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^p_τ 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^p_τ and $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} r^p_\tau$, 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^p_τ with $p_{m_i}(s', r|s,a)$
      Propagate state particles s^p_τ 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 $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} r^p_\tau$
    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 5 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.

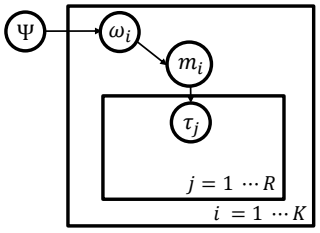

Figure 5: Plate representation for the BLRL approach. τ_j denotes trajectory $\{s, a, r, s'\}_j$. There are K different tasks and the agent samples R trajectories from each task.

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	$\{1, 20\}$	$\{1, 20\}$	$\{1, 20\}$	$\{1, 20\}$	$\{1, 20\}$	$\{1, 20\}$
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[2] 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

[2] 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 $\Gamma_2 = \{p \in \Gamma : D_\rho^{Re}(q||p) \geq \epsilon_h\}$. 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 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.

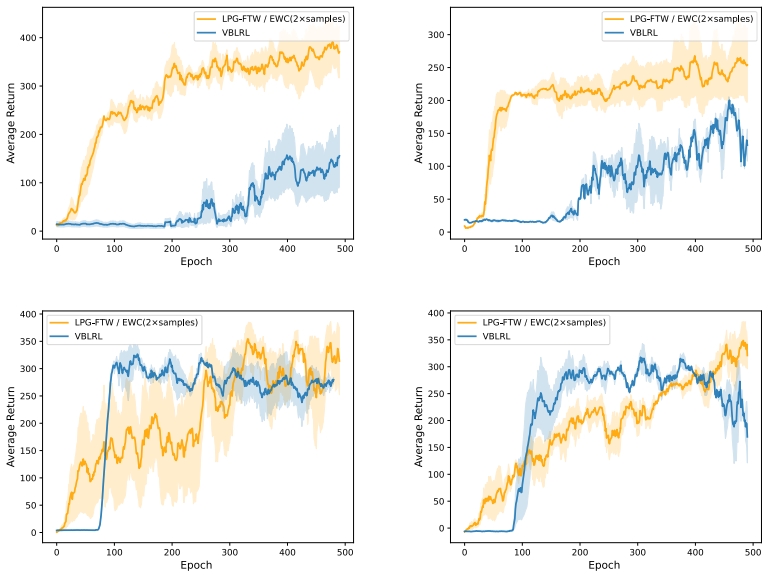

Figure 6: Average performance comparison for **single-task** baselines. Top-Left: **Hopper-Gravity**; Top-Right: **Hopper-Bodyparts**; Bottom-Left: **Walker-Gravity**; Bottom-Right: **Walker-Bodyparts**.

I 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.

J 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 all 300 tasks are shown in Figure 8 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

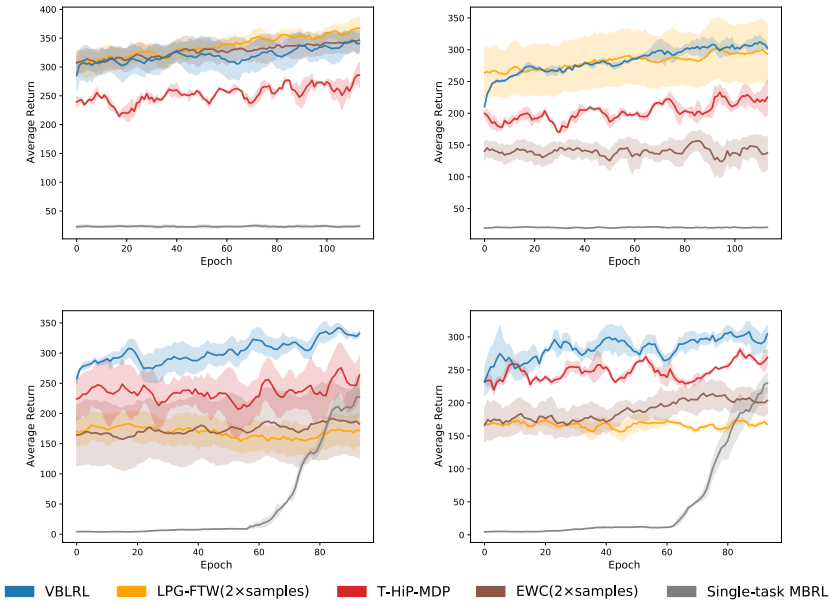

Figure 7: Average performance comparison for **lifelong RL algorithms**. Top-Left: **Hopper-Gravity**; Top-Right: **Hopper-Bodyparts**; Bottom-Left: **Walker-Gravity**; Bottom-Right: **Walker-Bodyparts**.

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.

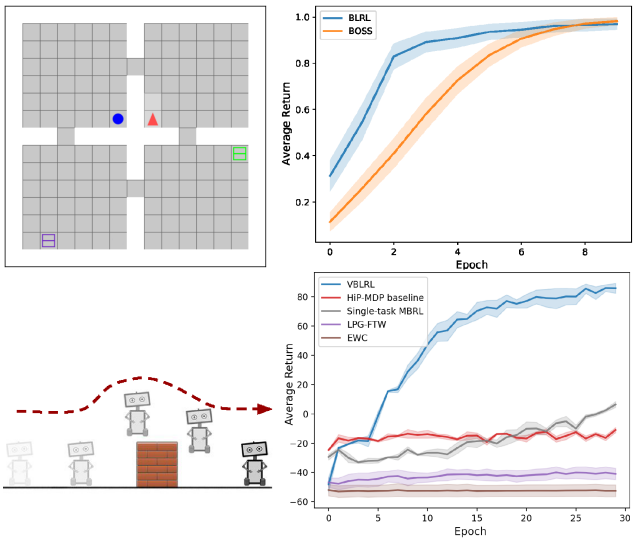

Figure 8: Top-left: Grid-World Item Searching; Top-right: Grid-World Item Searching evaluation results; Bottom-left: Box-jumping Task; Bottom-right: Box-jumping Task evaluation results.

K 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 8 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 8 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.

L How VBLRL models different categories of uncertainty

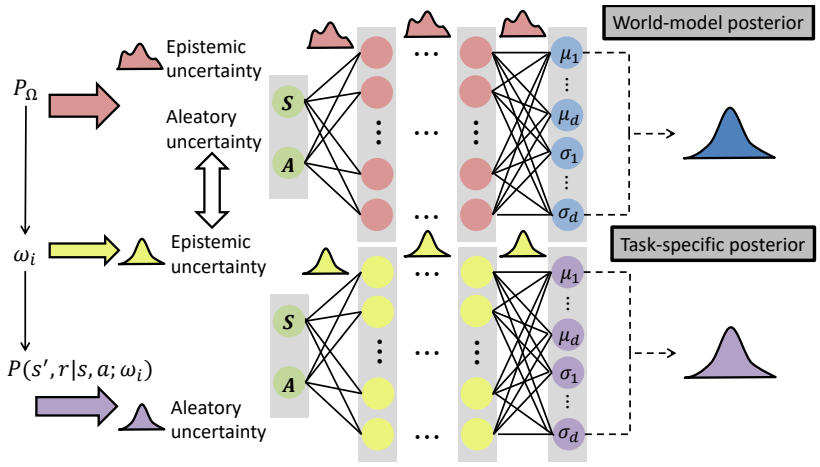

Figure 9: How VBLRL estimates different kinds of uncertainties in HiP-MDP. The world-model posterior captures the epistemic uncertainty of the general knowledge distribution (shared across all tasks controlled by the hidden parameters) via the internal variance of world-model BNN. As the learner is exposed 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.

M 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.

M.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.

N 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.

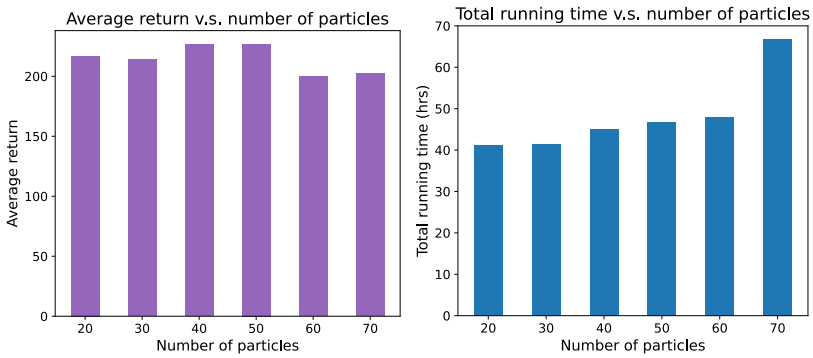

Figure 10: Left: How the average performance change with respect to different number of particles on Cheetah-Gravity domain. Right: How the total running hours on a Nvidia Geforce RTX 3090Ti change with respect to different number of particles on Cheetah-Gravity domain.

O 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:

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

For the world model posterior:

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

(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 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.