tasks). Moreover, LPG-FTW and EWC are built upon a relatively simple policy gradient baseline, which does not perform as well as our model-based baseline in the single-task setting within the same amount of interactions. Thus, we let them collect **twice the amount of samples** each iteration compared to VBLRL. We show in Appendix I that, with such modifications, in single-task settings the baselines achieve better or at least similar performance compared to the single-task version of VBLRL.

The results are shown in Table 1 and Figure 7 in Appendix J. For all six domains, we report the average performance of all the tasks at the beginning of training (**Start**) and after all training for each new task (**Train**), as well as the average performance for all previous tasks after training for a given number of tasks, which is our backward transfer test (**Back**). As shown in the results, our VBLRL shows better performance on all three test stages of the HalfCheetah domain and Walker domain, as well as better backward transfer performance on Hopper-gravity and Hopper-bodyparts than the other three algorithms. Comparing Figure 6 and Figure 7, we find that, in the Hopper domains, even though the single-task baseline used by LPG-FTW and EWC performed much better than VBLRL after we let them collect $\mathbf{2}\times$ samples each iteration, VBLRL still achieves comparable performance with LPG-FTW and EWC in lifelong RL experiments, suggesting that VBLRL is capable of making better use of the data in the lifelong learning setting. In the walker domains where the single-task baselines achieved similar performance, VBLRL shows significantly better performance than LPG-FTW/EWC in lifelong RL experiments. These comparisons also suggest that VBLRL's lifelong learning performance may be further improved when combined with better model-based deep RL algorithms, like Dreamer [20]. EWC fails in some of the tasks as it is hard to directly learn a single shared policy that achieves good performance when the tasks are highly diverse. T-HiP-MDP shows good results on some of the tasks because it is more sample-efficient in learning a shared model across all the tasks and more easily captures the world-model uncertainty. However, it cannot achieve as good performance as VBLRL as it is hard to model the task-specific uncertainty using only one model across all tasks, which also leads to negative backward transfer performance on tasks like Cheetah-Gravity. Comparing VBLRL's performance on the **Train** stage and **Back** stage, we also find that it shows positive backward transfer results on most tasks, without showing patterns of catastrophic forgetting. Overall, VBLRL's world-model posterior contributes to better forward transfer performance (**Start**), the learning of task-specific posterior contributes to

better forward transfer training for each new task (**Train**), and the combination of these two posteriors guides the agent to achieve better backward transfer performance (**Back**). Comparing our method to LPG-FTW and EWC highlights the advantage of model-based techniques for sample efficiency in lifelong RL setting, while comparing to T-HiP-MDP and single-task MBRL demonstrates the importance of estimating both global and local uncertainty.

5.2 Meta-world Domains

Meta-World [52] contains a suite of challenging evaluation benchmarks on a robotic SAWYER arm. We evaluated the performance of VBLRL as well as the baselines on two Meta-World task sets: Reach and Reach-Wall following the settings given by [54]. We used the v1 version of Meta-World, with state observations and dense versions of the reward functions. The hidden parameters in these cases are the goals we want the robot to reach, which controls the reward functions and are not included in the agent's observation space. We measured performance in terms of average cumulative rewards across tasks. As shown in Figure 2, VBLRL achieves significantly higher performance than the other baselines. VBLRL-Backward denotes VBLRL's average backward transfer performance on all the tasks and is higher than the final performance of VBLRL's forward training in both tasks, suggesting that our approach can achieve positive backward transfer. We also include experimental results on **MT10** and **MT50** domains of Meta-World[2] in Appendix H, which are not typical HiP-MDP settings but have more challenging variations between different tasks.

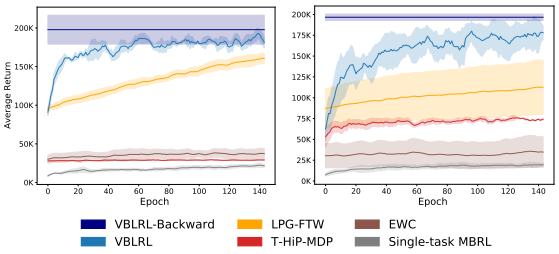

Figure 2: Average performance during training across all tasks. Left: Reach; Right: Reach-Wall.

5.3 Ablation Study

This section evaluates the essentiality of VBLRL's components. As shown in Figure 3 left, we tested different backward transfer strategy for VBLRL. Task-specific-Backward denotes the performance if directly using the learned task-specific model to do all the predictions without using world model. World-model-Backward denotes the performance if using world-model to do all the predictions without using each task-specific model. Compared to these baselines, VBLRL's backward transfer strategy achieves the best performance by combining these two kinds of models according to their confidence level. We also explored the influence of using a Bayesian neural network to model the task-specific model. As shown in the right side of the figure, using BNN enables faster adaptation to new tasks compared with using regular neural networks (VBLRL-Deterministic). We also include ablation studies on the number of particles in CEM planning in Appendix O.

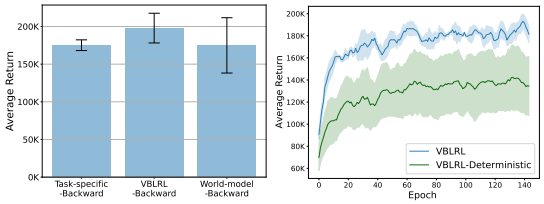

Figure 3: Ablation Study on Reach domain. Left: Backward transfer strategy; Right: Using Bayesian neural networks to model the task-specific posterior.

6 Conclusion and Discussion

To improve sample efficiency in lifelong RL, our work proposed a model-based lifelong RL approach that distills shared knowledge from similar MDPs and maintains a Bayesian posterior to approximate the distribution derived from that knowledge. We gave a sample-complexity analysis of the algorithm in the finite MDP setting. Then, we extended our method to use variational inference, which scales

[2]MT10 and MT50 consist of 10 and 50 different categories of manipulation tasks, including reach, push, drawer close, button press etc., and is a highly difficult setting for RL generalization.

better and supports both backward and forward transfer. Our experimental results show that the proposed algorithms enable faster training on new tasks through collecting and transferring the knowledge learned from preceding tasks.

One of the core questions in lifelong learning setting is how tasks should be related for transfer to be effective, and in this paper we are making the HiP-MDP assumption, that is, a family of tasks is generated by varying a latent task parameter vector drawn for each task according to the world-model distribution. The relationship between different tasks can be modeled in different ways in lifelong RL and this problem setting will affect the algorithm's performance. However, we do believe our method can help the agent transfer knowledge for tasks that are in other ways related, as we show in Appendix H.

The overall lifelong learning performance of our proposed algorithm is largely limited by the performance of the single-task model-based baseline. We show the advantages of model-based lifelong learning methods over model-free methods when the single-task baselines' performance are close. In practice, it's possible that the model-free lifelong RL methods still outperforms model-based methods simply because the model-free single-task baseline is much stronger.

Bayesian Neural Networks are more computationally expensive than regular neural networks. The longest running time among all our experiments is hopper which took around 96 hours to finish training and evaluation on all the tasks. This is in general acceptable time cost and can be potentially improved with more recent BNN techniques.

Acknowledgement

The authors thank the members of Brown bigAI for discussions and helpful feedback, and the anonymous reviewers for valuable feedback that improved the paper substantially. This research was supported by the NSF under grant #1955361 and CAREER award #1844960, and the DARPA Lifelong Learning Machines program under grant #FA8750-18-2-0117. The U.S. Government is authorised to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NSF or DARPA. This research was conducted using computational resources and services at the Center for Computation and Visualization, Brown University.

References

[1] Asmuth, J. and Littman, M. Learning is planning: Near Bayes-optimal reinforcement learning via Monte-Carlo tree search. In *UAI*, 2011.

[2] Asmuth, J., Li, L., Littman, M. L., Nouri, A., and Wingate, D. A Bayesian sampling approach to exploration in reinforcement learning. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, 2009*, pp. 19–26. AUAI Press, 2009.

[3] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. *CoRR*, abs/1505.05424, 2015.

[4] Botev, Z., Kroese, D. P., Rubinstein, R., and L'Ecuyer, P. Chapter 3 – the cross-entropy method for optimization. *Handbook of Statistics*, 31:35–59, 2013.

[5] Bou-Ammar, H., Eaton, E., Ruvolo, P., and Taylor, M. E. Online multi-task learning for policy gradient methods. In *ICML*, 2014.

[6] Brafman, R. I. and Tennenholtz, M. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. 3(null), 2003. ISSN 1532-4435.

[7] Brunskill, E. and Li, L. Pac-inspired option discovery in lifelong reinforcement learning. In *ICML*, 2014.

[8] Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pp. 4759–4770, 2018.

[9] des Combes, R. T., Bachman, P., and van Seijen, H. Learning invariances for policy generalization. In *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net, 2018.

[10] Doshi-Velez, F. and Konidaris, G. D. Hidden parameter markov decision processes: A semi-parametric regression approach for discovering latent task parametrizations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*, pp. 1432–1440. IJCAI/AAAI Press, 2016.

[11] Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *ICML*, 2016.

[12] Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[13] Finn, C., Xu, K., and Levine, S. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pp. 9537–9548, 2018.

[14] Fu, H., Tang, H., Hao, J., Chen, C., Feng, X., Li, D., and Liu, W. Towards effective context for meta-reinforcement learning: an approach based on contrastive learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pp. 7457–7465. AAAI Press, 2021.

[15] Ghavamzadeh, M., Mannor, S., Pineau, J., and Tamar, A. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015. ISSN 1935-8245.

[16] Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. L. Recasting gradient-based meta-learning as hierarchical bayes. In *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net, 2018.

[17] Graves, A. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011*, pp. 2348–2356, 2011.

[18] Guez, A., Silver, D., and Dayan, P. Efficient bayes-adaptive reinforcement learning using sample-based search. In *NIPS*, 2012.

[19] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, pp. 1856–1865. PMLR, 2018.

[20] Hafner, D., Lillicrap, T. P., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *ArXiv*, abs/1912.01603, 2020.

[21] Hinton, G. E. and van Camp, D. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, COLT 1993*, pp. 5–13. ACM, 1993.

[22] Hinton, G. E. and van Camp, D. Keeping the neural networks simple by minimizing the description length of the weights. In *COLT '93*, 1993.

[23] Houthooft, R., Chen, X., Duan, Y., Schulman, J., Turck, F. D., and Abbeel, P. VIME: variational information maximizing exploration. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, pp. 1109–1117, 2016.

[24] Isele, D., Rostami, M., and Eaton, E. Using task features for zero-shot knowledge transfer in lifelong learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*, pp. 1620–1626. IJCAI/AAAI Press, 2016.

[25] Isele, D., Rostami, M., and Eaton, E. Using task features for zero-shot knowledge transfer in lifelong learning. In *IJCAI*, 2016.

[26] Kaelbling, L. P., Littman, M. L., and Moore, A. W. Reinforcement learning: A survey. *J. Artif. Intell. Res.*, 4:237–285, 1996.

[27] Killian, T. W., Daulton, S., Doshi-Velez, F., and Konidaris, G. D. Robust and efficient transfer learning with hidden parameter markov decision processes. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pp. 6250–6261, 2017.

[28] Kirkpatrick, J., Pascanu, R., Rabinowitz, N. C., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016.

[29] Li, Z. and Hoiem, D. Learning without forgetting, 2017.

[30] Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pp. 6467–6476, 2017.

[31] Mendez, J. A. M., Wang, B., and Eaton, E. Lifelong policy gradient learning of factored policies for faster training without forgetting. *ArXiv*, abs/2007.07011, 2020.

[32] Nagabandi, A., Finn, C., and Levine, S. Deep online learning via meta-learning: Continual adaptation for model-based rl. *ArXiv*, abs/1812.07671, 2019.

[33] Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning, 2018.

[34] Osband, I., Blundell, C., Pritzel, A., and Roy, B. V. Deep exploration via bootstrapped dqn. In *NIPS*, 2016.

[35] Parisotto, E., Ba, L. J., and Salakhutdinov, R. Actor-mimic: Deep multitask and transfer reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[36] Puterman, M. L. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[37] Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5331–5340. PMLR, 2019.

[38] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Driessche, G. V. D., Graepel, T., and Hassabis, D. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[39] Strehl, A., Li, L., and Littman, M. Incremental model-based learners with formal learning-time guarantees. *ArXiv*, abs/1206.6870, 2006.

[40] Strehl, A. L., Li, L., and Littman, M. L. Reinforcement learning in finite mdps: Pac analysis. *J. Mach. Learn. Res.*, 10:2413–2444, 2009.

[41] Strens, M. J. A. A Bayesian framework for reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 943–950, 2000.

[42] Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[43] Teh, Y. W., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pp. 4496–4506, 2017.

[44] Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294, 1933.

[45] Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012*, pp. 5026–5033. IEEE, 2012.

[46] van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence,2016*, pp. 2094–2100. AAAI Press, 2016.

[47] Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. Learning to reinforcement learn. *CoRR*, abs/1611.05763, 2016.

[48] Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E., Zhang, S., Zhang, G., Abbeel, P., and Ba, J. Benchmarking model-based reinforcement learning. *CoRR*, abs/1907.02057, 2019.

[49] Wilson, A., Fern, A., Ray, S., and Tadepalli, P. Multi-task reinforcement learning: A hierarchical Bayesian approach. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), 2007*, volume 227 of *ACM International Conference Proceeding Series*, pp. 1015–1022. ACM, 2007.

[50] Wilson, A., Fern, A., Ray, S., and Tadepalli, P. Multi-task reinforcement learning: a hierarchical bayesian approach. In *ICML '07*, 2007.

[51] Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., and Ahn, S. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pp. 7343–7353, 2018.

[52] Yu, T., Quillen, D., He, Z., Julian, R. C., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *ArXiv*, abs/1910.10897, 2019.

[53] Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence, 2017.

[54] Zhang, J., Wang, J., Hu, H., Chen, T., Chen, Y., Fan, C., and Zhang, C. Metacure: Meta reinforcement learning with empowerment-driven exploration. In *ICML*, 2021.

[55] Zhang, T. From -entropy to kl-entropy: Analysis of minimum information complexity density estimation. *Annals of Statistics*, 34:2180–2210, 2006.

[56] Zintgraf, L. M., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., and Whiteson, S. Varibad: A very good method for bayes-adaptive deep RL via meta-learning. In *8th International Conference on Learning Representations, ICLR 2020*. OpenReview.net, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [N/A]

   (b) Did you mention the license of the assets? [N/A]

   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A VBLRL algorithm

Algorithm 1 Variational Bayesian Lifelong RL

Input: Initialize general knowledge(world) model $p_{wm}(\cdot|s,a;\omega_{wm})$, planning horizon T
for each task m_i from $i = 1, 2, 3, \cdots, M$ **do**
  Initialize task-specific model $p_{m_i}(\cdot|s,a;\omega_i)$ with parameters of general knowledge model p_{wm}
  for each episode **do**
    for Time $t = 0$ to TaskHorizon **do**
      Sample Actions $a_{t:t+T} \sim$ CEM($\cdot$)
      Propagate state particles s_τ^p with $p_{m_i}(s'|s,a;\omega_i)$
      Evaluate actions as $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} p_{m_i}(r|s,a;\omega_i)$
      Update CEM($\cdot$) distribution.
      Execute optimal actions $a^*_{t:t+T}$
    end for
    Add transitions to replay buffer D_{m_i}
    Update task-specific model according to Equation (9) given replay buffer D_{m_i}
    Update general knowledge model according to Equation (9) given replay buffers $\{D_{m_1}, \cdots, D_{m_i}\}$
  end for
end for

Note that in $p_{m_i}(\cdot|s,a;\omega_i)$ and $p_{m_{wm}}(\cdot|s,a;\omega_{wm})$, p stands for the task-specific/world probabilistic model we are using. In s_τ^p and $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} r_\tau^p$, p denotes one of the state particles $p \in \{1, \cdots, P\}$.

At the beginning of training (before encountering any tasks), the agent first randomly initialize the weights and bias of the world-model BNN $p_{wm}(\cdot|s,a;\omega_{wm})$. Then each time when the agent encounters a new task, the task-specific model $p_{m_i}(\cdot|s,a;\omega_i)$ for that task will be initialized by copying network parameters from the world-model BNN. Then for planning, at each step we begin by creating P particles from the current state $s^p_{\tau=t} = s_t \forall p$. Then, we sample N candidate action sequences $a_{t:t+T}$ from a learnable distribution. These two steps are the same as PETS [8]. Then we propagate the state–action pairs using the learned task-specific model $p_{m_i}(\cdot|s,a)$ (BNN) and use the cross entropy method [4] to update the sampling distribution to make the sampled action sequences close to previous action sequences that achieved high reward. We further calculate the cumulative reward estimated (via the learned model) for previously sampled sequences and select the current action based on the mean of that distribution. Then we can add the new transitions to the replay buffer. We update the task specific model according to Equation (9) by sampling from the replay buffer of the current task, and update the world model with samples from all previous tasks' replay buffers.

Algorithm 2 Variational Bayesian Lifelong RL (Backward transfer)

Input: Test task m_i, planning horizon T, task-specific model $p_{m_i}(s',r|s,a;\omega_i)$, general-knowledge model $p_{wm}(s',r|s,a;\omega_{wm})$
for Time $t = 0$ to TaskHorizon **do**
  for Trial $k = 1$ to K **do**
    Sample Actions $a_{t:t+T} \sim$ CEM($\cdot$)
    for each action **do**
      Propagate state particles s_τ^p with $p_{m_i}(s',r|s,a)$
      Propagate state particles s_τ^p with $p_{wm}(s',r|s,a)$
      Compute confidence level c_{m_i} for task-specific model and c_{wm} for general-knowledge model (**Definition 4.5**)
      Choose the propagation results from the model with higher confidence level c
    end for
    Evaluate actions as $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} r_\tau^p$
    Update CEM($\cdot$) distribution.
  end for
  Execute optimal actions $a^*_{t:t+T}$
end for

For backward transfer, given a previously encountered task m_i, at each planning step we predict the next state and reward with both task-specific model and world model. Then we compare the confidence level of these two predictions and choose to use the prediction results that have higher confidence level. The other planning procedures are the same as in forward training.

B BNN model

The form of the BNN we used is the same as in VIME [23]. We model the transition models as Gaussian distributions:

$$T(\cdot|s,a) = \mathcal{N}(f_\omega^\mu(s,a), f_\omega^\sigma(s,a)) \tag{11}$$

The function f_θ is represented as a Bayesian neural network parameterized by θ, which is further modeled as the posterior distribution parameterized by ϕ, predicts the mean μ_s, μ_r and variance σ_s, σ_r given current state and action s, a. We can view the BNN model in VBLRL as an infinite neural network ensemble by integrating out its parameters:

$$T(s', r|s,a) = \int_\Omega T(s', r|s, a; \omega) q(\omega; \phi) d\omega \tag{12}$$

Compared to previous model-based algorithms that use finite number of neural network ensembles (e.g. PETS), our choice of BNN is more suitable for lifelong RL as we only need to maintain one neural network for each task, and we can sample an unlimited number of predictions from it which better estimates the uncertainty and is essential in our setting where both dynamic function and reward function are not given unlike prior model-based RL methods.

C BLRL algorithm

Note that the single-task baseline that BLRL is built upon is BOSS, and **could be replaced by other Bayesian-exploration RL algorithm**. We use a hierarchical Bayesian model to represent the distribution over MDPs. Figure 4 shows our generative model in $plate\ notation$. Ψ is the parameter set that represents distribution P_Ω. It functions as the world-model posterior that aims to capture the common structure across different tasks. The resulting MDP m_i is created based on ω_i, which is one hidden parameter sampled from Ψ. We can sample from our approximation of Ψ to create and solve possible MDPs.

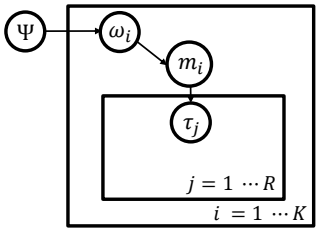

Figure 4: Plate representation for the BLRL approach. τ_j denotes trajectory $\{s, a, r, s'\}_j$. There are K different tasks and the agent samples R trajectories from each task.

The full algorithm is shown in Algorithm 3. Each time the agent encounters a new task m_i, it first initializes the task-specific posterior $p_{m_i}(\cdot|s_t, a_t)$ with the parameter values from the current world-model posterior p_{wm}, and then, for each timestep, selects actions following sampling-based Bayesian exploration procedures from this posterior [44; 2]. A set of sampled MDPs drawn from p_{m_i} is a concrete representation of the uncertainty within the current task.

Concretely, BLRL samples K models from the task-specific posterior whenever the number of transitions from a state–action pair has reached threshold B. Analogously to RMAX [6], we call a state–action pair **known** whenever it has been observed $N_{s_t,a_t} = B$ times. For each state–action pair, if it is **known**, we use the task-specific posterior to sample the model. If it is **unknown**, we instead sample from the world-model posterior. These models are combined into a merged MDP $m_i^\#$ and BLRL solves $m_i^\#$ with value iteration to get a policy $\pi^*_{m_i^\#}$. Intuitively, this approach creates optimism

in the face of uncertainty as the agent can choose actions based on the highest performing transition of the K models sampled, which drives exploration. The new policy $\pi^*_{m_i^\#}$ will be used to interact with the environment until a new state–action pair reaches the sampling threshold. The collected transitions from the current task will be used to update the task-specific posterior immediately, while the world-model posterior will be updated using transitions from **all** the previous tasks at a slower pace. For simple finite MDP problems in practice, we use the Dirichlet distribution (the conjugate for the multinomial) to represent the Bayesian posterior. Thus, the updating process for the posterior is straightforward to compute. Intuitively, BLRL rapidly adapts to new tasks as long as the prior of the task-specific model (that is, the world-model posterior) is close to the true underlying model and captures the uncertainty of the common structure of a set of tasks. Empirical evaluations of BLRL on gridworlds are given in the appendix.

Algorithm 3 Lifelong Bayesian Sampling Approach Algorithm

Input: K, B
initialize MDP set, the world-model posterior $p_{wm}(s_{t+1}, r_t|s_t, a_t)$
for each MDP m_i **do**
  $N_{s,a} \leftarrow 0, \forall s, a$
  $do_sample \leftarrow$ TRUE
  initialize the task-specific posterior $p_{m_i}(s_{t+1}, r_t|s_t, a_t) \leftarrow p_{wm}(s_{t+1}, r_t|s_t, a_t)$
  for all timesteps $t = 1, 2, 3, ...$ **do**
    if do_sample **then**
      Sample K models m_{i_1}, m_{i_2},$\cdots$,m_{i_K} from the task-specific posterior $p_{m_i}(s_{t+1}, r_t|s_t, a_t)$.
      Merge the models into the mixed MDP $m_i^\#$
      Solve $m_i^\#$ to obtain $\pi^*_{m_i^\#}$
      $do_sample \leftarrow$ FALSE
    end if
    Use $\pi^*_{m_i^\#}$ for action selection: $a_t \leftarrow \pi_{m_i^\#}(s_t)$ and observe reward r_t and next state s_{t+1}
    $N_{s_t,a_t} \leftarrow N_{s_t,a_t} + 1$
    Update the task-specific posterior distribution $p_{m_i}(s_{t+1}, r_t|s_t, a_t)$ for the current MDP
    if $N_{s_t,a_t} = B$ **then**
      Update the world-model posterior distribution $p_{wm}(s_{t+1}, r_t|s_t, a_t)$ with the collected transitions
      $do_sample \leftarrow$ TRUE
    end if
  end for
end for

D Experimental Setting

D.1 OpenAI Gym Mujoco Domains

Similar to [31], we evaluated on the HalfCheetah, Hopper, and Walker-2D environments. For the gravity domain, we select a random gravity value between $0.5g$ and $1.5g$ for each task. For the body-parts domain, we set the size and mass of each of the four parts of the body (head, torso, thigh, and leg) to a random value between $0.5\times$ and $1.5\times$ its nominal value. As shown in Appendix C of [31], these changes lead to highly diverse tasks for lifelong RL. Further, as required by CEM-based deep RL methods [48], we added a check-done function for Hopper and Walker following the settings in previous paper.

When implementing VBLRL, we found that in the first few episodes of each new tasks, the agent hasn't collected enough samples of the new task, which results in overfitting problems when training the task-specific. Thus, we use the world-model posterior instead to do the first few rounds of predictions and let the task-specific model begin training after collecting enough samples. The world-model has lower possibility of overfitting as its training data comes from all the previous tasks and has much larger quantity. The results shown in the experiments section are collected after the task-specific model starts collecting samples. We list the other implementation details below. The planning horizons are selected from values suggested by previous model-based RL papers [8; 48]. We

find that in Hopper and Walker, using regular neural networks instead of Bayesian neural networks to model the **task-specific posterior** also works fine (the world model still uses BNN).

Hyper-parameters	**CG**	**CB**	**HG**	**HB**	**WG**	**WB**	**Reach**	**Reach-Wall**
# iterations	100	100	100	100	100	100	150	150
# Steps (each iteration)	100	100	400	400	400	400	150	150
learning rate (world model)	0.001	0.001	0.0006	0.0006	0.0006	0.0006	0.001	0.001
learning rate (task-specific model)	0.0005	0.0005	0.0006	0.0006	0.0006	0.0006	0.001	0.001
planning horizon	20	20	30	30	30	30	1	1
kl-divergence weight	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
# particles (CEM)	50	50	1	1	1	1	1	1
batch size (world-model)	8×64	8×64	8×64	8×64	8×64	8×64	8×64	8×64
batch size (task-specific)	256	256	256	256	256	256	256	256
# tasks	40	40	20	20	20	20	30	30
search population size	500	500	500	500	500	500	500	500
# elites (CEM)	50	50	50	50	50	50	50	50

Table 2: Hyperparameters for different task sets

For LPG-FTW and EWC, we use the original source code[3] with parameters and model architectures suggested in the original paper. Specifically, we select step size from $\{0.005, 0.05, 0.5\}$. For LPG-FTW, e use $\lambda = 1e-5, \mu = 1e-5$ and select k from 3,5,10. For EWC, we select λ from $\{1e-6, 1e-7, 1e-4\}$. For HiP-MDP baseline, we modify the original algorithm for a fair comparison. We replace the DDQN algorithm used in T-HiP-MDP with the exact same CEM planning method we used in VBLRL as well as the same parameters. And we use the same model architecture of Bayesian Neural network by modifying the baseline algorithm to also predict reward for each state-action pair (the original method only considers next-state prediction).

For BOSS and BLRL, we set the number of sampled models $K = 5$, and $\gamma = 0.95, \Delta = 0.01$ for value iteration.

We reported the results averaged over three random seeds, and the error bar shows one standard deviation. All experiments were run on our university's high performance computing cluster.

One of the limitations of the current experiments is that we did not evaluate our algorithm on image-based environments. We leave this for future work.

D.2 Meta-World Domains

The hyperparameters used are included in Table 4.

D.3 Grid-World Item Searching

Our testbed consists of a collection of houses, each of which has four rooms. The goal of each task is to find a specific object (blue, green or purple) in the current house. The type of each room is sampled based on an underlying distribution given by the environment. Each room type has a corresponding probability distribution of which kind of objects can be found in rooms of this type. Different tasks/houses vary in terms of which rooms are which types and precisely where objects are located in the room (the task's hidden parameters). Room types are sampled from a joint distribution.

Room type probability	**Room 1**	**Room 2**	**Room 3**	**Room 4**
Top-left	0.4	0	0.4	0.2
Bottom-left	0	0.8	0	0.2
Top-right	0.1	0	0	0.9
Bottom-right	0	0	0.8	0.2

Table 3: Room type probability distribution

[3] https://github.com/Lifelong-ML/LPG-FTW

Object type probability	**Blue ball**	**Green box**	**Purple box**
Room 1	0	0.3	0
Room 2	0	0.2	1
Room 3	0.6	0	0
Room 4	0	0	0

Table 4: Object type probability distribution

D.4 Box-jumping Task

We use a simplified version of jumping task [9] as a simple testbed for the proposed algorithm VBLRL. We select a random position of obstacle between $15 \sim 33$ for each task. The 4-element state vector describes the (x, y) coordinates of the agent's current position, and its velocity in the x and y directions. The agent can choose from two actions: jump and right. The reward function for this box-jumping task is:

$$R_t = \mathbb{I}\{s_t \text{ reach the right wall}\} - \mathbb{I}\{s_{t+1} \text{ hit the obstacle}\} + \dot{x}_t \cdot \mathbb{I}\{s_{t+1} \text{ not hit the obstacle}\} \quad (13)$$

E Proof for Lemma 4.1

In the following proofs as well as in the main text, n and T **both** denote the number of samples collected from the environment.

We first rewrite the Bayesian posterior density $g(\omega|D_i^T)$ with respect to π as:

$$\begin{aligned} g(\omega|D_i^T) &= \frac{p(D_i^T|\omega)}{\int_\Gamma p(D_i^T|\omega)d\pi(\omega)} \\ &= \frac{\prod_{t=1}^T p(s^{t+1}, r^t|D_i^t, a^t; \omega)}{\int_\Gamma \prod_{t=1}^T p(s^{t+1}, r^t|D_i^t, a^t; \omega)d\pi(\omega)} \\ &= \frac{\prod_{t=1}^T p(s^{t+1}, r^t|D_i^t, a^t; \omega)}{\mathbb{E}_\pi \prod_{t=1}^T p(s^{t+1}, r^t|D_i^t, a^t; \omega)} \\ &= \frac{\prod_{t=1}^T p(s^{t+1}, r^t|D_i^t, a^t; \omega)}{\prod_{t=1}^T q(s^{t+1}, r^t|D_i^t, a^t)} \end{aligned} \quad (14)$$

Then, we refer to the following lemma [55] which is a known information-theoretical inequality:

Lemma E.1. *Assume that $f(\omega)$ is a measurable real-valued function on Γ, and $g(\omega)$ is a density with respect to π; we have*

$$\mathbb{E}_\pi g(\omega) f(\omega) \leq D_{KL}(gd\pi||d\pi) + \ln \mathbb{E}_\pi \exp(f(\omega)) \quad (15)$$

We refer the readers to the original paper for detailed proof.

Based on the definition of $R_n(g)$, we have:

$$\begin{aligned} R_n(g) &= \mathbb{E}_\pi g(\omega_i) \sum_{t=1}^T \ln \frac{q(s^{t+1}, r^t|D_i^t, a^t)}{p(s^{t+1}, r^t|D_i^t, a^t; \omega_i)} + D_{KL}(gd\pi||d\pi) \\ &= \mathbb{E}_\pi \frac{\prod_{t=1}^T p(s^{t+1}, r^t|D_i^t, a^t; \omega_i)}{\prod_{t=1}^T q(s^{t+1}, r^t|D_i^t, a^t)} \sum_{t=1}^T \ln \frac{q(s^{t+1}, r^t|D_i^t, a^t)}{p(s^{t+1}, r^t|D_i^t, a^t; \omega_i)} \\ &+ \mathbb{E}_\pi \frac{\prod_{t=1}^T p(s^{t+1}, r^t|D_i^t, a^t; \omega_i)}{\prod_{t=1}^T q(s^{t+1}, r^t|D_i^t, a^t)} \sum_{t=1}^T \ln \frac{p(s^{t+1}, r^t|D_i^t, a^t; \omega_i)}{q(s^{t+1}, r^t|D_i^t, a^t)} \\ &= \mathbb{E}_\pi \frac{\prod_{t=1}^T p(s^{t+1}, r^t|D_i^t, a^t; \omega_i)}{\prod_{t=1}^T q(s^{t+1}, r^t|D_i^t, a^t)} \Big[\ln \frac{\prod_{t=1}^T q(s^{t+1}, r^t|D_i^t, a^t)}{\prod_{t=1}^T p(s^{t+1}, r^t|D_i^t, a^t; \omega_i)} - \ln \frac{\prod_{t=1}^T q(s^{t+1}, r^t|D_i^t, a^t)}{\prod_{t=1}^T p(s^{t+1}, r^t|D_i^t, a^t; \omega_i)} \Big] \\ &= 0 \end{aligned} \quad (16)$$

Then, let $f(\omega) = -\sum_{t=1}^{T} \ln \frac{q(s^{t+1}, r^t | D_i^t, a^t)}{p(s^{t+1}, r^t | D_i^t, a^t; \omega)}$ in Lemma E.1, we have

$$\begin{aligned} R_n(\cdot) &= D_{KL}(gd\pi || d\pi) - \mathbb{E}_\pi g(\omega) f(\omega) \\ &\geq \ln \mathbb{E}_\pi \exp(f(\omega)) \\ &= \ln \mathbb{E}_\pi \frac{\prod_{t=1}^{T} p(s^{t+1}, r^t | D_i^t, a^t; \omega_i)}{\prod_{t=1}^{T} q(s^{t+1}, r^t | D_i^t, a^t)} \\ &= \ln \frac{\prod_{t=1}^{T} q(s^{t+1}, r^t | D_i^t, a^t)}{\prod_{t=1}^{T} q(s^{t+1}, r^t | D_i^t, a^t)} \\ &= 0 \end{aligned} \tag{17}$$

Combine Equation 16 and 17, we have that

$$\inf R_n(\cdot) \geq 0 = R_n(g) \tag{18}$$

Thus, we have that $g(\omega)$ attains the infimum of $R_n(\cdot)$.

F Proof for Proposition 4.2

In general, instead of using the critical prior-mass radius $\varepsilon_{\pi,n}$ to describe certain characteristics of the Bayesian prior as in Corollary 5.2 of Zhang [55], we define and use the prior-mass radius d_π in Proposition 4.2, which is independent of the sample size n and measures the distance between the prior and true distribution.

Firstly, by definition of KL divergence: As $T \to \infty$, in Lemma 4.1

$$\sum_{t=1}^{T} \ln \frac{q(s^{t+1}, r^t | D_i^t, a^t)}{p(s^{t+1}, r^t | D_i^t, a^t; \omega_i)} \to T \cdot D_{KL}(q || p(\cdot | \omega_i))$$

Then, we use n instead of T to denote the number of samples collected, and we rewrite the original form for infimum of $R_n(\cdot)$ as:

$$\begin{aligned} \inf R_n(g) &= \inf[\mathbb{E}_\pi g(\omega_i) \cdot n \cdot D_{KL}(q || p(\cdot | \omega_i)) + D_{KL}(gd\pi || d\pi)] \\ &= \inf[\mathbb{E}_\pi g(\omega_i) D_{KL}(q || p(\cdot | \omega_i)) + \frac{1}{n} D_{KL}(gd\pi || d\pi)] \end{aligned} \tag{19}$$

Given Equation (19) and Following [55], we define the Bayesian resolvability as

$$\begin{aligned} r_n(q) &= \inf_g [\mathbb{E}_\pi g(\omega_i) D_{KL}(q || p(\cdot | \omega_i)) + \frac{1}{n} D_{KL}(gd\pi || d\pi)] \\ &= -\frac{1}{n} \ln \mathbb{E}_\pi e^{-n D_{KL}(q || p(\cdot | \omega_i))}. \end{aligned} \tag{20}$$

Intuitively, the Bayesian resolvability controls the complexity of the density estimation process. Based on this definition and our previous definitions of d_π, we can derive a simple and intuitive estimate of the standard Bayesian resolvability.

Lemma F.1. *The resolvability of standard Bayesian posterior defined in (20) can be bounded as*

$$r_n(q) \leq \frac{n+1}{n} d_\pi$$

proof. For all $d > 0$, we have

$$\begin{aligned} r_n(q) &= -\frac{1}{n} \ln \mathbb{E}_\pi e^{-n D_{KL}(q || p(\cdot | \omega_i))} \leq -\frac{1}{n} \ln[e^{-nd} \pi(p \in \Gamma : D_{KL}(q || p) \leq d)] \\ &= d + \frac{1}{n} \times [-\ln \pi(p \in \Gamma : D_{KL}(q || p) \leq d)] \leq \frac{n+1}{n} d_\pi \end{aligned}$$

□

This bound links the Bayesian resolvability to the number of samples n and prior-mass radius d_π which is a fixed property of the density given a specific prior and the true underlying density. Intuitively, the Bayesian posterior is better behaved when the Bayesian prior is closer to the true distribution (d_π is smaller) and more samples are used (n is larger).

Now we can prove the main theorem of Lemma 1. Let $\rho = \frac{1}{2}$, $\epsilon_h = \frac{2\varepsilon_n + (4\eta-2)h}{\delta/4}$, define $\Gamma_1 = \{p \in \Gamma : D_\rho^{Re}(q||p) < \epsilon_h\}$ and $\Gamma_2 = \{p \in: D_\rho^{Re}(q||p) \geq \epsilon_h\}$. We let $a = e^{-nh}$ and define $\pi'(\theta) = a\pi(\theta)C$ when $\theta \in \Gamma_1$ and $\pi'(\theta)C$ when $\theta \in \Gamma_2$, where the normalization constant $C = (a\pi(\Gamma_1) + \pi(\Gamma_2))^{-1} \in [1, 1/a]$. Firstly,

$$\mathbb{E}_X \pi'(\Gamma_2|X)\epsilon_h \leq \mathbb{E}_X \mathbb{E}_{\pi'} \pi'(\theta|X) \frac{1}{2}||p-q||_1^2 \leq \mathbb{E}_X \mathbb{E}_{\pi'} \pi'(\theta|X) D_{KL}(q||p)$$

according to the Markov inequality (with probability at least $1-\delta$) and Pinsker's inequality. Then, according to Theorem 5.2 and Proposition 5.2 in Zhang's paper,

$$\begin{aligned}
&\mathbb{E}_X \mathbb{E}_{\pi'} \pi'(\theta|X) D_{KL}(q||p) \leq \frac{\eta \ln \mathbb{E}_{\pi'} e^{-nD_{KL}(q||p(\cdot|\omega_i))}}{\rho(\rho-1)n} \\
&+ \frac{\eta-\rho}{\rho(1-\rho)n} \inf_{\{\Gamma_j\}} \ln \sum_j \pi'(\Gamma_j)^{(\eta-1)/(\eta-\rho)} (1 + r_{ub}(\Gamma_j))^n \\
&\leq \frac{\eta h - (\eta/n) \ln \mathbb{E}_\pi e^{-nD_{KL}(q||p(\cdot|\omega_i))}}{\rho(1-\rho)} + \frac{\eta-\rho}{\rho(1-\rho)} \Big[\frac{(\eta-1)h}{\eta-\rho} + \varepsilon_{upper,n} \Big(\frac{\eta-1}{\eta-\rho} \Big) \Big] \\
&= \frac{(2\eta-1)h}{\rho(1-\rho)} + \frac{-(\eta/n) \ln \mathbb{E}_\pi e^{-nD_{KL}(q||p(\cdot|\omega_i))} + (\eta-\rho)\varepsilon_{upper,n}((\eta-1)/(\eta-\rho))}{\rho(1-\rho)}
\end{aligned}$$

Then, using the definitions of d_π, we further obtain

$$\begin{aligned}
&\mathbb{E}_X \pi'(\Gamma_2|X)\epsilon_h \\
&\leq \frac{(2\eta-1)h}{\rho(1-\rho)} + \frac{\eta \inf_{d>0}[d - \frac{1}{n} \ln \pi(\{p \in \Gamma : D_{KL}(q||p) \leq d\})] + (\eta-\rho)\varepsilon_{upper,n}((\eta-1)/(\eta-\rho))}{\rho(1-\rho)} \\
&\leq \frac{(2\eta-1)h}{\rho(1-\rho)} + \frac{\eta(1+\frac{1}{n})d_\pi + (\eta-\rho)\varepsilon_{upper,n}((\eta-1)/(\eta-\rho))}{\rho(1-\rho)} \\
&= \frac{(2\eta-1)h + \varepsilon_n}{\rho(1-\rho)}
\end{aligned}$$

We use η instead of γ which is used in the original paper to avoid confusion with the discount factor. Then, we further divide both sides by ϵ_h and obtain $\pi'(\Gamma|X) \leq 0.5$. Then, by definition,

$$\begin{aligned}
\pi(\Gamma_2|X) &= a\pi'(\Gamma_2|X)/(1-(1-a)\pi'(\Gamma|X)) \\
&\leq \frac{a}{a+1} = \frac{1}{1+e^{nh}}
\end{aligned}$$

Thus, we get the desired bound.

G Proof for Proposition 4.4

The result shown in Proposition 4.4 can be derived simply by replacing the Bayesian concentration sample complexity term in BOSS with the result in Lemma 4.3. So the central part is the proof of Proposition 4.2, which we already did. The other parts are the same as the proof in BOSS, so we refer the readers to BOSS's original paper and omit the proof here. Note that the single-task baseline that BLRL is built upon is BOSS, and could be replaced by other Bayesian-exploration RL algorithm.

H Further experiments on MetaWorld

We include additional experiments on MT10 and MT50 domains of MetaWorld (v2) in Figure 10. MT10 and MT50 consist of 10 and 50 different categories of manipulation tasks, including reach, push, drawer close, button press etc. [4], and is a highly difficult setting for multi-task RL. We modified these two domains to be lifelong learning settings: instead of getting all 10/50 tasks at the same time at the beginning, we let them arrive sequentially. For instance, the first task is "reach" and we let the agent interact with it for 150 iterations, and then switch to "push" and the agent cannot collect data from "reach" anymore, after another 150 iterations the task changes to "pick-place", etc. We compare the results with LPG-FTW, T-HiP-MDP as well as Single-task MBRL (train the agent from scratch for each new task without using the world-model posterior to initialize the task specific model). The total running time on MT50 is around 48 hours on a Nvidia Geforce RTX 3090Ti GPU. With the limited number of interactions for each task, our proposed algorithm is still able to achieve reasonably good performance in such settings. In MT10, the total number of tasks (10) is relatively small so the world-model distribution may be far from accurate, but VBLRL still performs better than Single-task MBRL which does not use the world model to initialize the task-specific model. In MT50, the number of tasks is large enough (50) and we can see that VBLRL significantly outperforms all the other baselines. This is consistent with our intuition that sample efficiency improves in the task specific regime when the world model approaches the environment distribution. As the single-task model-based RL baseline we used here is relatively simple (CEM planning) to be comparable with the single-task baseline used by LPG-FTW, it is reasonable to expect the lifelong learning performance to be even better after combining our framework with model-based methods like Dreamer [20]; we leave such evaluations for future work.

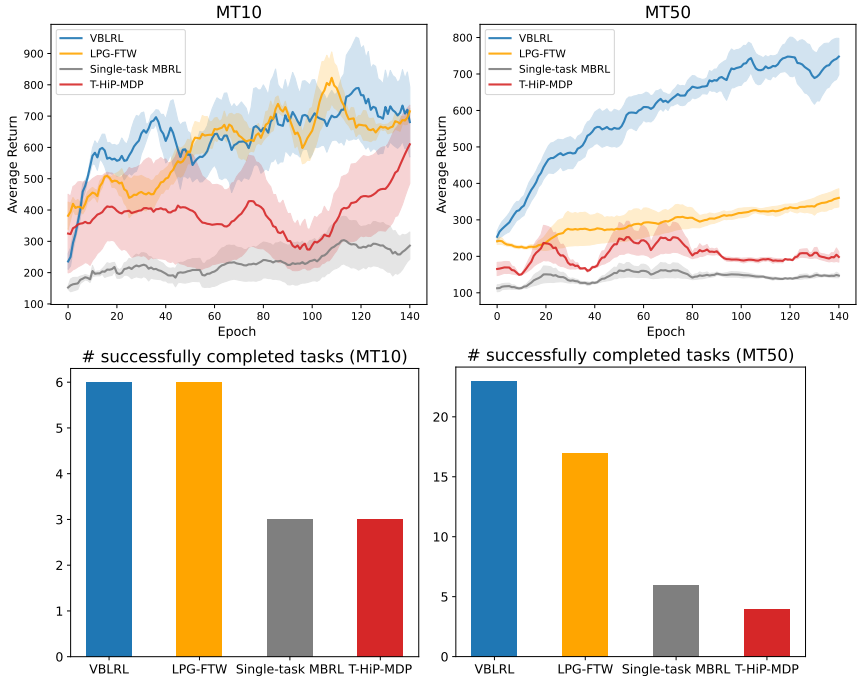

Figure 5: Top: Average performance comparison of VBLRL, LPG-FTW, T-HiP-MDP, Single-task MBRL on MT10 and MT50. Bottom: Comparison of the **maximum** number of successfully completed tasks on MT10 and MT50 for different method. The performanc of VBLRL is similar to LPG-FTW on MT10 but significantly outperforms LPG-FTW on MT50 as the world-model posterior becomes more accurate when the number of tasks is larger.

[4] https://meta-world.github.io/

I Single-task baseline comparison on Mujoco domain

In this section, we compare the performance of the single-task version of our algorithm with the single-task version of LPG-FTW/EWC. Note that to make it a fair comparison, we let the model-free single-task RL baseline used by LPG-FTW/EWC collect $2.0\times$ more samples (interactions with the environment) than VBLRL as in the lifelong learning setting.

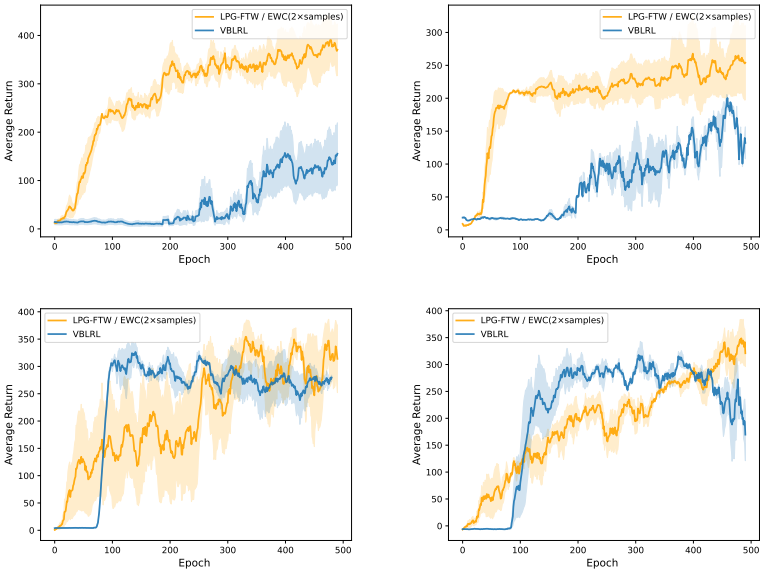

Figure 6: Average performance comparison for **single-task** baselines. Top-Left: **Hopper-Gravity**; Top-Right: **Hopper-Bodyparts**; Bottom-Left: **Walker-Gravity**; Bottom-Right: **Walker-Bodyparts**.

J Full lifelong RL comparison on Mujoco domain

As LPG-FTW and EWC are built upon a model-free RL baseline with relatively lower sample efficiency, we let LPG-FTW/EWC collect $2.0\times$ more samples (interactions with the environment) than VBLRL as in the single-task learning setting. Comparing Figure 6 and Figure 7, we find that in the Hopper domains, even though the single-task baseline used by LPG-FTW and EWC performs much better than VBLRL after we let them collect $2\times$ samples each iteration, in lifelong RL experiments VBLRL still achieves comparable performance with LPG-FTW and EWC. In the walker domains where the single task baselines achieves similar performance, VBLRL shows significant better performance than LPG-FTW/EWC in lifelong RL experiments.

K How VBLRL models different categories of uncertainty

L Grid-World Item Searching

We also evaluate BLRL in a simple Grid-World domain. Our testbed consists of a collection of houses, each of which has four rooms. The goal of each task is to find a specific object (blue, green or purple) in the current house. The type of each room is sampled based on an underlying distribution given by the environment. Each room type has a corresponding probability distribution of which kind of objects can be found in rooms of this type. Different tasks/houses vary in terms of which rooms are which types and precisely where objects are located in the room (the task's hidden parameters).

To simplify the problem, instead of modeling the whole MDP distribution, we use BLRL to model the object distribution as the Bayesian posterior and sample MDPs from the distribution. We use BOSS with a fixed prior (no intertask transfer) as our baseline. The average training performance of

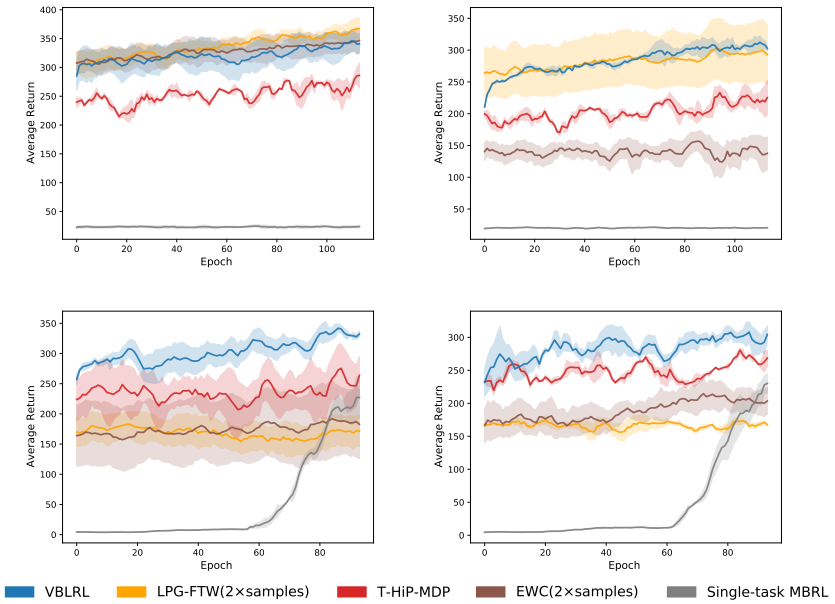

Figure 7: Average performance comparison for **lifelong RL algorithms**. Top-Left: **Hopper-Gravity**; Top-Right: **Hopper-Bodyparts**; Bottom-Left: **Walker-Gravity**; Bottom-Right: **Walker-Bodyparts**.

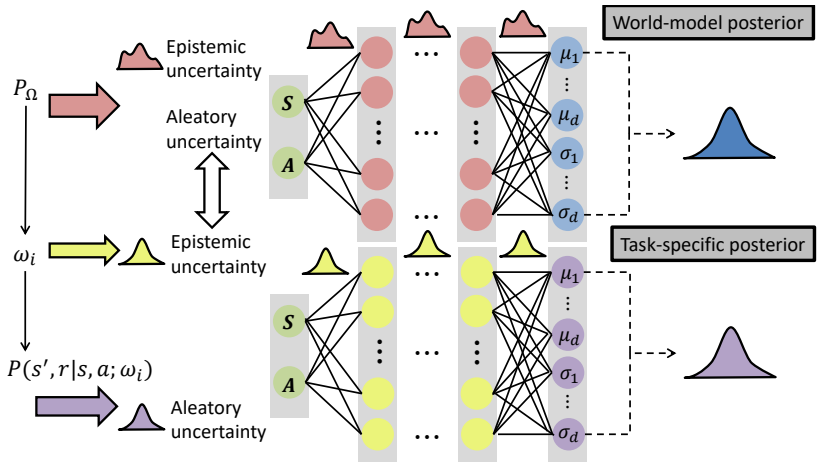

Figure 8: How VBLRL estimates different kinds of uncertainties in HiP-MDP. The world-model posterior captures the epistemic uncertainty of the general knowledge distribution (shared across all tasks controlled by the hidden parameters) via the internal variance of world-model BNN. As the learner is exposed to more and more tasks, the posterior should converge to P_Ω. The task-specific posterior captures the epistemic uncertainty of the current task i, which comes from the alleatory uncertainty of the world model when generating ω_i for a new task, via the internal variance of task-specific BNN. The posterior should output the highest probability for ω near the true ω_i as the agent collects enough data from the task. The aleatory uncertainty of the final prediction is measured by the output variance of the prediction.

all 300 tasks are shown in Figure 9 top right. Each task consists of 10 epochs, with 21 sample steps for each epoch. Within the limited steps allotted for each task, BLRL is able to discover and transfer the common knowledge and helps the agent quickly adapt to new tasks as the training goes on. In comparison, running BOSS with a fixed prior is able to find the optimal policy eventually but needs more sample steps and learns more slowly than BLRL.

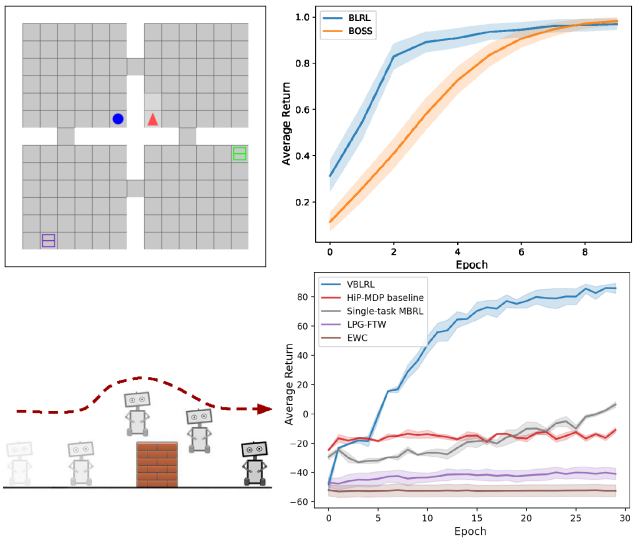

Figure 9: Top-left: Grid-World Item Searching; Top-right: Grid-World Item Searching evaluation results; Bottom-left: Box-jumping Task; Bottom-right: Box-jumping Task evaluation results.

M Box-Jumping Task

We use a simplified version of the jumping task [9] as a testbed for the proposed algorithm VBLRL. As shown in Figure 9 bottom left, the goal of the agent is to reach the right side of the screen by jumping over the obstacle. The agent can only choose from two actions: jump and right. It will hit the obstacle unless the jump action is chosen at precisely the right time. We set different obstacle positions as different tasks, constituting the HiP-MDP hidden parameters. The 4-element state vector describes the (x, y) coordinates of the agent's current position, and its velocity in the x and y directions.

Figure 9 bottom right presents the average performance during training across all 300 tasks. Each task is run for 30 episodes. VBLRL clearly learns faster than the HiP-MDP baseline and reaches better final performance. Thus, in the lifelong RL setting, separating the updating processes of the world-model posterior and the task-specific posterior can lead to better learning efficiency.

N Additional explanation of the algorithm

Here we first provide an example to help the readers better understand our plate notation. In our Gridworld Item Searching case, Ψ represents the parameters of P_Ω, which is the room-type and object distribution. For each task, the environment samples a hidden parameter ω, which is the actual room and object layout of this house, from this distribution P_Ω. The sampled ω then will result in an MDP m and let the agent interact with it.

N.1 planning algorithm

With the transition dynamics and reward functions, a planning algorithm like CEM is not the only way to solve the MDP to get an optimal policy. Another option would be using other Deep RL algorithms like Soft actor-critic [19] with data generated from the model. However, in this case, incorporating a deep RL algorithm means that we need to introduce additional neural networks (that is, policy/value networks) for each task. The update signal from the RL loss is usually stochastic and weak, which is

even worse in this case when our model is still far from accurate. So, here we assume applying a planning algorithm is a better way to get the policy.

O Ablation study of the number of particles

We also run ablation studies on the number of particles in CEM planning. The agent performance is close when the number of particles is around 50. The computational complexity drops as we use fewer particles and is especially larger when the number of particles ≥ 70.

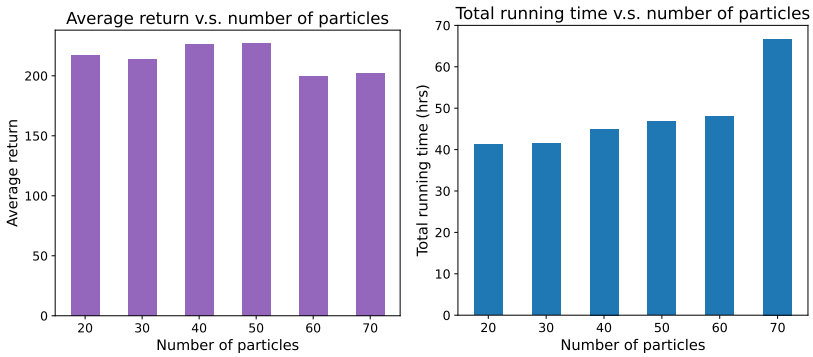

Figure 10: Left: How the average performance change with respect to different number of particles on Cheetah-Gravity domain. Right: How the total running hours on a Nvidia Geforce RTX 3090Ti change with respect to different number of particles on Cheetah-Gravity domain.

P Coin Example

Consider a coin-flipping environment. We want to find the sample complexity of the unbiased coin (i.e. How many times we need to flip this coin such that our posterior samples are accurate.). Consider a Dirichlet prior, $\alpha_0 = (n_1, n_2)$ and $\theta_0 = (\frac{1}{2}, \frac{1}{2})$. We want to find sample complexity B such that the posterior likelihood for a coin with heads likelihood in $[0.5 - \epsilon, 0.5 + \epsilon]$ is at least $1 - \delta$.

Note that the Dirichlet distribution on the two-dimensional simplex is the Beta distribution. The Multinomial distribution with two outcomes is the Binomial distribution. That is, given the process

$$H \sim Bin(H|\rho = 0.5, B), \tag{21}$$

$$\hat{\rho} \sim Beta(\hat{\rho}|\alpha = H + n_1, \beta = B - H + n_2), \tag{22}$$

choose a value B such that

$$P(0.5 - \epsilon \leq \hat{\rho} \leq 0.5 + \epsilon) \geq 1 - \delta, \tag{23}$$

$$\sum_{H=0}^{B} Bin(H|\rho = 0.5, B) \cdot \int_{\hat{\rho}=0.5-\epsilon}^{0.5+\epsilon} Beta(\hat{\rho}|\alpha = H + n_1, \beta = B - H + n_2) \geq 1 - \delta. \tag{24}$$

Here, n_1 and n_2 capture the prior. The smallest B that satisfies Equation 24 can be found numerically.

We set $\epsilon = 0.1$ and $\delta = 0.3$. Here are the results of sample complexity B given different values of n_1, n_2:

We fix the sum of (n_1, n_2) as 10. As shown in the results, the value of sample complexity B becomes lower as we use a more accurate prior (from $(10, 0)$ to $(5, 5)$ and from $(0, 10)$ to $(5, 5)$).

In general, for the task-specific posterior, we can relate B, ϵ and δ with the following equation:

$$\int_{P_0} Dir(P_0|\Phi^{True}) \Big[\sum_{\mathbf{N}:||\mathbf{N}||_1 = B} Mult(N|P_0, B) \Big[\int_{P:||P(\Phi) - P_0(\Phi)|| \leq \epsilon} Dir(P|\Phi_{old} + \mathbf{N}) dP \Big] \Big] dP_0 \geq 1 - \delta \tag{25}$$

(n_1, n_2)	lowest B
(0,10)	78
(1,9)	68
(2,8)	58
(3,7)	49
(4,6)	42
(5,5)	40
(6,4)	42
(7,3)	48
(8,2)	58
(9,1)	68
(10,0)	78

For the world model posterior:

$$\sum_{\mathbf{N}:||\mathbf{N}||_1=B_w} Mult(N|P_{w_0}, B_w)\Big[\int_{P:||P_w(\Phi)-P_{w_0}(\Phi)||\leq\epsilon} Dir(P_w|\Phi_{w_{old}}+\mathbf{N})dP_w\Big] \geq 1-\delta \quad (26)$$

For each task, first we pick a true model P_0 according to the true distribution and initialize the task-specific prior $\Phi_{old} = \Phi_w$. Then, we make some observations from the world. Once we have the true model and the observations, we can calculate how many models are ϵ-close to the true model, weighted according to their posterior likelihood.