Journal of Machine Learning Research 22 (2021) 1-39 Submitted 6/21; Revised 9/21; Published 11/21

VariBAD: Variational Bayes-Adaptive Deep RL via Meta-Learning

Luisa Zintgraf LUISA.ZINTGRAF@CS.OX.AC.UK
University of Oxford
Wolfson Building, Parks Road, OX1 3QD Oxford (UK)

Sebastian Schulze SEBASTIAN.SCHULZE@ENG.OX.AC.UK
University of Oxford

Cong Lu CONG.LU@STATS.OX.AC.UK
University of Oxford

Leo Feng LEO.FENG@MILA.QUEBEC
Mila, Université de Montréal

Maximilian Igl MIGL@GOOGLE.COM
University of Oxford; Waymo

Kyriacos Shiarlis KYRIACOS@GOOGLE.COM
Waymo

Yarin Gal YARIN@CS.OX.AC.UK
University of Oxford

Katja Hofmann KATJA.HOFMANN@MICROSOFT.COM
Microsoft Research

Shimon Whiteson SHIMON.WHITESON@CS.OX.AC.UK
University of Oxford

Editor: George Konidaris

Abstract

Trading off exploration and exploitation in an unknown environment is key to maximising expected online return during learning. A Bayes-optimal policy, which does so optimally, conditions its actions not only on the environment state but also on the agent's uncertainty about the environment. Computing a Bayes-optimal policy is however intractable for all but the smallest tasks. In this paper, we introduce variational Bayes-Adaptive Deep RL (variBAD), a way to meta-learn approximately Bayes-optimal policies for complex tasks. VariBAD simultaneously meta-learns a variational auto-encoder to perform approximate inference, and a policy that incorporates task uncertainty directly during action selection by conditioning on both the environment state and the approximate belief. In two toy domains, we illustrate how variBAD performs structured online exploration as a function of task uncertainty. We further evaluate variBAD on MuJoCo tasks widely used in meta-RL and show that it achieves higher online return than existing methods. On the recently proposed Meta-World ML1 benchmark, variBAD achieves state of the art results by a large margin, fully solving two out of the three ML1 tasks for the first time.

Keywords: Reinforcement Learning, Meta Learning, Bayes-Adaptive Markov Decision Processes, Approximate Variational Inference, Recurrent Networks

1. Introduction

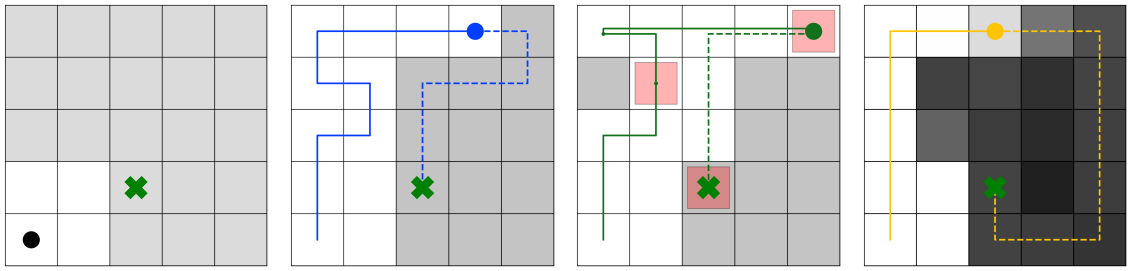
Reinforcement learning (RL) is typically concerned with finding an *optimal policy* that maximises expected return for a given Markov decision process (MDP) with an unknown reward and transition function. If these were known, the optimal policy could in theory be computed without environment interactions. By contrast, learning in an *unknown* environment usually requires trading off exploration (learning about the environment) and exploitation (taking promising actions). Balancing this trade-off is key to maximising expected return *during learning*, which is desirable in many settings, particularly in high-stakes real-world applications like healthcare and education (Liu et al., 2014; Yauney and Shah, 2018). A *Bayes-optimal policy*, which does this trade-off optimally, conditions actions not only on the environment state but on the agent's own uncertainty about the current MDP.

In principle, a Bayes-optimal policy can be computed using the framework of Bayes-adaptive Markov decision processes (BAMDPs) (Martin, 1967; Duff and Barto, 2002), in which the agent maintains a belief distribution over possible environments. Augmenting the state space of the underlying MDP with this belief yields a BAMDP, a special case of a belief MDP (see Kaelbling et al. (1998) for a discussion of belief MDPs in the context of partially-observable MDPs). A Bayes-optimal agent maximises expected return in the BAMDP by systematically seeking out the data needed to quickly reduce uncertainty, but only insofar as doing so helps maximise expected return. Its performance is bounded from above by the optimal policy for the given MDP, which does not need to take exploratory actions but requires prior knowledge about the MDP to compute.

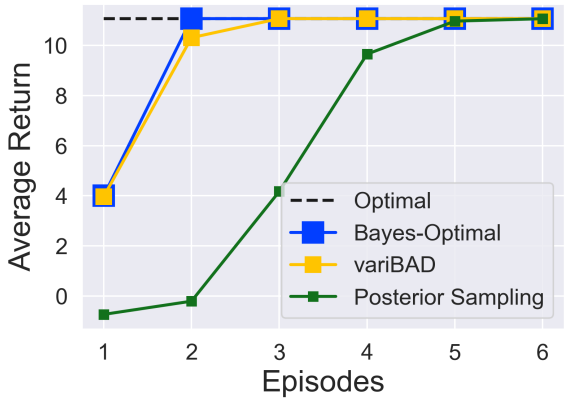
Unfortunately, planning in a BAMDP, i.e., computing a Bayes-optimal policy that conditions on the augmented state, is intractable for all but the smallest tasks. A common shortcut is to rely instead on *posterior sampling* (Thompson, 1933; Strens, 2000; Osband et al., 2013). Here, the agent periodically samples a single hypothesis MDP (e.g., at the beginning of an episode) from its posterior, and the policy that is optimal *for the sampled MDP* is followed until the next sample is drawn. Planning is far more tractable since it is done on a regular MDP, not a BAMDP. However, posterior sampling's exploration can be highly inefficient and far from Bayes-optimal.

Consider the example of a gridworld in Figure 1, where the agent must navigate to an *unknown* goal located in the grey area (1a). To maintain a posterior, the agent can uniformly assign non-zero probability to cells where the goal could be, and zero to all other cells. A Bayes-optimal strategy searches the set of goal positions that the posterior considers possible, until the goal is found (1b). Posterior sampling samples a possible goal position, takes the shortest route there, and then resamples a new goal from the updated posterior (1c). This example illustrates that posterior sampling is much less efficient since the agent's uncertainty is not reduced optimally (e.g., states are revisited unnecessarily). A key challenge is to learn approximately Bayes-optimal policies while retaining the tractability of posterior sampling. In addition, the inference involved in maintaining a posterior belief, needed even for posterior sampling, may itself be intractable.

In this paper, we combine ideas from Bayesian RL, approximate variational inference, and meta-learning to tackle these challenges, and equip an agent with the ability to strategically explore unseen (but related) environments for a given distribution, in order to maximise its expected online return.



(a) Environment (b) Bayes-Optimal (c) Posterior Sampling (d) variBAD



(e) Performance

Figure 1: Illustration of different exploration strategies. (a) Environment: The agent starts at the bottom left and has to navigate to an unknown goal, located in the grey area. (b) A Bayes-optimal exploration strategy that systematically searches possible grid cells to find the goal, shown in solid (past actions) and dashed (future actions) blue lines. A simplified posterior is shown in the background in grey ($p = 1/$(number of possible goal positions left) of containing the goal) and white ($p = 0$). (c) Posterior sampling, which repeatedly samples a possible goal position (red squares) from the current posterior, takes the shortest route to the sampled goal, and then updates its posterior. (d) Exploration strategy learned by variBAD. The grey background represents the approximate posterior the agent has learned. The predictions are not perfectly calibrated, and the model generally predicts slightly higher probabilities, shown by the darker shade compared to (b). (e) Average return over all possible environments, over six episodes with 15 steps each (after which the agent is reset to the starting position). VariBAD results are averaged across 20 random seeds. The performance of any exploration strategy is bounded above by the optimal behaviour (of a policy with access to the true goal position). The Bayes-optimal agent matches this behaviour from the second episode, whereas posterior sampling needs six rollouts. VariBAD closely approximates Bayes-optimal behaviour in this environment.

More specifically, we propose *variational Bayes-Adaptive Deep RL* (variBAD), a way to meta-learn to perform approximate inference on an unknown task,[1] and incorporate task uncertainty directly during action selection. Given a distribution over MDPs $p(M)$, we represent a single MDP M using a learned, low-dimensional stochastic latent variable m and simultaneously meta-train:

1. A variational auto-encoder that can infer the posterior distribution over m in a new task, given the agent's experience, while interacting with the environment, and

2. A policy that conditions on this posterior belief over MDP embeddings, and thus learns how to trade off exploration and exploitation when selecting actions *under task uncertainty*.

Figure 1e shows the performance of our method versus the hard-coded optimal strategy (with privileged goal information), the Bayes-optimal strategy, and posterior sampling strategy. VariBAD's performance closely matches the Bayes-optimal one, matching optimal performance from the third rollout.

We evaluate our approach on two toy domains, the GridWorld shown above and a 2D PointRobot Navigation task, which illustrate how variBAD maximises expected online return using the meta-learned approximate belief. We further evaluate variBAD on the widely used MuJoCo meta-RL benchmarks, and show that variBAD exhibits superior exploratory behaviour at test time compared to existing methods, achieving higher returns during learning. Lastly, on the recently proposed challenging Meta-World ML1 benchmark, variBAD achieves state of the art performance with a large margin compared to existing methods, fully solving two out of the three ML1 benchmark tasks for the first time. As such, variBAD opens a path to tractable approximate Bayes-optimal exploration for deep reinforcement learning.[2]

2. Background

We define a Markov decision process (MDP) as a tuple $M = (\mathcal{S}, \mathcal{A}, R, T, T_0, \gamma, H)$ with $\mathcal{S}$ a set of states, $\mathcal{A}$ a set of actions, $R(r_{t+1}|s_t, a_t, s_{t+1})$ a reward function, $T(s{=}s_{t+1}|s_t, a_t)$ a transition function, $T_0(s{=}s_0)$ an initial state distribution, γ a discount factor, and H the horizon.[3] In the standard RL setting, we want to learn a policy π that maximises $\mathcal{J}(\pi) = \mathbb{E}_{T_0,T,\pi}\left[\sum_{t=0}^{H-1} \gamma^t R(r_{t+1}|s_t, a_t, s_{t+1})\right]$, the expected return. Here, we consider a multi-task meta-learning setting, which we introduce next.

1. We use the terms environment, task, and MDP, interchangeably.
2. This paper extends the conference paper titled "VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning" published in the Proceedings of the International Conference on Learning Representations (ICLR) 2020 by Zintgraf, Shiarlis, Igl, Schulze, Gal, Hofmann, and Whiteson (Zintgraf et al., 2020). The journal version substantially extends the experimental evaluation, adding results on: Sparse 2D Navigation, MuJoCo AntGoal and Humanoid, and Meta-World ML1. Additionally, we perform empirical analyses over choices in our loss function (modelling horizon and KL regularisation), and study the robustness of the latent dimensionality.
3. Henceforth $T(s{=}s_{t+1}|s_t, a_t)$ and $T_0(s{=}s_0)$ are denoted $T(s_{t+1}|s_t, a_t)$ and $T_0(s_0)$ for brevity.

2.1 Training Setup

We adopt the standard meta-learning setting where we have a distribution $p(M)$ over MDPs from which we can sample during meta-training, with an MDP $M_i \sim p(M)$ defined by a tuple $M_i = (\mathcal{S}, \mathcal{A}, R_i, T_i, T_{i,0}, \gamma, H)$. Across tasks, the reward and transition functions can vary, but typically share some structure. The index i represents an unknown task description (e.g., a goal position or natural language instruction) or task ID. Sampling an MDP from $p(M)$ is typically done by sampling a reward and transition function from a distribution $p(R, T)$. During meta-training, we aim to maximise the expected adaptation performance across MDPs. That is, batches of tasks are repeatedly sampled, and a small training procedure is performed on each of them, with the goal of learning to learn. At meta-test time, the agent is evaluated based on the average online return it achieves within a fixed amount of time on a new task drawn from p, requiring it to learn fast and trade-off efficiently between exploration and exploitation. Doing this well requires at least two things: (1) incorporating prior knowledge obtained in related tasks, and (2) reasoning about task uncertainty when selecting actions to trade off exploration and exploitation. In the following, we combine ideas from meta-learning and Bayesian RL to tackle these challenges.

2.2 Bayesian Reinforcement Learning

When the MDP is unknown, optimal decision making has to trade off exploration and exploitation when selecting actions. In principle, this can be done by taking a Bayesian approach to reinforcement learning formalised as a Bayes-Adaptive MDP (BAMDP), the solution to which is a Bayes-optimal policy (Bellman, 1956; Duff and Barto, 2002; Ghavamzadeh et al., 2015).

In the Bayesian formulation of RL, we assume that the transition and reward functions are distributed according to a prior $b_0 = p(R, T)$. Since the agent does not have access to the true reward and transition function, it can maintain a belief

$$b_t(R, T) = p(R, T|\tau_{:t}),$$

which is the posterior over the MDP given the agent's experience $\tau_{:t} = \{s_0, a_0, r_1, s_1, \ldots, s_t\}$ up until the current timestep t. This is often done by maintaining a distribution over the model parameters.

To allow the agent to incorporate the task uncertainty into its decision-making, this belief can be augmented to the state, resulting in hyper-states $s_t^+ \in \mathcal{S}^+ = \mathcal{S} \times \mathcal{B}$, where $\mathcal{B}$ is the belief space. These transition according to

$$\begin{aligned} T^+(s_{t+1}^+|s_t^+, a_t, r_t) &= T^+(s_{t+1}, b_{t+1}|s_t, a_t, r_t, b_t) \\ &= T^+(s_{t+1}|s_t, a_t, b_t)\ T^+(b_{t+1}|s_t, a_t, r_t, b_t, s_{t+1}) \\ &= \mathbb{E}_{b_t}\left[T(s_{t+1}|s_t, a_t)\right]\ \delta(b_{t+1} = p(R, T|\tau_{:t+1})), \end{aligned}$$

i.e., the new environment state s_t is the expected new state w.r.t. the current posterior distribution of the transition function, and the belief is updated deterministically according to Bayes rule. The reward function on hyper-states is defined as the expected reward under the current posterior (after the state transition) over reward functions,

$$R^+(s_t^+, a_t, s_{t+1}^+) = R^+(s_t, b_t, a_t, s_{t+1}, b_{t+1}) = \mathbb{E}_{b_{t+1}}\left[R(s_t, a_t, s_{t+1})\right].$$

This results in a BAMDP $M^+ = (\mathcal{S}^+, \mathcal{A}, R^+, T^+, T_0^+, \gamma, H^+)$ (Duff and Barto, 2002), which is a special case of a belief MDP, i.e., the MDP formed by taking the posterior beliefs maintained by an agent in a partially observable MDP and reinterpreting them as Markov states (Cassandra et al., 1994). In an arbitrary belief MDP, the belief is over a hidden state that can change over time. In a BAMDP, the belief is over the transition and reward functions, which are constant for a given task. The agent's objective is now to maximise the expected return in the BAMDP,

$$\mathcal{J}^+(\pi) = \mathbb{E}_{b_0, T_0^+, T^+, \pi} \left[\sum_{t=0}^{H^+-1} \gamma^t R^+(r_{t+1} | s_t^+, a_t, s_{t+1}^+) \right], \tag{1}$$

i.e., maximise the expected return in an initially unknown environment, while learning, within the horizon H^+. The MDP horizon H is distinct from the BAMDP horizon H^+. Although they often coincide, we might instead want the agent to act Bayes-optimal within the first N MDP episodes, so $H^+ = N \times H$. Trading off exploration and exploitation optimally depends heavily on how much time the agent has left (e.g., to decide whether information-seeking actions are worth it).

The objective in Equation 1 is maximised by the Bayes-optimal policy, which automatically trades off exploration and exploitation: it takes exploratory actions to reduce its task uncertainty *only insofar as it helps to maximise the expected return within the horizon* by improving expected future returns. The BAMDP framework is powerful because it provides a principled way of formulating Bayes-optimal behaviour. However, solving the BAMDP is hopelessly intractable for most interesting problems. The main challenges are as follows.

- We typically do not know how to parameterise the reward and/or transition model,
- The belief update (computing the posterior $p(R, T|\tau_{:t})$) is often intractable, and
- Even with the correct posterior, planning in belief space is typically intractable.

In the following, we propose a method that simultaneously meta-learns the reward and transition functions, how to perform inference in an unknown MDP, and how to use the belief to maximise expected online return. Since the Bayes-adaptive policy is learned end-to-end with the inference framework, no planning is necessary at test time. We make minimal assumptions (no privileged task information is required during training), resulting in a highly flexible and scalable approach to Bayes-adaptive Deep RL.

3. Bayes-Adaptive Deep RL via Meta-Learning

In this section, we present variBAD and describe how we tackle the challenges outlined above. We start by describing how to represent reward and transition functions, and (posterior) distributions over these. We then consider how to meta-learn to perform approximate variational inference in a given task, and finally put all the pieces together to form our training objective.

In the typical meta-learning setting, the reward and transition functions that are unique to each MDP are unknown, but also share some structure across the MDPs M_i in $p(M)$. We know that there exists a true i which represents either a task description or task ID,

but we do not have access to this information. We therefore represent this value using a learned stochastic latent variable m_i. For a given MDP M_i we can then write

$$R_i(r_{t+1}|s_t, a_t, s_{t+1}) \approx R(r_{t+1}|s_t, a_t, s_{t+1}; m_i), \tag{2}$$
$$T_i(s_{t+1}|s_t, a_t) \approx T(s_{t+1}|s_t, a_t; m_i), \tag{3}$$

where R and T are shared across tasks. Since we do not have access to the true task description or ID, we need to *infer* m_i given the agent's experience up to time step t collected in M_i,

$$\tau^{(i)}_{:t} = (s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{t-1}, a_{t-1}, r_t, s_t),$$

i.e., we want to infer the posterior distribution $p(m_i|\tau^{(i)}_{:t})$ over m_i given $\tau^{(i)}_{:t}$ (from now on, we drop the sub- and superscript i for ease of notation).

Recall that our goal is to *learn a belief over the MDPs, and given a posteriori knowledge of the environment compute the optimal action.* Given the above reformulation, it is now sufficient to reason about the embedding m, instead of the transition and reward dynamics. This is particularly useful when deploying deep learning strategies, where the reward and transition function can consist of millions of parameters, but the embedding m can be a small vector.

3.1 Approximate Inference

Computing the exact posterior is typically not possible: we do not have access to the MDP (and hence the transition and reward function), and marginalising over tasks is computationally infeasible. Consequently, we need to learn a model of the environment $p_\theta(\tau_{:H^+}|a_{:H^+-1})$, parameterised by θ, together with an amortised inference network $q_\phi(m|\tau_{:t})$, parameterised by ϕ, which allows fast inference at runtime *at each timestep* t. The action-selection policy is not part of the MDP, so an environmental model can only give rise to a distribution of trajectories when conditioned on actions, which we typically draw from our current policy, $a \sim \pi$. At any given time step t, our model learning objective is thus to maximise

$$\mathbb{E}_{\rho(M,\tau_{:H^+})}\left[\log p_\theta(\tau_{:H^+}|a_{:H^+-1})\right], \tag{4}$$

where $\rho(M, \tau_{:H^+})$ is the trajectory distribution induced by our policy and we slightly abuse notation by denoting by τ the state-reward trajectories, excluding the actions. In the following, we drop the conditioning on $a_{:H^+-1}$ to simplify notation.

Instead of optimising Equation 4 directly, which is intractable, we can optimise a tractable lower bound, defined with a learned approximate posterior $q_\phi(m|\tau_{:t})$ which can be estimated by Monte Carlo sampling (for the full derivation see AppendixA):

$$\begin{aligned}\mathbb{E}_{\rho(M,\tau_{:H^+})}\left[\log p_\theta(\tau_{:H^+})\right] &\geq \mathbb{E}_\rho\left[\mathbb{E}_{q_\phi(m|\tau_{:t})}\left[\log p_\theta(\tau_{:H^+}|m)\right] - KL(q_\phi(m|\tau_{:t})||p_\theta(m))\right] \\ &= ELBO_t.\end{aligned} \tag{5}$$

The term $\mathbb{E}_q[\log p(\tau_{:H^+}|m)]$ is often referred to as the reconstruction loss, and $p_\theta(\tau_{:t}|m)$ as the decoder. The term $KL(q(m|\tau_{:t})||p_\theta(m))$ is the KL-divergence between our variational posterior q_ϕ and the prior over the embeddings $p_\theta(m)$.

Instead of using the same fixed prior for each timestep and ELBO_t, we set the prior to our previous posterior, $q_\phi(m|\tau_{:t-1})$, with initial prior $q_\phi(m) = \mathcal{N}(0, I)$. We thus have $KL(q(m|\tau_{:t})||q(m|\tau_{t-1}))$. This is akin to a Bayesian filtering update, and works better empirically, as we confirm in Section 6.2. We take the gradient w.r.t. the entire term (i.e., through the current and previous approximate posterior).

As can be seen in Equation 5 and Figure 2, when the agent is at timestep t, we encode the *past trajectory* $\tau_{:t}$ to get the current posterior $q(m|\tau_{:t})$ since this is all the information available to perform inference about the current task. We then decode the entire trajectory $\tau_{:H^+}$ *including the future*, i.e., model $\mathbb{E}_q[p(\tau_{:H^+}|m)]$. This is different than the conventional VAE setup (and possible since we have access to this information during training). Decoding not only the past but also the future is important because this way, variBAD learns to perform inference about unseen states given the past. The reconstruction term $\log p(\tau_{:H^+}|m)$ factorises as

$$\begin{aligned}\log p(\tau_{:H^+}|m, a_{:H^+-1}) &= \log p((s_0, r_0, \ldots, s_{t-1}, r_{t-1}, s_t)|m, a_{:H^+-1}) \\ &= \log p(s_0|m) + \sum_{i=0}^{H^+-1} \left[\log p(s_{i+1}|s_i, a_i, m) + \log p(r_{i+1}|s_i, a_i, s_{i+1}, m)\right].\end{aligned} \tag{6}$$

Here, $p(s_0|m)$ is the initial state distribution T_0', $p(s_{i+1}|s_i, a_i; m)$ the transition function T', and $p(r_{i+1}|s_t, a_t, s_{i+1}; m)$ the reward function R'. Below, we include T_0' in T' for ease of notation.

3.2 Training Objective

We can now formulate a training objective for learning the approximate posterior distribution over task embeddings, the policy, and the generalised reward and transition functions R' and T'. We use deep neural networks to represent the individual components:

1. The encoder $q_\phi(m|\tau_{:t})$, parameterised by ϕ;
2. An approximate transition function $T' = p_\theta^T(s_{i+1}|s_i, a_i; m)$ and an approximate reward function $R' = p_\theta^R(r_{i+1}|s_t, a_t, s_{i+1}; m)$ which are jointly parameterised by θ; and
3. A policy $\pi_\psi(a_t|s_t, q_\phi(m|\tau_{:t}))$ parameterised by ψ and dependent on ϕ.

An overview of the network architecture is shown in Figure 2.

The policy $\pi_\psi(a_t|s_t, q_\phi(m|\tau_{:t}))$ is conditioned on the environment state s_t and the current approximate belief $b_t \equiv q_\phi(m||\tau_{:t})$ over m. This is similar to the formulation of BAMDPs introduced in Section 2.2, with the difference that we learn a unifying distribution over MDP embeddings, instead of the transition/reward function directly. This makes learning easier since there are fewer parameters to perform inference over, and we can use data from all tasks to learn the shared reward/transition function. The posterior can be represented by the distribution's parameters (e.g., mean and standard deviation if q is Gaussian).

Our overall objective is to maximise

$$\mathcal{L}(\phi, \theta, \psi) = \mathbb{E}_{p(M)}\left[\mathcal{J}(\psi, \phi) + \lambda \sum_{t=0}^{H^+} ELBO_t(\phi, \theta)\right], \tag{7}$$

where $J(\psi, \phi)$ is the expected return as defined in Section 2.

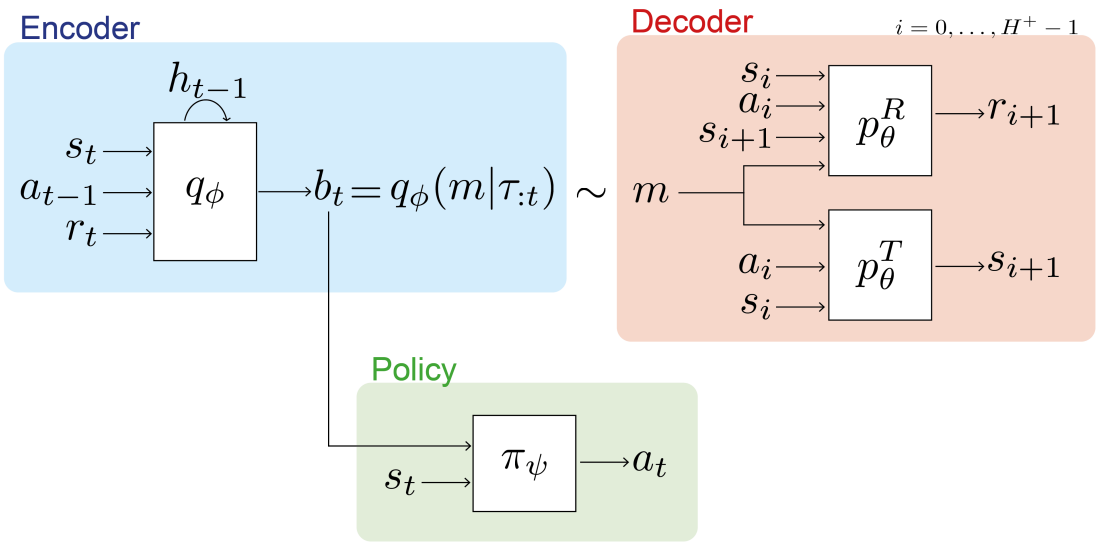


Figure 2: VariBAD architecture: A trajectory of states, actions and rewards is processed online using an RNN to produce the posterior over task embeddings, $q_\phi(m|\tau_{:t})$. The posterior is trained using a decoder which predicts past and future states and rewards from current states and actions. The policy conditions on the posterior in order to act in the environment and is trained using RL.

3.3 Meta Training

During meta-training, we train the policy and the VAE using Equation 7.

In Equation 7, λ weights the supervised model learning objective against the RL loss. This is necessary since parameters ϕ are shared between the VAE and the policy. However, we found that backpropagating the RL loss through the encoder is typically unnecessary in practice. Not doing so speeds up training considerably, avoids the need to trade off these losses, and prevents interference between gradients of opposing losses. In ablation studies (see Appendix C.5) we found that backpropagating both losses through the encoder can marginally improve performance in some cases, but can be detrimental in others, and depends strongly on the relative weighting between the VAE and the RL loss. In our experiments, we therefore optimise the policy and VAE alternatingly, with separate optimisers.

Meta-Training the VAE. The VAE is trained by approximating expectations in Equation 7 using Monte Carlo samples, and using the reparameterisation trick (Kingma and Welling, 2014). For $t = 0$, we use the prior $q_\phi(m) = \mathcal{N}(0, I)$. We encode past trajectories using a recurrent network, but other types of encoders could be considered like the ones used in Zaheer et al. (2017); Garnelo et al. (2018); Rakelly et al. (2019). The encoder outputs an approximate belief over the current task m, by predicting the mean and variance of a Gaussian distribution. The decoder is trained using samples from this approximate posterior, and by separately predicting the rewards and state transitions of the entire current trajectory (i.e., including future steps to which we have access during meta-training).

Equation 7 shows that the ELBO appears for all possible context lengths t. This way, variBAD can learn to perform inference online (while the agent is interacting with an environment), and decrease its uncertainty over time given more data. In practice, we may subsample a fixed number of ELBO terms for computational efficiency if H^+ is large.

Meta-Training the Policy. We meta-train the policy using proximal policy optimisation (Schulman et al., 2017, PPO), but other methods can in principle be used, including off-policy methods (see Dorfman et al. (2020)). The policy receives the approximate belief as an input in addition to the state, allowing it to meta-learn how to act approximately Bayes-optimal. This approximate belief, $q_\phi(m|\tau_{:t})$, is represented by two vectors: the mean and variance of a Gaussian distribution with diagonal covariance matrix. These two vectors are predicted by the encoder, and are concatenated with the environment state before being passed to the policy. In the default case, this is treated as part of the observation, and the RL loss is not backpropagated through the encoder.

We train the RL agent and the VAE using different data buffers: the policy is only trained with the most recent data since we use an on-policy algorithm in our experiments; and for the VAE we maintain a separate, larger buffer of observed trajectories.

3.4 Meta Testing

At meta-test time, we roll out the policy in randomly sampled test tasks to evaluate performance, by performing forward passed through the encoder (to compute the approximate belief) and the policy (to choose how to act given the state and approximate belief). The belief is updated after every step by feeding in the new (state, action, reward) tuple, allowing the agent to adapt online. The decoder is not used at test time, and no gradient updates are performed on the encoder or policy network: the agent has learned to act approximately Bayes-optimal during meta-training.

4. Related Work

Bayesian Reinforcement Learning. Bayesian methods for RL can be used to quantify uncertainty to support action-selection, and provide a way to incorporate prior knowledge into the algorithms (we refer the reader Ghavamzadeh et al. (2015) for a review). A Bayes-optimal policy is one that optimally trades off exploration and exploitation, and thus maximises expected online return during learning. While such a policy can in principle be computed using the BAMDP framework, it is unfortunately hopelessly intractable for all but the smallest tasks. Existing methods are therefore restricted to small and discrete state and action spaces (Asmuth and Littman, 2011; Guez et al., 2012, 2013), or a discrete set of tasks (Brunskill, 2012; Poupart et al., 2006).

VariBAD opens a path to tractable approximate Bayes-optimal exploration for deep RL by leveraging ideas from meta-learning and approximate variational inference, with the only assumption that we can meta-train on a set of related tasks. Existing approximate Bayesian RL methods often require us to define a prior or belief update on the reward and transition functions, and rely on (possibly expensive) sample-based planning procedures. Due to the use of deep neural networks however, variBAD lacks the formal guarantees enjoyed by some of the methods mentioned above.

Posterior sampling (Strens, 2000; Osband et al., 2013), which extends Thompson sampling (Thompson, 1933) from bandits to MDPs, estimates a posterior distribution over MDPs, in the same spirit as variBAD. This posterior is used to periodically sample a single hypothesis MDP (e.g., at the beginning of an episode), and the policy that is optimal *for the sampled MDP* is followed subsequently. This approach is less efficient than Bayes-optimal behaviour and therefore typically has lower expected return during learning.

A related approach for inter-task transfer of abstract knowledge is to pose policy search with priors as Markov Chain Monte Carlo inference (Wingate et al., 2011). Similarly Guez et al. (2013) propose a Monte Carlo Tree Search method for Bayesian planning for tractable, sample-based approximately Bayes-optimal behaviour. Osband et al. (2018) note that non-Bayesian treatment for decision making can be arbitrarily suboptimal and propose a simple randomised prior based approach for structured exploration. Some recent deep RL methods use stochastic latent variables for structured exploration (Gupta et al., 2018; Rakelly et al., 2019), which gives rise to behaviour similar to posterior sampling. Other ways to use the posterior for exploration are, e.g., certain reward bonuses (Kolter and Ng, 2009; Sorg et al., 2012) and methods based on optimism in the face of uncertainty (Kearns and Singh, 2002; Brafman and Tennenholtz, 2002). Non-Bayesian methods for exploration are often used in practice, such as other exploration bonuses (e.g., via state-visitation counts) or using uninformed sampling of actions (e.g., ϵ-greedy action selection). Such methods are prone to wasteful exploration that does not help maximise expected reward.

Related to BAMDPs are *contextual MDPs*, where the task description is referred to as a context, on which the environment dynamics and rewards depend (Hallak et al., 2015; Jiang et al., 2017; Dann et al., 2019; Modi and Tewari, 2019). Research in this area is often concerned with developing tight bounds by putting assumptions on the context, such as having a small known number of contexts, or that there is a linear relationship between the contexts and dynamics/rewards. Similarly, the framework of hidden parameter (HiP-) MDPs assumes that there is a set of low-dimensional latent factors which define a family of related dynamical systems (with shared reward structure). The assumptions in this line of work are similar to the assumption we make in Equations 2 and 3 (Doshi-Velez and Konidaris, 2016; Killian et al., 2017; Yao et al., 2018). These methods however do not directly learn Bayes-optimal behaviour; instead, they allow for a longer training period in new environments to infer the latents and train the policy.

Meta-Learning Bayes-Adaptive Policies. It has been shown theoretically (Ortega et al., 2019) and empirically (Mikulik et al., 2020) that meta-trained agents approximate Bayes-optimal agents on the given task distribution. A prominent model-free meta-RL approach to this problem is to use the dynamics of recurrent networks for fast adaptation (RL^2, Wang et al. (2016); Duan et al. (2016)). At every time step, the network receives an auxiliary input consisting of the preceding action and reward. This allows learning within a task to happen online, entirely in the dynamics of the recurrent network, and no gradient adaptation is needed at meta-test time. If we remove the decoder (Fig 2) and the VAE objective (Eq 4), variBAD reduces to this setting, i.e., the main differences are that we use a stochastic latent variable (an inductive bias for representing uncertainty), together with a decoder to reconstruct previous *and future* transitions and rewards (which acts as an auxiliary loss to encode the task in latent space and deduce information about unseen states). While model-free meta-learning methods (RL^2, Wang et al. (2016); Duan et al. (2016)) can meta-learn

approximately Bayes-optimal policies (Ortega et al., 2019; Mikulik et al., 2020), we show empirically that they do not scale as well in terms of performance as variBAD. In addition, variBAD provides explicit belief representations, which can be used in downstream tasks or for auxiliary tasks (Zintgraf et al., 2021).

Closely related to our approach is the work of Humplik et al. (2019). Like variBAD, their algorithm conditions the policy on a meta-learned approximate belief. This approximate belief is learned using privileged information during meta-training such as a task description. In comparison, variBAD meta-learns to represent the belief in an unsupervised way, and does not rely on privileged task information.

Building on the conference version of this work, (Zintgraf et al., 2020), Dorfman et al. (2020) propose a method to meta-learn Bayes-adaptive policies from offline data, and additionally demonstrate good results when using an off-policy variant of variBAD. They use variBAD in combination with the off-policy algorithms DQN (Mnih et al., 2015) (for discrete control) and SAC (Haarnoja et al., 2018) (for continuous control), demonstrating that variBAD can be combined with different RL algorithms.

Other Meta Reinforcement Learning Settings. Another popular approach to meta-RL is to learn an initialisation of the model, such that at test time, only a few gradient steps are necessary to achieve good performance (Finn et al., 2017; Nichol and Schulman, 2018). These methods do not directly account for the fact that the initial policy needs to explore, a problem addressed, a.o., by Stadie et al. (2018) (E-MAML) and Rothfuss et al. (2019) (ProMP). In terms of model complexity, E-MAML and ProMP are relatively lightweight, since they typically consist of a feed-forward policy. RL^2 and variBAD use recurrent modules, which increases model complexity but allows online adaptation. Other methods that perform gradient adaptation at test time are, e.g., Houthooft et al. (2018) who meta-learn a loss function conditioned on the agent's experience that is used at test time to learn a policy (from scratch); and Sung et al. (2017) who learn a meta-critic that can criticise any actor for any task, and is used at test time to train a policy. Compared to variBAD, these methods separate exploration (before gradient adaptation) and exploitation (after gradient adaptation) at test time by design, making them less sample efficient.

Skill and Task Embeddings. Learning (variational) task or skill embeddings for meta or transfer reinforcement learning is used in a variety of approaches. Hausman et al. (2018) use approximate variational inference to learn an embedding space of skills (using a different lower bound than variBAD). At test time a new embedder is learned that interpolates between learned skills. Arnekvist et al. (2019) learn a stochastic embedding of optimal Q-functions for different skills, and condition the policy on (samples of) this embedding. Adaptation at test time is done in latent space. Co-Reyes et al. (2018) learn a latent space of low-level skills that are controlled by a higher-level policy, framed within a hierarchical RL setting. This embedding is learned using a VAE to encode state trajectories and decode states and actions. Zintgraf et al. (2019) learn a deterministic task embedding, trained similarly to MAML (Finn et al., 2017). Similar to variBAD, Zhang et al. (2018) use learned dynamics and reward modules to learn a latent representation which the policy conditions. They show that transferring the (fixed) encoder to new environments helps learning. Perez et al. (2018) learn dynamic models with auxiliary latent variables, and use them for model-predictive control. Lan et al. (2019) learn a task embedding where the encoder is updated at test time using gradient descent, and the policy is fixed. Sæmundsson et al. (2018)

explicitly learn an embedding of the environment model, which is subsequently used for model predictive control (and not, like in variBAD, for exploration). In the field of imitation learning, some approaches embed expert demonstrations to represent the task; e.g., Wang et al. (2017) use variational methods and Duan et al. (2017) learn deterministic embeddings.

VariBAD differs from the above methods mainly in what the embedding represents (i.e., task uncertainty) and how it is used: the policy conditions on the posterior *distribution* over MDPs, allowing it to reason about task uncertainty and trade off exploration and exploitation online. Our objective in Equation 5 explicitly optimises for Bayes-optimal behaviour. Unlike some of the above methods, we do not use the model at test time, but model-based planning is a natural extension for future work.

Variational Inference and Meta-Learning. A main difference of variBAD to many existing Bayesian RL methods is that we meta-learn the inference procedure, i.e., how to do a posterior update. Apart from (RL) methods mentioned, related work in this direction can be found, a.o., in Garnelo et al. (2018); Gordon et al. (2019); Choi et al. (2019). In contrast to these supervised settings, variBAD's inference procedure is tailored to Bayes-optimal RL, a sequential setting where the objective of inference is to model beliefs over MDPs.

Partially Observable Markov Decision Processes (POMDPs). Several deep learning approaches to model-free reinforcement learning (Igl et al., 2019) and model learning for planning (Tschiatschek et al., 2018) in partially observable Markov decision processes have recently been proposed and use approximate variational inference methods. VariBAD by contrast focuses on BAMDPs (Martin, 1967; Duff and Barto, 2002; Ghavamzadeh et al., 2015), a special case of POMDPs where the transition and reward functions constitute the hidden state and the agent must maintain a belief over them. While in general the hidden state in a POMDP can change at each time-step, in a BAMDP the underlying task, and therefore the hidden state, is fixed per task. We exploit this property by learning an embedding that is *fixed* over time, unlike approaches like the one by Igl et al. (2019) which use filtering to track the changing hidden state. While we use the power of deep approximate variational inference, other approaches for BAMDPs often use more accurate but less scalable methods, e.g., Lee et al. (2019) discretise the latent distribution and use Bayesian filtering for the posterior update.

5. Empirical Evaluation

In this section we first investigate the properties of variBAD on the didactic gridworld domain from Figure 1. These results show that variBAD performs *structured* and *online* exploration as it infers the task. We then evaluate variBAD on several more challenging domains: a sparse 2D navigation task, a range of tasks from the MuJoCo benchmark, and the Meta-World ML1 benchmark. On these, variBAD achieves state of the art performance. In Section 6, we perform ablation studies to motivate our design choices, and test how robust variBAD is to the size of the latent space.

The two main baselines we consider are RL^2 (Duan et al., 2016; Wang et al., 2016) and PEARL (Rakelly et al., 2019). RL^2 can be seen as a model-free version of variBAD. In principle, it can learn to act approximately Bayes-optimal by performing task inference in the recurrent state of the RNN. RL^2 only consists of an encoder and a policy, and trains both end-to-end using an RL loss only. PEARL is akin to posterior sampling, which is

a sub-optimal exploration strategy in theory compared to Bayes-optimal exploration (see Figure 1), but has the advantage that computing the optimal policy for a single MDP (sample from the approximate posterior) is more tractable than doing so in a BAMDP.

Experimental details, hyperparameters, as well as additional results, can be found in the appendix. The source code is available at `https://github.com/lmzintgraf/varibad` .

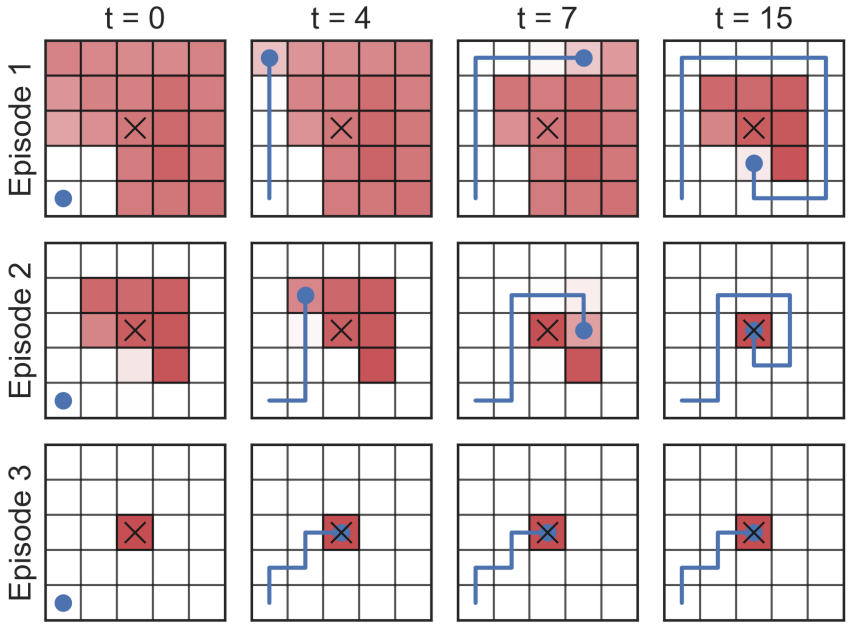
5.1 GridWorld

To gain insight into variBAD's properties, we start with a didactic gridworld environment. The task is to reach a goal in a 5×5 gridworld (see the visualisation in Figure 1a). The goal can be anywhere except around the starting cell, which is at the bottom left, and is selected uniformly at random. The goal is unobserved by the agent, inducing task uncertainty and necessitating exploration. Actions are: *up, right, down, left, stay*, and are executed deterministically. The horizon within the MDP is $H = 15$, and we set the horizon in the BAMDP to $H^+ = 4 \times H = 60$, i.e., we train our agent to maximise performance for 4 MDP episodes. After 15 steps the agent is reset to its starting position (but the goal position stays the same). The agent gets a sparse reward signal: -0.1 on non-goal cells, and $+1$ on the goal cell (repeatedly if the agent stays on the goal). We use a latent dimensionality of 5 (see Section 6.3 for a study on how different latent sizes affect performance). The Bayes-optimal strategy is to explore until the goal is found (see Figure 1b), and stay at the goal or return to it when reset to the initial position.

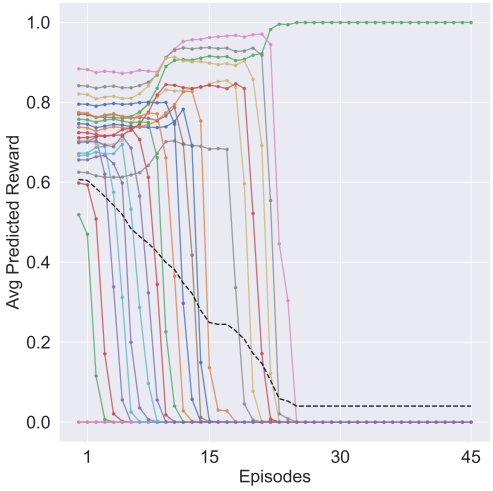
Figure 3 illustrates how variBAD behaves at test time with deterministic actions (i.e., all exploration is done by the deterministic policy, not via sampling). In Figure 3a we see how the agent interacts with the environment over the course of three episodes (with a fixed goal), with the red background visualising the approximate posterior belief, using the learned reward function. VariBAD learns the correct prior and adjusts its belief correctly over time: It predicts no reward for cells it has visited, and high expected rewards for unvisited cells. It explores the remaining cells until it finds the goal, at which point its posterior collapses to the correct task. As the agent gathers more data, more and more cells are excluded ($p(rew = 1) = 0$, white cells), until eventually the agent finds the goal.

Figure 3b show the reward predictions: each line represents a grid cell and its value the probability of receiving a reward at that cell. As the agent gathers more data, more and more cells are excluded ($p(rew = 1) = 0$), until eventually the agent finds the goal. At this point, the predictions correspond to the true reward function. In Figure 3c we visualise the 5-dimensional latent space (mean and variance in separate sub-plots). We see that once the agent finds the goal, the posterior concentrates: the variance drops close to zero, and the mean converges to a fixed value.

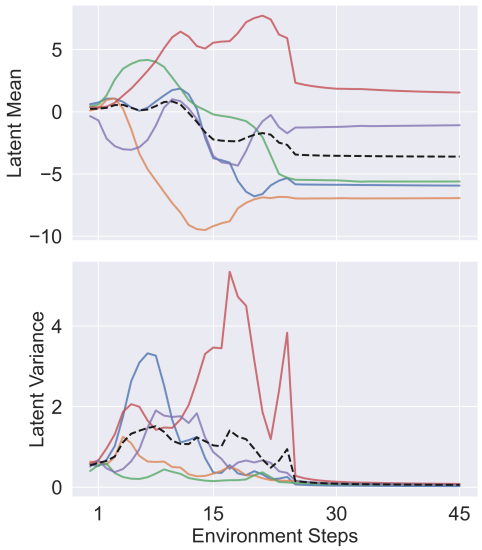
As we saw in Figure 1e, the behaviour of variBAD closely matches that of the Bayes-optimal policy, which optimally trades off exploration and exploitation in an unknown environment, and outperforms posterior sampling. Our results on this gridworld indicate that variBAD is an effective way to approximate Bayes-optimal control, and has the additional benefit of giving insight into the task belief of the policy.



(a) Example Rollout

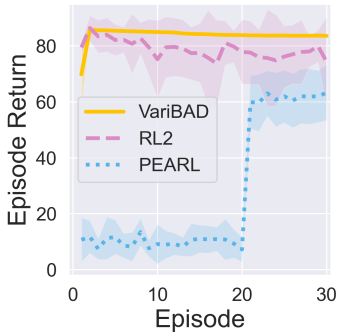


(b) Reward Predictions

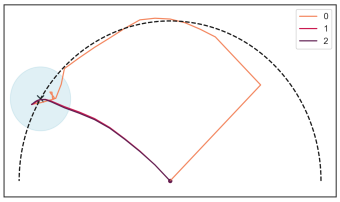


(c) Latent Space

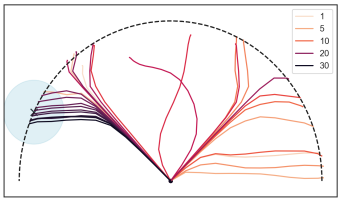
Figure 3: Behaviour of variBAD in the gridworld environment. (a) Hand-picked but representative example test rollout. The red background indicates the posterior probability of receiving a reward at that cell. (b) Probability of receiving a reward for each cell, as predicted by the decoder, over the course of interacting with the environment (average in black, goal state in green). (c) Visualisation of the latent space; each line is one latent dimension, the black line is the average.



(a) Meta-Test Performance



(b) VariBAD example rollout



(c) PEARL example rollout

Figure 4: Meta-test performance on the Sparse 2D Navigation environment. (a): Performance at meta-test time (averaged over the task distribution). (b) and (c): Example rollouts of meta-trained variBAD (b) and PEARL (c) agents. PEARL does not optimize for optimal exploration, and thus it requires many more episodes to find the goal. On the other hand, variBAD optimises for optimal exploration, efficiently covering possible goal locations, and is able to quickly find the goal.

5.2 Sparse 2D Navigation

We evaluate on a Point Robot 2D navigation task used by Gupta et al. (2018); Rakelly et al. (2019); Humplik et al. (2019), to further illustrate how variBAD performs online adaptation. The agent must navigate to an unknown goal sampled along the border of a semicircle of radius 1.0, and receives a reward relative to its proximity to the goal when it is within a goal radius of 0.2. Since this is a sparse reward environment, the Bayes-optimal exploration strategy includes walking along the semi-circle until the goal is found.

Figure 4a shows the average performance of PEARL, RL^2, and variBAD at test time, when rolling out the agent for 30 episodes in a single task. VariBAD adapts much faster to the task compared to PEARL. RL^2 also adapts rapidly but is less stable compared to variBAD when rolled out for a much longer time than during training: both variBAD and RL^2 were trained to perform well over three consecutive episodes.

To shed light on the performance difference between variBAD and PEARL, Figures 4b and 4c visualise representative example rollouts for meta-trained variBAD and PEARL agents. We picked examples where the target goals are at the end of the semi-circle, which we found are most difficult for the agent. PEARL performs posterior sampling for exploration which means it is restricted to a fixed hypothesised goal position during each rollout. On the other hand, variBAD (Figure 4b) strategically explores the space within an episode to find the goal, which is more efficient. Once the goal is found, both variBAD and PEARL are able to quickly return to it.

The two toy experiments, GridWorld and PointRobot, illustrate how variBAD makes decisions: it adapts to the task *online* while updating the approximate belief, which allows it to rapidly adapt to new tasks. In the following sections, we test variBAD on more challenging meta-RL benchmarks, MuJoCo and Meta-World ML1.

5.3 MuJoCo Continuous Control Meta-Learning Tasks

We show that variBAD can scale to more complex meta-learning settings by employing it on MuJoCo (Todorov et al., 2012) locomotion tasks commonly used in the meta-RL literature.[4] We consider the Ant-Dir, HalfCheetahDir, and Humanoid environments, where the agent has to run either forwards or backwards (i.e., there are only two tasks); the HalfCheetahVel environment where the agent has to run at different velocities; the Ant-Goal environment where the agent has to navigate to an initially unknown goal position; and the Walker environment where the system parameters are randomised. The rewards in these environments are dense, so that in principle the agent only needs a few exploratory actions to infer the task by observing the rewards it receives.

VariBAD and RL^2 were trained to maximise performance within two episodes (mainly so that we can roll them out for multiple episodes at test time; we are primarily interested in their adaptation behaviour within the first episode). PEARL (Rakelly et al., 2019), was trained using the open sourced code[5], to maximise performance after 3-5 episodes, depending on the environment. E-MAML (Stadie et al., 2018) and ProMP (Rothfuss et al., 2019) were trained using the open sourced code by Rothfuss et al. (2019)[6], to maximise performance after 1 gradient step on 10-20 rollouts. To get an approximate upper bound on performance, we train a multi-task agent which conditions on a task descriptor, and an ensemble of expert agents (one per task) whose performances are averaged, using PPO.

Figure 5 shows the average performance at test time across different tasks. While we show the return of the agent during the first five rollouts, we emphasise that our primary interest lies in the agent's performance during the first episode, *while learning* about the environment, which tells us how well the agent can trade off exploration and exploitation.

Only variBAD and RL^2 are able to adapt to the task at hand within a single episode. VariBAD outperforms RL^2 in all environments except HalfCheetahVel where they are on par, and is close to the multi-task agent's performance in several environments. We generally found that learning with RL^2 is slower and less stable (see learning curves and runtime comparisons in Appendix C). We hypothesise that this is because the reinforcement learning loss is backpropagated through an RNN. In variBAD on the other hand, we train the encoder RNN with a supervised loss only. Even though the first rollout includes exploratory steps, this matches the optimal multitask policy (which is conditioned on the true task description) up to a small margin.

The methods PEARL (Rakelly et al., 2019), E-MAML (Stadie et al., 2018), and ProMP (Rothfuss et al., 2019) are not designed to maximise reward during a single rollout, and perform poorly in the first episode. They all require substantially more environment interactions in each new task to achieve good performance. PEARL, which is akin to posterior sampling, only starts performing well starting from the third episode. We evaluated E-MAML and ProMP by performing gradient steps after every episode; however, they are typically updated after collecting data for 20 episodes.

4. The MuJoCo environments are taken from `https://github.com/katerakelly/oyster` .
5. The implementation which we used for our PEARL experiments was published by Rakelly et al. (2019) and can be found at `https://github.com/katerakelly/oyster` .
6. The implementation which we used for our E-MAML and ProMP experiments was published by Rothfuss et al. (2019) and can be found at `https://github.com/jonasrothfuss/ProMP` .

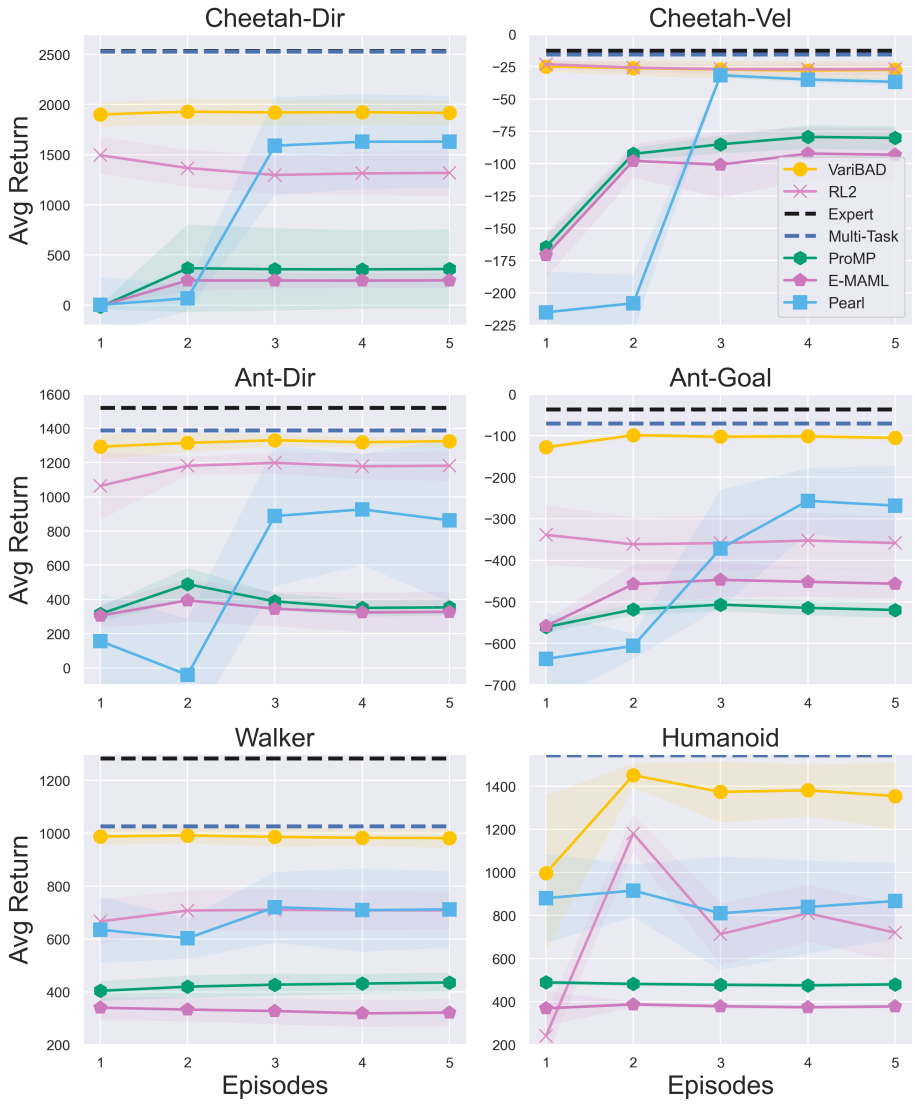


Figure 5: Average test performance on six different MuJoCo environments, trained separately with 10 seeds per MuJoCo environment per method. The meta-trained policies are rolled out for 5 episodes to show how they adapt to the task. Values shown are averages across tasks (95% confidence intervals shaded). Being an online adaptation method, variBAD adapts within the first episode. It outperforms other methods, even when these are given longer than one episode to adapt, and even though the first episode includes exploratory actions. RL^2 is also an online adaptation method and can adapt to the task within the first episode. On most environments, variBAD outperforms RL^2 (Wang et al., 2016; Duan et al., 2016) significantly. The other methods, PEARL (Rakelly et al., 2019), E-MAML (Stadie et al., 2018), and ProMP (Rothfuss et al., 2019), need at least one episode to adapt by design.

Method	Episode	Reach	Push	Pick-Place
MAML*	10	48	74	12
PEARL*	10	38	71	28
RL2*	10	45	87	24
variBAD	**1**	**100**	**100**	29 (6/20 seeds)
variBAD	2	100	100	29

Table 1: Meta-test success rates on the ML1 Meta-World benchmark. *Results taken from Yu et al. (2019). VariBAD was trained to maximise expected online return within 2 episodes (20 random seeds per setting). The first (few) episodes often *includes exploratory actions*, yet variBAD has higher success rate in episode 1 than existing methods. For the Pick-Place environment, in brackets we report the number of seeds that learned something.

5.4 Meta-World

Finally, we evaluate variBAD on the challenging Meta-World ML1 benchmark (Yu et al., 2019), which has emerged as a key challenge for the meta-learning community. In the Meta-World environment, a robot arm has to perform different tasks like pushing objects to an (initially unknown) target location. There are three variants of the ML1 benchmark: Reach (where the robot has to reach different initially unknown goal positions), Push (where the robot arm has to push objects to initially unknown goal positions), and Pick-Place (where the robot arm has to pick up an object, and place it near a target goal in 3D space).

Table 1 shows the results for variBAD and several baselines on the ML1 benchmark. VariBAD achieves state of the art results. On Reach and Push, variBAD outperforms the previous state of the art results by a significant margin, and is the first to fully solve these tasks. On the harder task Pick & Place, variBAD performs on par with PEARL.

Though the benchmark allows for up to 10 episodes for adaptation, we train variBAD to optimise expected online return within two episodes. As these results show, variBAD can adapt rapidly even within the first episode, which includes exploratory actions. On the Pick-Place task, variBAD either learns to solve the task (6 seeds), or it does not meta-learn at all (14 seeds). We hypothesise that variBAD (and the other methods) face a meta-exploration challenge: the agent does not explore enough during meta-training, which prevents it from learning how to adapt to the task. We found that increasing the episode horizon from 150 to 200 helps mitigate this problem to some extend (results not shown). We address the challenge of meta-exploration more generally in follow-up work (Zintgraf et al., 2021).

6. Empirical Analysis

In this section, we first examine the impact of different VAE loss formulations on the performance and approximate beliefs to shed light on our design choices. We further analyse how variBAD's performance is impacted by different choices for the latent dimension.

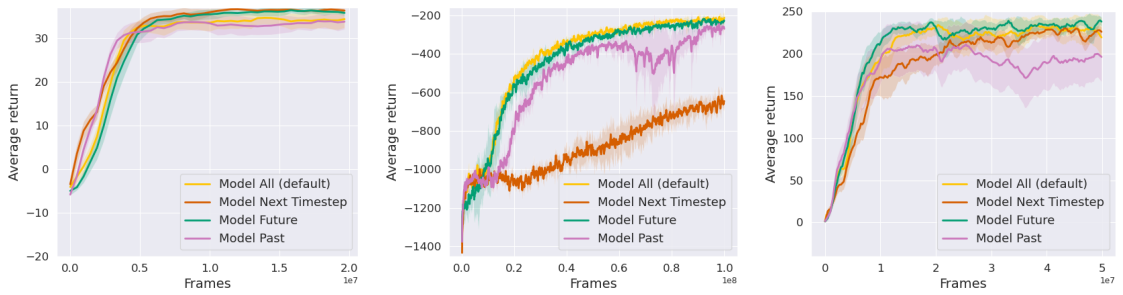


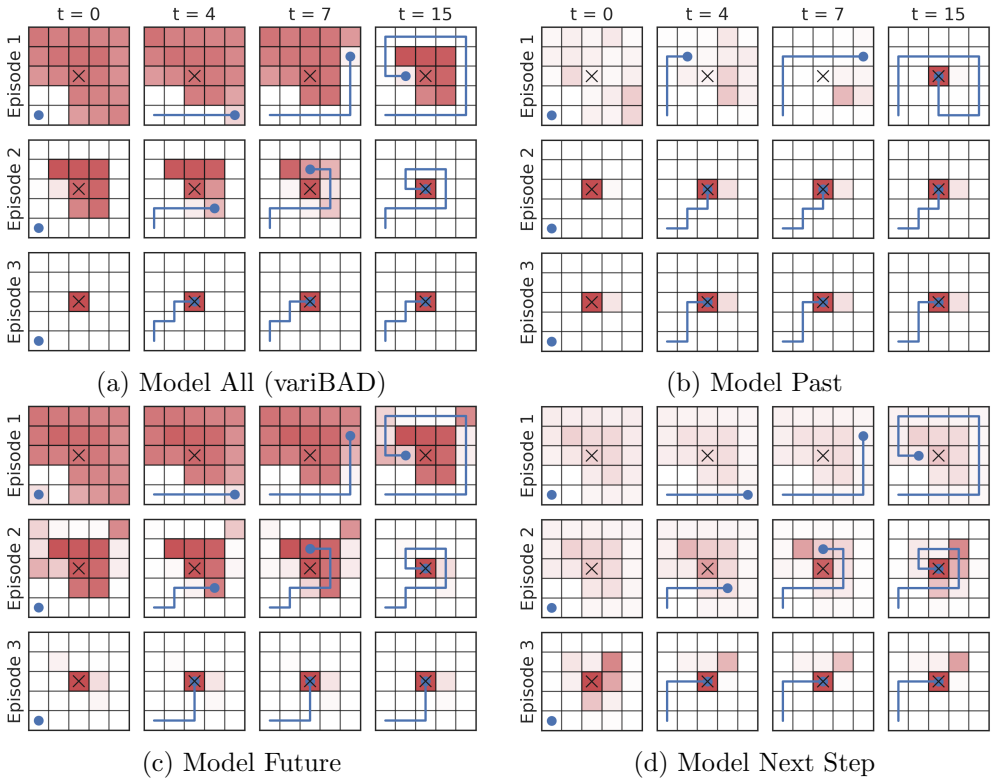
Figure 6: Performance of VAE decoding targets for (from left): GridWorld, AntGoal, and Pointrobot (15/5/20 seeds). VariBAD's default settings (modelling the past and future) perform well, and behave expected in terms of predicted beliefs (Fig 7a). Other choices either underperform (modelling only the past or only the next step), or have undesired effects on the beliefs (modelling only the future, Fig 7c).

6.1 Modelling Horizon

When formulating our objective in Equation 4, we argue that at any time t in its trajectory, an agent should be able to model entire trajectories. In Equation 6 we see that this amounts to reconstructing transitions and rewards from the past ($\leq t$) and the future ($> t$). The intuition is that the VAE has to represent the task information in its belief, which encompasses a sufficient statistic about transitions the agent has already observed, and a belief about transitions it can observe in the future. To analyse this choice empirically, we test how different "modelling horizons" affect performance and the learned beliefs: modelling only the past ($\leq t$), only the future ($> t$), or only one step into the future ($t+1$).

Figure 6 shows the resulting learning curves for the GridWorld, Mujoco AntGoal, and PointRobot tasks. The traditional VAE approach of reconstructing only the embedded part of the trajectory (past) tends to produce latent codes not sufficiently informative to predict future transitions and therefore leads to suboptimal performance. Similarly, targeting only the subsequent transition (next) does not encode sufficient information to reliably inform the policy. While decoding only the entire remainder of the observed training trajectories (future) performs well, we find that this has an undesired effect on the learned beliefs, discussed below.

Figure 7 visualises the approximate belief for different decoding targets in the Gridworld environment. We do so by plotting the reward predictions of the decoder from the VAE latent for each cell in the grid. Decoding only the past or only the next step leads to embeddings that predict no or only spurious rewards until the goal has been found (Figures 7b and 7d). Only decoding the future enables learning about the rewards prior to actually encountering them, but leads to spurious predictions for visited states (Figure 7c): predictions of non-zero rewards at visited non-goal states are not penalised, as these are unlikely to be revisited. These artefacts are cleared up by decoding full trajectories (Figure 7a), which is the default setting we chose for the variBAD objective.



(a) Model All (variBAD)

(b) Model Past

(c) Model Future

(d) Model Next Step

Figure 7: Belief visualisation in the Gridworld environment for different decoding targets. Compared to the default variBAD objective which models both the past and the future (a), the approximate belief is less accurate when decoding only the past (b), only the future (c), or only one step into the future (d). In (b) the agent only learns to decode the past, and assigns low probability to all cells (seen and unseen), when it has not found the goal yet. This strategy minimises the loss: for seen cells it correctly predicts that they do not contain the goal; for unseen cells it predicts the same (even though one of them actually contains the goal), which however is not penalised through the loss. In (c) the agent only learns to decode the future, and assigns high probability to all unseen cells, and also to some seen cells, when it has not found the goal yet. Again, this minimises the loss: for unseen cells it correctly predicts that they could contain the goal; for seen cells the loss does not incentivise it to predict the correct thing and the predictions are noisy. We believe some predictions are zero because there is a chance of the agent re-visiting states. In (d) the agent only learns to predict the immediate-next step, and assigns non-zero (but low) probability to these, when it has not found the goal yet. Only when decoding both the entire past and future (default setting, (a)), does the agent correctly predict that seen cells do not contain the goal, and unseen cells could all contain the goal. All methods (a)-(d) learn to correctly predict the goal position with high certainty once they have seen it once.

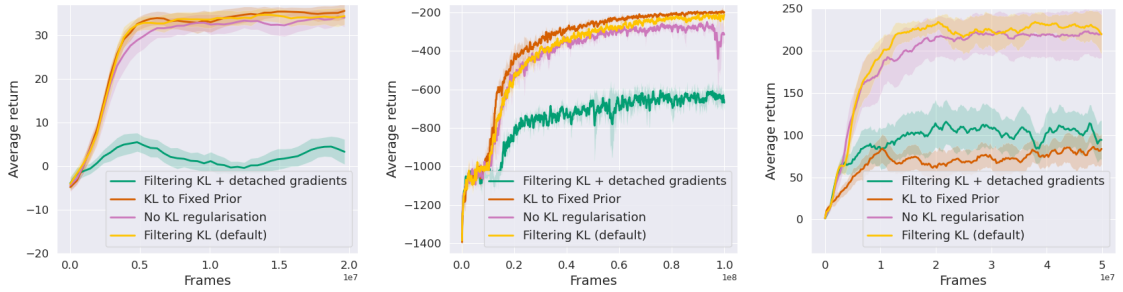


Figure 8: Comparison of VAE regularisation penalties

Figure 9: Performance of different KL regularisations in the VAE. Domains from left to right are GridWorld (15 seed average), Ant-Goal (5 seed average) and Pointrobot (20 seed average). Using the filtering KL objective performs well consistently across these environments. Using the KL to a fixed prior works well except in the PointRobot environment. We hypothesise this is due to the sparsity of the environment, and the fact that "excluding" certain regions without rewards becomes more challenging when the prior is fixed.

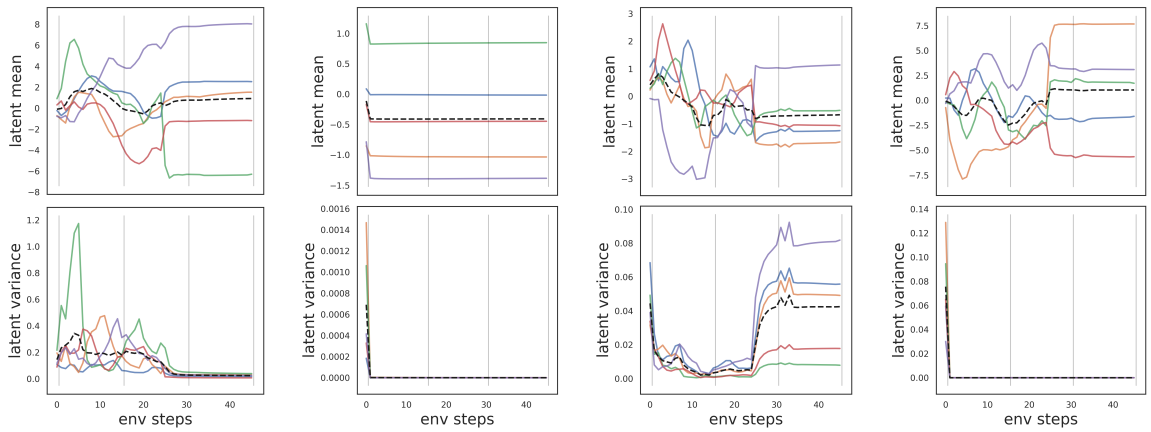
6.2 KL Regularisation

The second component of the ELBO, the KL term between the posterior and prior, acts as a regulariser and prior for the latent codes. In the variBAD objective, the prior is set to the previous approximate posterior, i.e., the KL term is defined as $KL(q(m|\tau_{:t})||q(m|\tau_{t-1}))$, which is akin to Bayesian filtering. The gradient is taken w.r.t. both terms in the KL.

In this subsection, we empirically motivate this choice, by comparing to using a fixed standard Normal distribution as a prior, $\mathcal{N}(0, I)$, across all times steps ("fixed prior"), detaching the gradient of the approximate previous posterior, or not doing KL regularisation.

The learning curves are shown in Figure 8 for the GridWorld, AntGoal, and Pointrobot environment, and visualisations of the learned approximate beliefs in the GridWorld are shown in Figure 10. Using a fixed prior is not sufficiently flexible to allow the learning of good latent codes in all cases: the performance is significantly worse in PointRobot (Fig 8, right), and when rolling out the meta-learned policy the variance increases sharply after the agent finds the goal in the GridWorld (Fig 10c). This is undesirable—the variance should collapse once the agent is certain about the task. By forcing the VAE to adhere to a fixed prior we artificially increase uncertainty as the embedding is encouraged to regress towards the prior even once the goal has been identified.

When training with the default variBAD objective we get reasonable variance estimates that decay with progressing exploration and collapse when the goal is found (Fig 10a). Using no KL regularisation at all slightly reduces performance (Fig 8) and a total collapse in variance (Fig 10c). A lack of regularisation allows the VAE to encode observed trajectories as point masses with no variance, indicating it overfits by producing a unique code for each of them. Detaching the gradient on the previous approximate posterior significantly harms performance in all cases (Fig 8) and yields an inflexible latent space (Fig 10b).



(a) Filtering KL (b) Detached Gradient (c) Fixed Prior (d) No regularisation

Figure 10: Visualisation of latent estimates for different decoding targets, for a representative GridWorld rollout. The agent finds the goal after $\sim$ 25 steps. When using a filtering KL (variBAD's default setting), the variance collapses once the goal is found (a). When using a fixed Gaussian prior (c), the posterior behaves unexpectedly: the variance *increases* once the goal is found.

6.3 Belief Dimensionality

Finally, we examine how the size of the latent belief impacts performance in Figure 11, for the GridWorld and Ant-Goal tasks. As expected, if the latent dimensionality is too small, then not all relevant information can be retained and the policy is unable to adapt to different tasks. This only happens when using a dimensionality of size 1 or 2, and any choice larger than this leads to decent performance. Interestingly, very large parameterisations (1000 for GridWorld and 300 for AntGoal which was the maximum we could fit into the memory of a single GPU) have a comparatively minute impact despite artificially increasing the size of the state space on which the policy acts. In practice this means that as long as we do not underparameterise the latent dimension, we achieve good performance.

7. Conclusion & Future Work

We presented variBAD, a novel deep RL method to learn approximately Bayes-optimal behaviour, which uses meta-learning to exploit knowledge obtained in related tasks and perform approximate inference in unknown environments. In a didactic gridworld environment, our agent closely matches Bayes-optimal behaviour, and in more challenging MuJoCo tasks, variBAD outperforms existing methods in terms of achieved reward during a single episode. On the recently proposed Meta-World ML1 benchmark, variBAD outperforms existing methods by a large margin and fully solves two out of the three benchmark tasks for the first time. In summary, we believe variBAD opens a path to tractable approximate Bayes-optimal exploration for deep reinforcement learning.

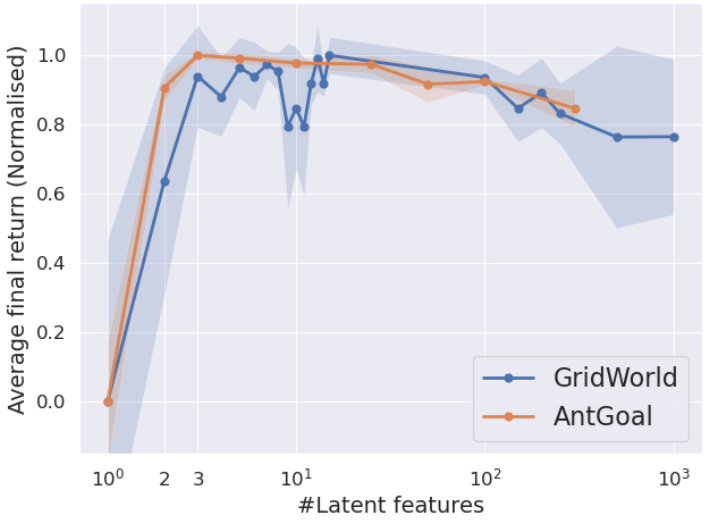


Figure 11: Normalised performance for different number of latent features, for the Gridworld and Ant environment. As long as the latent is not underparameterised, variBAD achieves good performance. Only when vastly overparameterising the latent we see a small performance decrease.

There are several interesting directions of future work based on variBAD. For example, we currently do not use the decoder at test time, but it could be used for online planning, or to get a sense for how wrong the predictions are (which might indicate we are out of distribution, and further training is necessary). Another exciting direction for future research is considering settings where the training and test distribution of environments are different. Generalising to out-of-distribution tasks poses additional challenges and in particular for variBAD two problems are likely to arise: the inference procedure will be wrong (the prior and/or posterior update) and the policy will not be able to interpret a changed posterior. In this case, further training of both the encoder/decoder might be necessary, together with updates to the policy and/or explicit planning.

Acknowledgments

We thank Anuj Mahajan, Joost van Amersfoort, Andrei Rusu, Dushyant Rao, and everyone at WhiRL for useful discussions and feedback. Luisa Zintgraf is supported by the 2017 Microsoft Research PhD Scholarship Program, and the 2020 Microsoft Research EMEA PhD Award. Sebastian Schulze is supported by Dyson. Maximilian Igl and Cong Lu are supported by the UK EPSRC CDT in Autonomous Intelligent Machines and Systems. This work was supported by a generous equipment grant and a donated DGX-1 from NVIDIA, and enabled in part by computing resources provided by Compute Canada. This project has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme (grant agreement number 637713).

Appendix A. Full ELBO derivation

Equation 5 can be derived as follows.

$$\begin{aligned}
\mathbb{E}_{\rho(M,\tau_{:H})}\left[\log p_\theta(\tau_{:H})\right] &= \mathbb{E}_\rho\left[\log \int p_\theta(\tau_{:H},m)\frac{q_\phi(m|\tau_{:t})}{q_\phi(m|\tau_{:t})}dm\right] \\
&= \mathbb{E}_\rho\left[\log \mathbb{E}_{q_\phi(m|\tau_{:t})}\left[\frac{p_\theta(\tau_{:H},m)}{q_\phi(m|\tau_{:t})}\right]\right] \\
&\geq \mathbb{E}_{\rho,\ q_\phi(m|\tau_{:t})}\left[\log \frac{p_\theta(\tau_{:H},m)}{q_\phi(m|\tau_{:t})}\right] \\
&= \mathbb{E}_{\rho,\ q_\phi(m|\tau_{:t})}\left[\log p_\theta(\tau_{:H}|m) + \log p_\theta(m) - \log q_\phi(m|\tau_{:t})\right] \\
&= \mathbb{E}_\rho\left[\mathbb{E}_{q_\phi(m|\tau_{:t})}\left[\log p_\theta(\tau_{:H}|m)\right] - KL(q_\phi(m|\tau_{:t})||p_\theta(m))\right] \\
&= ELBO_t.
\end{aligned}$$

Appendix B. Experiments: GridWorld

Here we provide additional remarks and figures for the GridWorld results from Section 5.1.

B.1 Additional Remarks

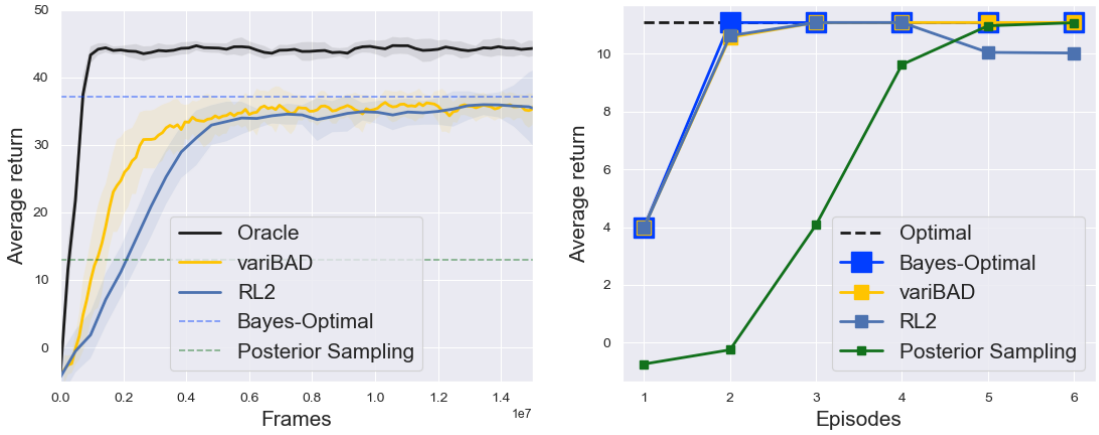
Figure 3c visualises how the latent space changes as the agent interacts with the environment. As we can see, the value of the latent dimensions starts around mean 0 and variance 1, which is the prior we chose for the beginning of an episode. Given that the variance increases for a little bit before the agent finds the goal, this prior might not be optimal. A natural extension of variBAD is therefore to also learn the prior to match the task at hand.

B.2 Comparison to RL2

Figure 12a shows the learning curves for variBAD and RL^2, in comparison to a multitask policy (which has access to the goal position). We trained these policies on a horizon of $H^+ = 4 \times H = 60$, i.e., on a BAMDP in which the agent has to maximise online return within four episodes. We indicate the values of a hard-coded Bayes-optimal policy, and a hard-coded posterior sampling policy using dashed lines.

Figure 12b shows the end-performance of variBAD and RL^2, compared to the hard-coded optimal policy (which has access to the goal position), Bayes-optimal policy, and posterior sampling policy. VariBAD and RL^2 both closely approximate the Bayes-optimal solution. By inspecting the individual runs, we found that variBAD learned the Bayes-optimal solution for 4 out of 20 seeds, RL^2 zero times. Both otherwise find solutions that are very close to Bayes-optimal, with the difference that during the second rollout, the cells left to search are not all on the shortest path from the starting point.

VariBAD and RL^2 were trained on 4, and evaluated on 6 episodes. After the fourth rollout, we do *not* fix the latent / hidden state, but continue rolling out the policy as before. We find that the performance of RL^2 drops again after the fourth episode: this is likely due to instabilities in the 128-dimensional hidden state. VariBAD's latent representation, the approximate task posterior, is concentrated and does not change with more data.



(a) Learning curves.

(b) Average return per test episode.

Figure 12: Results for the GridWorld toy environment. Results are averages over 20 seeds (with 95% confidence intervals for the learning curve).

Appendix C. Experiments: MuJoCo

In this section we provide the learning curves (C.1) for the MuJoCo environments from Section 5.3. We also provide additional analyses of the learned agent behaviour (C.2 and how the latent space behaves at test time C.3). C.4 provides a runtime comparison, and C.5 an ablation on backpropagating the RL loss through the VAE encoder.

C.1 Learning Curves

Figure 13 shows the learning curves for the MuJoCo environments for all approaches. The multitask and expert policies were trained using PPO. PEARL (Rakelly et al., 2019) was trained using the reference implementation provided by the authors. The environments we used are also taken from this implementation. E-MAML (Stadie et al., 2018) and ProMP (Rothfuss et al., 2019) were trained using the reference implementation provided by Rothfuss et al. (2019).

As we can see, PEARL is much more sample efficient in terms of number of frames than the other methods (Fig 13), which is because it is an off-policy method. On-policy vs off-policy training is an orthogonal issue to our contribution, but an extension of variBAD to off-policy methods has been done in Dorfman et al. (2020). Doing posterior sampling using off-policy methods also requires PEARL to use a different encoder (to maintain order invariance of the sampled trajectories) which is non-recurrent (and hence faster to train, see next section).

For all MuJoCo environments, we trained variBAD with a reward decoder only (even for Walker, where the dynamics change, we found that this has superior performance).

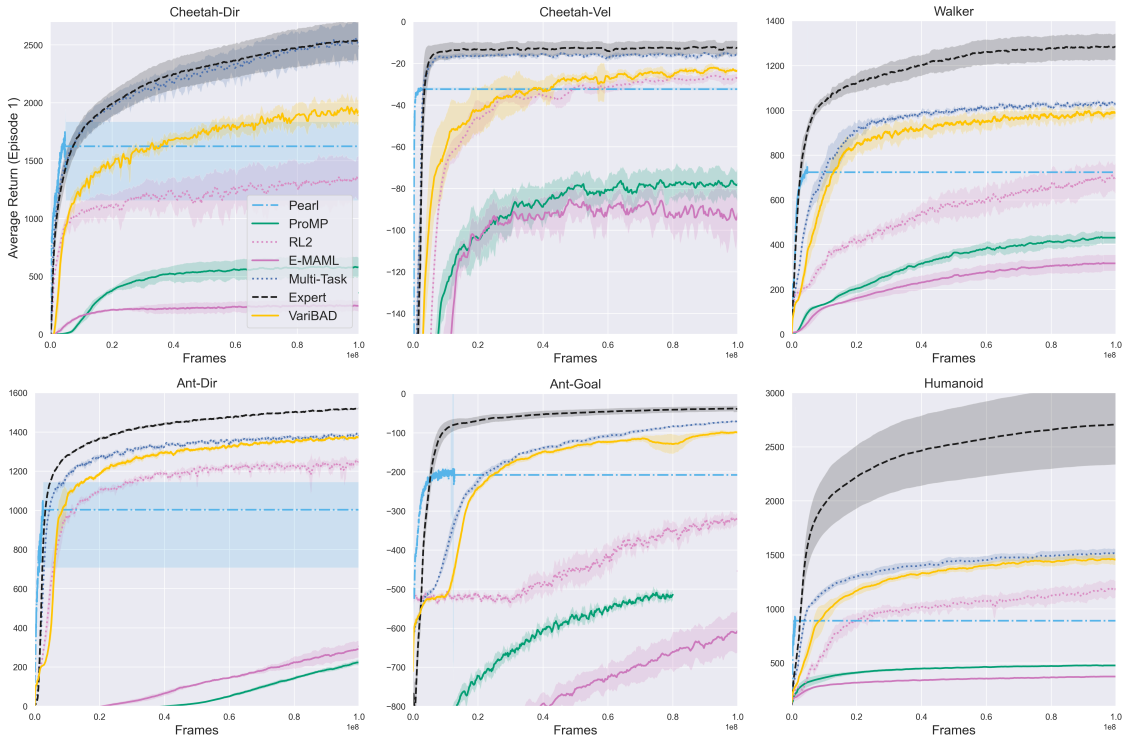
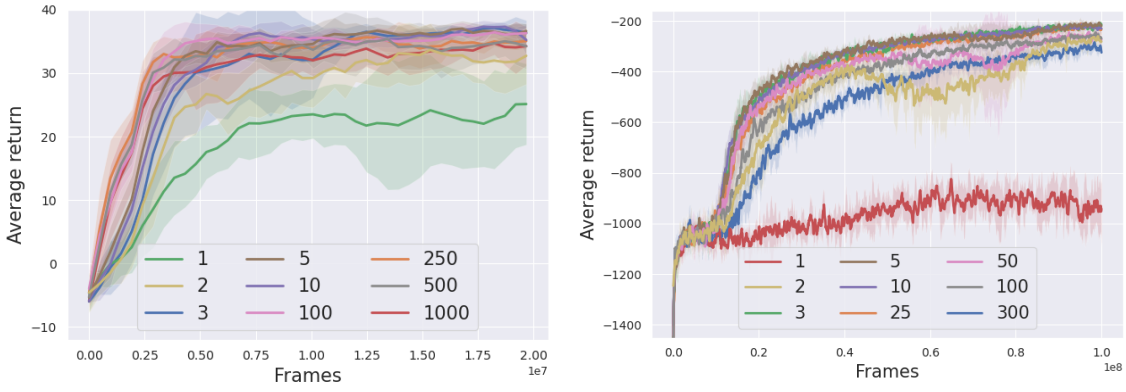


Figure 13: Learning curves for the MuJoCo results presented in Section 5.3. The plots show the performance at the N-th rollout. For variBAD and RL^2, $N = 2$. For PEARL, $N = 10$. For ProMP and E-MAML, $N = 30 - 60$ (3 gradient steps on rollouts of length 10-20 depending on the environment).



(a) GridWorld.

(b) AntGoal.

Figure 14: Learning curves for different VAE latent dimensions, for the GridWorld environment (a) and the MuJoCo AntGoal environment (b).

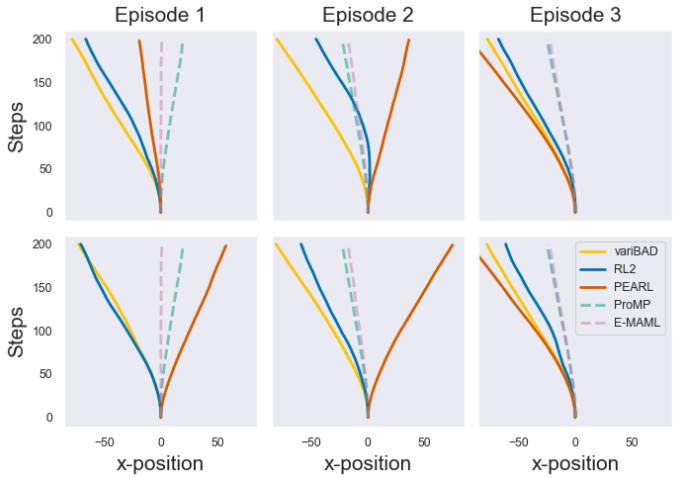


Figure 15: Test time behaviour for the task "walk left" in HalfCheetahDir. The x-axis reflects the agent's position; the y-axis the environment steps (to be read from bottom to top). Rows are separate examples, columns the number of rollouts.

C.2 CheetahDir Test Time Behaviour

To get a sense for where these differences between the different approaches might stem from, consider Figure 15 which shows example behaviour of the policies during the first three rollouts in the HalfCheetahDir environment, for the task "go left". VariBAD and RL^2 adapt to the task online, whereas PEARL acts according to the current sample, which in the first two rollouts can mean walking in the wrong direction. For a visualisation of the variBAD latent space at test time for this environment see Figure 16.

C.3 Latent Space Visualisation

A nice feature of variBAD is that it can give us insight into the uncertainty of the agent about what task it is in. Figure 16 shows the latent space for the HalfCheetahDir tasks "go right" (top row) and "go left" (bottom row). We observe that the latent mean and log-variance adapt rapidly, within just a few environment steps (left and middle figures). This is also how fast the agent adapts to the current task (right figures). As expected, the variance decreases over time as the agent gets more certain. It is interesting to note that the values of the latent dimensions swap signs between the two tasks.

Visualising the belief in the reward/state space directly, as we have done in the GridWorld example, is more difficult for MuJoCo tasks, since we now have continuous states and actions. What we could do instead, is to additionally train a model that predicts a ground-truth task description (separate from the main objective and just for further analysis, since we do not want to use this privileged information for meta-training). This would give us a more direct sense of what task it thinks it is in.

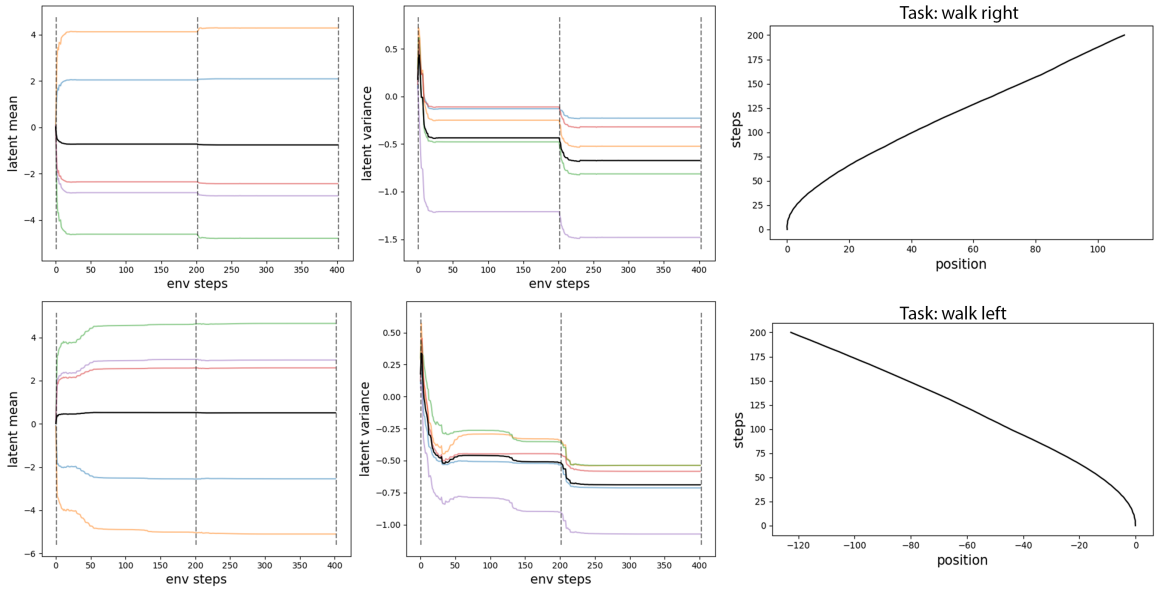


Figure 16: Visualisation of the latent space at meta-test time, for the HalfCheetahDir environment and the tasks "go right" (top) and the task "go left" (bottom). Left: value of the posterior mean during a single rollout (200 environment steps). The black line is the average value. Middle: value of the posterior log-variance during a single rollout. Right: Behaviour of the policy during a single rollout. The x-axis show the position of the Cheetah, and the y-axis the step (should be read from bottom to top).

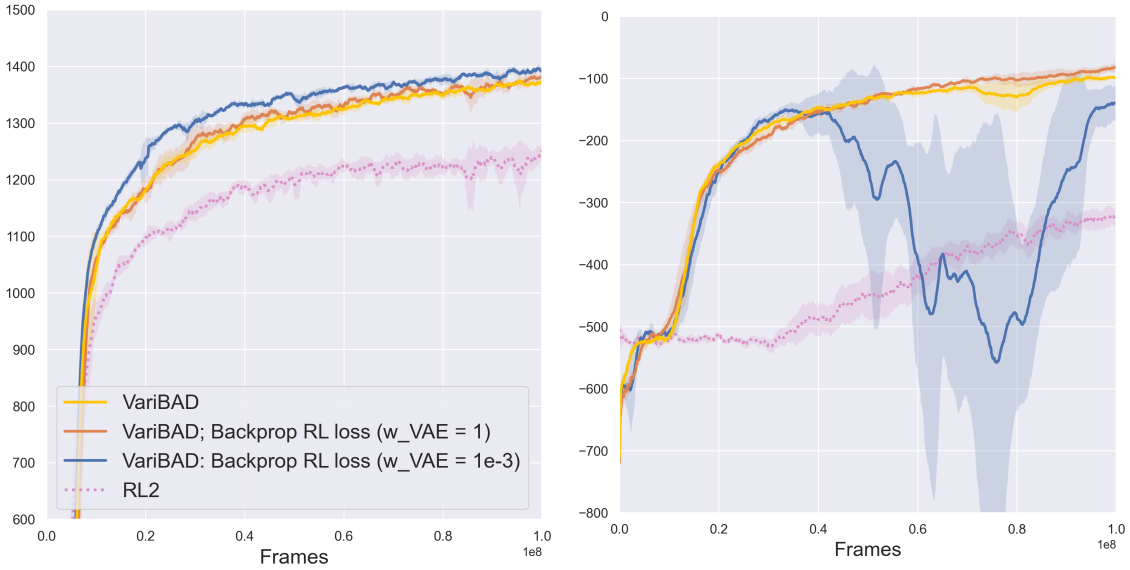
C.4 Runtime Comparison

The following are rough estimates of average run-times for the HalfCheetahDir environment (from what we have experienced; we often ran multiple experiments per machine, so some of these might be overestimated and should be mostly understood as giving a relative sense of ordering).

- ProMP, E-MAML: 5-8 hours
- variBAD: 48 hours
- RL^2: 60 hours
- PEARL: 24 hours

E-MAML and ProMP have the advantage that they do not have a recurrent part such as variBAD or RL^2. Forward and backward passes through recurrent networks can be slow, especially with large horizons.

Even though both variBAD and RL^2 use recurrent modules, we observed that variBAD is faster than RL^2 when training the policy with PPO. This is because we do not back-propagate the RL-loss through the recurrent part, which allows us to make the PPO mini-



(a) AntDir (b) AntGoal

Figure 17: Learning curves for the MuJoCo AntDir (a) and AntGoal (b) environments, when backpropagating the RL loss through the encoder. Depending on the task, this can help (a) or hurt (b) performance, and critically depends on the relative weight between VAE and RL loss.

batch updates without having to re-compute the embeddings (so it saves us a lot of forward/backward passes through the recurrent model). This difference would likely be less if we used a different RL algorithm which does not rely on this many forward/backward passes per policy update.

C.5 Ablation Study: Backpropagating the RL loss Through the Encoder

In the main experiments presented in the paper, we do not backpropagate the RL loss through the encoder. Instead, we alternate between updating the VAE with the ELBO loss, and updating the policy with the PPO loss (and detaching the gradient when feeding the belief into the policy). We do so because this performs sufficiently well, and has two advantages: first, we do not have to calibrate the relative weighting between the RL and the VAE loss; second, it is much faster in practice because otherwise we would have to re-compute the belief embedding for each PPO mini-batch.

Figure 17 shows the learning curves of the variBAD agent in the AntDir and AntGoal environments, when backpropagating the RL loss compared to our standard setting and compared to RL^2. These results show that sometimes, combining the RL and VAE loss can marginally improve performance (Figure 17a), but it can also significantly hurt performance (Figure 17b) if the relative weighting is not calibrated correctly. In terms of experiment runtime, when backpropagating the RL loss through the encoder is as slow as RL^2 (around 66% slower in these environments).

Appendix D. Experiments: Meta-World

Figure 18 shows the learning curves for the Meta-World ML1 tasks for variBAD (20 seed averages with 95% confidence intervals). We followed the evaluation protocol of Yu et al. (2019).

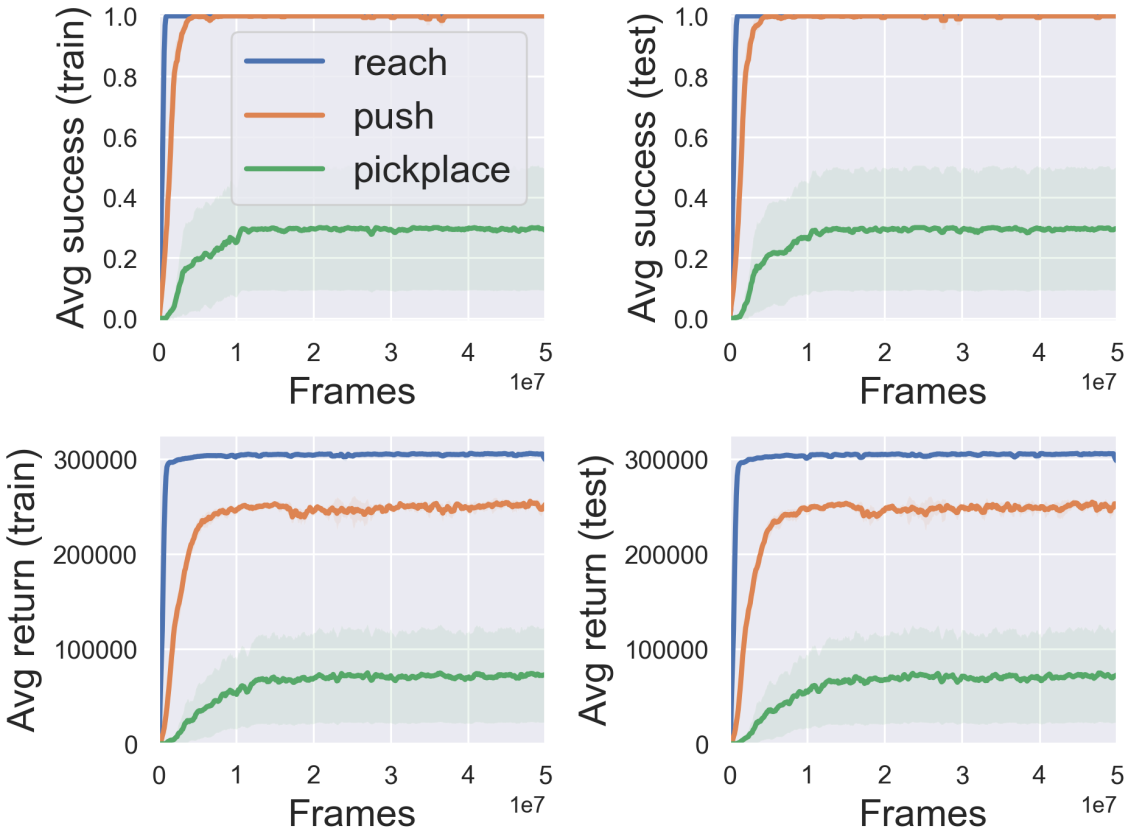


Figure 18: Learning curves for the different ML1 Meta-World tasks. Shown are the success rates (top row) and average returns (bottom row) for the training set (first column) and test set (second column).

Appendix E. Hyperparameters

We used the PyTorch framework (Paszke et al., 2017) for our experiments. The hyperparameters for GridWorld, MuJoCo CheetahDir, PointRobot and MetaWorld ML1-Push can be found in the tables below. For more details, see our reference implementation at `https://github.com/lmzintgraf/varibad`.

We used different number of seeds per experiment to balance significance of results and computational required, due to the inherent randomness/difficulty of different tasks. For the main experiments, we used 20 seeds for GridWorld/Navigation/Meta-World, and 10

seeds per MuJoCo environment. For the ablation studies, we used fewer for MuJoCo (5 instead of 10), and GridWorld (15 instead of 20) due to computational constraints.

	Grid World	Cheetah Dir	Point Robot	ML1 Push
max_rollouts_per_task	4	2	3	3
policy_state_embedding_dim	16	64	64	64
policy_latent_embedding_dim	16	64	64	64
norm_state_for_policy	True	True	True	True
norm_latent_for_policy	True	True	True	True
norm_rew_for_policy	True	True	True	True
norm_actions_pre_sampling	False	True	False	False
norm_actions_post_sampling	False	False	False	False
policy_layers	[32]	[128, 128]	[128, 128, 128]	[128, 128, 128]
policy_activation_function	tanh	tanh	tanh	tanh
policy_initialisation	normc	normc	normc	normc
policy_anneal_lr	False	False	False	False
policy	ppo	ppo	ppo	ppo
policy_optimiser	adam	adam	adam	adam
ppo_num_epochs	2	16	2	2
ppo_num_minibatch	4	4	8	8
ppo_clip_param	0.05	0.1	0.1	0.1
lr_policy	0.0007	0.0007	0.0007	0.0007
num_processes	16	16	16	16
policy_num_steps	60	800	200	200
policy_eps	1e-08	1e-08	1e-08	1e-08
policy_value_loss_coef	0.5	0.5	0.5	0.5
policy_entropy_coef	0.01	0.01	0.001	0.001
policy_gamma	0.95	0.97	0.99	0.99
policy_use_gae	True	True	True	True
policy_tau	0.95	0.9	0.9	0.9
use_proper_time_limits	False	True	True	True
encoder_max_grad_norm	None	1.0	None	None
decoder_max_grad_norm	None	1.0	None	None
lr_vae	0.001	0.001	0.001	0.001
size_vae_buffer	100000	10000	10000	10000
precollect_len	5000	5000	5000	5000
vae_buffer_add_thresh	1	1	1	1
vae_batch_num_trajs	25	10	15	15
tbptt_stepsize	None	50	None	None
vae_subsample_elbos	None	50	None	None
vae_subsample_decodes	None	50	None	None

	Grid World	Cheetah Dir	Point Robot	ML1 Push
vae_avg_reconstruction_terms	False	False	False	False
num_vae_updates	3	1	3	3
pretrain_len	0	0	0	0
kl_weight	0.01	0.1	1.0	1.0
action_embedding_size	0	16	16	16
state_embedding_size	8	32	32	32
reward_embedding_size	8	16	16	16
encoder_layers_before_gru	[]	[]	[]	[]
encoder_gru_hidden_size	64	128	128	128
encoder_layers_after_gru	[]	[]	[]	[]
latent_dim	5	5	5	5
decode_reward	True	True	True	True
rew_loss_coeff	1.0	1.0	1.0	1.0
input_prev_state	False	True	False	False
input_action	False	True	False	False
reward_decoder_layers	[32, 32]	[64, 32]	[64, 32]	[64, 32]
decode_state	False	False	False	False
state_loss_coeff	1.0	1.0	1.0	1.0
state_decoder_layers	[32, 32]	[64, 32]	[64, 32]	[64, 32]
disable_kl_term	False	False	False	False
rlloss_through_encoder	False	False	False	False

References

Isac Arnekvist, Danica Kragic, and Johannes A Stork. Vpe: Variational policy embedding for transfer reinforcement learning. In *International Conference on Robotics and Automation*, 2019.

John Asmuth and Michael L Littman. Learning is planning: near bayes-optimal reinforcement learning via monte-carlo tree search. In *Conf on Uncertainty in Artificial Intelligence*, 2011.

Richard Bellman. A problem in the sequential design of experiments. *Sankhyā: The Indian Journal of Statistics (1933-1960)*, 16(3/4):221–229, 1956.

Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, pages 3:213–231, 2002.

Emma Brunskill. Bayes-optimal reinforcement learning for discrete uncertainty domains. In *International Conference on Autonomous Agents and Multiagent Systems*, 2012.

Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *Twelfth National Conference on Artificial Intelligence*, 1994. AAAI Classic Paper Award, 2013.

Kristy Choi, Mike Wu, Noah Goodman, and Stefano Ermon. Meta-amortized variational inference and learning. In *International Conference on Learning Representation*, 2019.

John D Co-Reyes, YuXuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International Conference on Machine Learning*, 2018.

Christoph Dann, Lihong Li, Wei Wei, and Emma Brunskill. Policy certificates: Towards accountable reinforcement learning. 2019.

Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta learning of exploration. *arXiv preprint arXiv:2008.02598*, 2020.

Finale Doshi-Velez and George Konidaris. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *International Joint Conference on Artificial Intelligence*, 2016.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv:1611.02779*, 2016.

Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in Neural Information Processing Systems*, 2017.

Michael O'Gordon Duff and Andrew Barto. *Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, Univ of Massachusetts at Amherst, 2002.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. In *ICML 2018 Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6): 359–483, 2015.

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representation*, 2019.

Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Processing Systems*, 2012.

Arthur Guez, David Silver, and Peter Dayan. Scalable and efficient bayes-adaptive reinforcement learning based on monte-carlo tree search. *Journal of Artificial Intelligence Research*, 48:841–883, 2013.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Processing Systems*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representation*, 2018.

Rein Houthooft, Yuhua Chen, Phillip Isola, Bradly Stadie, Filip Wolski, OpenAI Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. In *Advances in Neural Information Processing Systems*, 2018.

Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. *arXiv:1905.06424*, 2019.

Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In *International Conference on Machine Learning*, 2019.

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low bellman rank are pac-learnable. In *International Conference on Machine Learning*, 2017.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002.

Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. Robust and efficient transfer learning with hidden parameter markov decision processes. In *Advances in neural information processing systems*, 2017.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representation*, 2014.

J Zico Kolter and Andrew Y Ng. Near-bayesian exploration in polynomial time. In *International Conference on Machine Learning*, 2009.

Lin Lan, Zhenguo Li, Xiaohong Guan, and Pinghui Wang. Meta reinforcement learning with task embedding and shared policy. In *International Joint Conference on Artificial Intelligence*, 2019.

Gilwoo Lee, Brian Hou, Aditya Mandalika, Jeongseok Lee, and Siddhartha S Srinivasa. Bayesian policy optimization for model uncertainty. In *International Conference on Learning Representation*, 2019.

Yun-En Liu, Travis Mandel, Emma Brunskill, and Zoran Popovic. Trading off scientific knowledge and user learning with multi-armed bandits. In *EDM*, pages 161–168, 2014.

James John Martin. *Bayesian decision problems and Markov chains*. Wiley, 1967.

Vladimir Mikulik, Grégoire Delétang, Tom McGrath, Tim Genewein, Miljan Martic, Shane Legg, and Pedro A Ortega. Meta-trained agents implement bayes-optimal agents. In *Advances in neural information processing systems*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Aditya Modi and Ambuj Tewari. Contextual markov decision processes using generalized linear models. In *Reinforcement Learning for Real Life Workshop at the International Conference on Machine Learning*, 2019.

Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv:1803.02999*, 2018.

Pedro A Ortega, Jane X Wang, Mark Rowland, Tim Genewein, Zeb Kurth-Nelson, Razvan Pascanu, Nicolas Heess, Joel Veness, Alex Pritzel, Pablo Sprechmann, et al. Meta-learning of sequential strategies. *arXiv:1905.03030*, 2019.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, 2013.

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

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Christian F Perez, Felipe Petroski Such, and Theofanis Karaletsos. Efficient transfer learning and online adaptation with latent variable models for continuous control. In *Continual Learning Workshop, NeurIPS 2018*, 2018.

Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An analytic solution to discrete bayesian reinforcement learning. In *International Conference on Machine Learning*, 2006.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning*, 2019.

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: proximal meta-policy search. In *International Conference on Learning Representation*, 2019.

Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. Meta reinforcement learning with latent variable gaussian processes. In *Conference on Uncertainty in Artificial Intelligence*, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Jonathan Sorg, Satinder Singh, and Richard L Lewis. Variance-based rewards for approximate bayesian reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, 2012.

Bradly C Stadie, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. Some considerations on learning to explore via meta-reinforcement learning. In *Advances in Neural Processing Systems*, 2018.

Malcolm Strens. A bayesian framework for reinforcement learning. In *International Conference on Machine Learning*, 2000.

Flood Sung, Li Zhang, Tao Xiang, Timothy Hospedales, and Yongxin Yang. Learning to learn: Meta-critic networks for sample efficient learning. *arXiv:1706.09529*, 2017.

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

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pages 5026–5033. IEEE, 2012. ISBN 978-1-4673-1737-5.

Sebastian Tschiatschek, Kai Arulkumaran, Jan Stühmer, and Katja Hofmann. Variational inference for data-efficient model learning in pomdps. *arXiv:1805.09281*, 2018.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. In *Annual Meeting of the Cognitive Science Community*, 2016.

Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. Robust imitation of diverse behaviors. In *Advances in Neural Information Processing Systems*, 2017.

David Wingate, Noah D Goodman, Daniel M Roy, Leslie P Kaelbling, and Joshua B Tenenbaum. Bayesian policy search with policy priors. In *International Joint Conference on Artificial Intelligence*, 2011.

Jiayu Yao, Taylor Killian, George Konidaris, and Finale Doshi-Velez. Direct policy transfer via hidden parameter markov decision processes. In *LLARLA Workshop, FAIM*, 2018.

Gregory Yauney and Pratik Shah. Reinforcement learning with action-derived rewards for chemotherapy and clinical trial dosing regimen selection. In *Machine Learning for Healthcare Conference*, pages 161–226, 2018.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, 2019. URL `https://arxiv.org/abs/1910.10897`.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Processing Systems*, 2017.

Amy Zhang, Harsh Satija, and Joelle Pineau. Decoupling dynamics and reward for transfer learning. In *ICLR workshop track*, 2018.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*, 2020.

Luisa Zintgraf, Leo Feng, Cong Lu, Maximilian Igl, Kristian Hartikainen, Katja Hofmann, and Shimon Whiteson. Exploration in approximate hyper-state space for meta reinforcement learning. In *International Conference on Machine Learning*, 2021.

Luisa M Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, 2019.