Predictive Coding Enhances Meta-RL To Achieve Interpretable Bayes-Optimal Belief Representation Under Partial Observability

Po-Chen Kuo
University of Washington
pckuo@uw.edu

Han Hou
Allen Institute for Neural Dynamics
han.hou@alleninstitute.org

Will Dabney
Google DeepMind
wdabney@gmail.com

Edgar Y. Walker
University of Washington
eywalker@uw.edu

Abstract

Learning a compact representation of history is critical for planning and generalization in partially observable environments. While meta-reinforcement learning (RL) agents can attain near Bayes-optimal policies, they often fail to learn the compact, interpretable Bayes-optimal belief states. This representational inefficiency potentially limits the agent's adaptability and generalization capacity. Inspired by predictive coding in neuroscience—which suggests that the brain predicts sensory inputs as a neural implementation of Bayesian inference—and by auxiliary predictive objectives in deep RL, we investigate whether integrating self-supervised predictive coding modules into meta-RL can facilitate learning of Bayes-optimal representations. Through state machine simulation, we show that meta-RL with predictive modules consistently generates more interpretable representations that better approximate Bayes-optimal belief states compared to conventional meta-RL across a wide variety of tasks, even when both achieve optimal policies. In challenging tasks requiring active information seeking, only meta-RL with predictive modules successfully learns optimal representations and policies, whereas conventional meta-RL struggles with inadequate representation learning. Finally, we demonstrate that better representation learning leads to improved generalization. Our results strongly suggest the role of predictive learning as a guiding principle for effective representation learning in agents navigating partial observability.

1 Introduction

In real-world environments, agents, biological or artificial, rarely have complete information about the environmental states crucial for decision-making and planning, as observations are often noisy and non-stationary. This issue is known in reinforcement learning (RL) as partial observability [1, 2, 3], and is a major challenge in the deployment of real-world RL systems [4]. In partially observable environments, learning optimal policies depends on the entire history of past interactions. An open question in RL under partial observability is how to learn an efficient representation that serves as a compact summary of the history while retaining the information crucial for policy learning.

Partially observable tasks can be formalized as partially observable Markov Decision Processes (POMDPs, Fig. 1A) [2, 3], allowing a Bayesian treatment by maintaining and updating a belief state using Bayesian inference [5]. To this end, meta-RL, in particular memory-based meta-RL, has been shown to provide powerful deep RL methods for developing agents that efficiently adapt under uncertainty [6, 7, 8, 9, 10, 11]. Meta-RL has been shown to behave near Bayes-optimally under partial observability, both theoretically and empirically [12, 13]. However, its learned representations are not equivalent to the minimally sufficient, Bayes-optimal belief states [13], hindering its interpretability, robustness, generalization capability, and learning of complex exploration [10, 11, 13].

Predictive learning has been suggested as a central coding principle that guides efficient neural processing in neuroscience. Humans and animals are shown to achieve near Bayes-optimal behavior under uncertainty, leading to theories and experiments investigating efficient Bayes-optimal representation and computation in the brain [14, 15, 16]. Predictive coding, postulating that the brain continuously predicts upcoming observations and updates its internal world models [17, 18], provides a neurally plausible implementation of Bayesian inference [19, 20] and has been instrumental in explaining diverse neural circuits including feature learning in sensory cortices [21, 22], motor control in the cerebellum [23], value learning in the striatum [24], and cognitive maps in the hippocampus [25]. In deep learning, predictive learning has been explored as a powerful self-supervised technique to support representation learning and improve task performance [26, 27]. In deep RL, predictive objectives are shown to regularize learning and prevent overfit or representational collapse [28, 29], improve sample efficiency [30], and link to Bayesian filtering [31]. Most recent breakthrough employs predictive learning to derive general agents that can solve a wide range of challenging tasks [32].

The main contribution of this paper is a systematic investigation of whether integrating self-supervised predictive coding into meta-RL yields interpretable, Bayes-optimal belief representations in partially observable environments. Although recent meta-RL models have explored predictive objectives to improve generalization and representation quality [9, 10], the exact mechanisms by which predictive learning enhances latent representations—and how this relates to improved performance—remain unclear. To fill this gap, we move beyond performance measures that lack interpretability to directly compare meta-RL agents against Bayes-optimal solutions, evaluating both behavior and representation across diverse partially observable tasks varying in belief-update complexity. Specifically, we:

- Employ a meta-RL framework incorporating self-supervised future predictive modules, which enables direct interpretation and comparison with Bayes-optimal belief states.
- Show systematically that meta-RL with predictive modules yields interpretable, task-relevant representations that capture underlying environmental structures and dynamics across diverse POMDPs, and significantly improves planning, exploration, and generalization.
- Demonstrate via rigorous state machine simulation analysis that meta-RL with predictive modules consistently generates representations that more closely approximate Bayes-optimal beliefs than conventional black-box meta-RL, providing a guiding principle for learning both optimal representations and policies in partially observable environments.

2 Background and related work

Partially observable Markov decision process (POMDP) A POMDP is defined by the tuple $\big(S, A, T, R, \Omega, O, \gamma\big)$, where S is the (finite) hidden state space, A the action space, $T(s' \mid s, a)$ the transition probabilities, $R(s, a)$ the reward function, Ω the observation space, $O(o \mid s', a)$ the emission function, and $\gamma \in [0, 1)$ the discount factor. Since the agent does not have access to the true state s, the decision process is non-Markovian with respect to s and an optimal POMDP policy generally depends on the entire history $\tau_{:t} = (o_1, \ldots, a_{t-1}, o_t)$. By maintaining a belief state $b_t \equiv p(s|h_t)$, a posterior over S summarizing the history, the POMDP can be cast as a fully-observable *belief* MDP with transitions in belief state b given by the Bayesian update: $b'(s') = \eta\, O\big(o \mid s', a\big) \sum_{s \in S} T\big(s' \mid s, a\big)\, b(s)$, where η is a normalization factor. In this formulation, belief states b are the minimally sufficient statistic of the history summarizing all the past information compactly, and the policy $\pi(b)$ over belief states restores the Markov property. Unfortunately, planning in a POMDP, i.e., computing a Bayes-optimal policy over belief states, is generally intractable for all but the simplest tasks [2, 3, 5].

Memory-based meta-RL Exemplified by RL^2 [6, 7], memory-based meta-RL emerges as a powerful method for learning adaptive policies across related tasks. Often parameterized as recurrent neural networks (RNNs), RL^2 implicitly maintains a memory of the past history in the recurrent states and learns a history-dependent policy by maximizing return over the task distribution using policy gradient algorithms (Fig. 1B). This design enables the agent to adapt dynamically as it interacts with

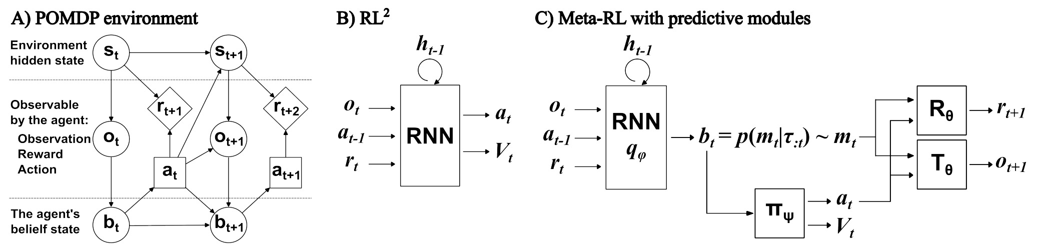

Figure 1: **Representation learning under partial observability using meta-RL with predictive modules.** A) In POMDPs, a belief state b_t over the environment hidden state is maintained for policy learning. B) Memory-based meta-RL (e.g. RL^2) simultaneously learns representation and policy using only the reward signal. C) The proposed meta-RL with self-supervised predictive modules separates representation learning with predictive coding and policy learning with reward signal.

the environment, effectively learning an internal RL algorithm via its recurrent computation [11]. Through end-to-end meta-learning over a task distribution, RL^2 provides effective black-box agents that achieve rapid adaptation to new task instances and generalize across related tasks.

The training objective of meta-RL can be recast and understood as approximating the Bayes-optimal policy for a task distribution [12]. Since history is sufficient for computing the belief state, black-box meta-RL can in principle learn Bayes-optimal policies by encoding the belief state in its hidden state. Several studies have demonstrated that belief states can be decoded from RNN hidden states [10, 33, 34]. Moreover, Ortega et al. [12] provided a theoretical framework for interpreting meta-RL as Bayes-optimal solutions, and Mikulik et al. [13] empirically showed that meta-RL agents learn near Bayes-optimal policy in bandit problems. However, a comparison of meta-learned representations in RL^2 using state machine simulation suggests that their representations are not equivalent to the minimally-sufficient Bayes-optimal states, likely due to failures in injectivity, which may hinder the agent's interpretability, robustness, and ability to generalize [13]. In fact, in practice, black-box meta-RL may struggle to efficiently learn Bayes-optimal policies in tasks requiring exploration that is temporally-extended and qualitatively different from exploitation behavior [9, 11, 35].

Predictive coding neural circuits In neuroscience, theoretical and experimental studies have explored predictive processing as a fundamental computational objective across brain regions and as a neural implementation of Bayesian computation. Sensory areas are often modeled as hierarchical neural networks predicting future sensory inputs [17, 21, 22], potentially supporting efficient neural representation for perceptual inference [15]. Motor systems are thought to predict the sensory consequences of the motor outputs [36], which can underlie efficient integration in sensorimotor learning [16]. The hippocampus, with neural activity believed to forecast an animal's upcoming experiences, is critical for relational learning and computing a predictive map or transition model to support efficient navigation [37, 38]. Building on the relationship between predictive learning and efficient neural representation, recent work has begun to explore adding predictive objectives to encourage artificial agents to develop activities similar to those observed in the brain [39].

Predictive learning in meta-RL Recent work has explored alternative architectures or auxiliary training objectives to learn latent representations that are predictive of future observations or rewards for augmenting meta-RL agents. For instance, Igl et al. [8] showed deep policy can be improved by adding predictive auxiliary loss, and proposed particle filter method for approximating belief inference. Several studies further decoupled learning recurrent generative models of the environment and used the learned latent representations as inputs for model-free RL algorithms, demonstrating improved performance especially for robotics tasks that are difficult for black-box models [40, 41, 42, 43]. For a special type of POMDP where hidden states are stationary, several works considered approximating task inference by taking advantage of privileged information [44], posterior sampling [45], or variational inference [9], providing effective meta-RL models for rapid adaptation to new task instances. Recent works have explored how to further improve representation learning through state-space modeling [10], normalizing flow technique [46], or decoupled belief modeling with separate random policies [47]. On the other hand, learning latent dynamical world models has also been proposed for model-based deep RL approaches, including SOLAR [48], PlaNet [30] and Dreamer

[32]. These studies demonstrated that learning latent world models through multi-step predictive objectives can strongly improve planning and performance in challenging control benchmarks.

Complementary to the above empirical progress of integrating predictive learning to improve performance, our work provides the interpretability foundation for why predictive objectives work in deep RL by focusing on understanding its meta-learned representation and computation.

3 Meta-RL with self-supervised predictive modules

Black-box meta-RL like RL^2 (Fig. 1B) requires simultaneously learning history representation and policy using only reward signals, potentially suffering from inadequate representation and policy learning [11, 13]. To overcome this challenge, we employ an end-to-end meta-RL framework with self-supervised predictive modules to learn explicit belief representations. Our motivations are two-fold: algorithmically, a good representation should summarize the relevant history predictive of the future and reflect uncertainty; neurobiologically, modular neural circuits are believed to perform future predictions, attaining efficient representations under predictive coding [17].

The proposed model, meta-RL with predictive modules, consists of a variational autoencoder (VAE) for predictive representation learning and a policy network operating on the learned representation (Fig. 1C). The predictive modules begin with an RNN encoder q_ϕ, which takes as input current observation o_t, reward r_t, and previous action a_{t-1}. The RNN outputs a low-dimensional bottleneck b_t which is the posterior distribution over the latent states m_t conditioned on the history $\tau_{:t}$. The posterior is trained via reward and observation prediction akin to predictive coding using the reward decoder R_θ and the observation decoder T_θ, respectively, to predict upcoming rewards and observations given the action taken by the policy network. The predictive modules optimize the evidence lower bound (ELBO) using the reparameterization technique [49], with KL regularization loss similar to Bayesian filtering (see A.1 for details). This approach learns an explicit probabilistic belief representation b_t over the latent state, facilitating a direct interpretation as belief states in POMDPs. The policy network π_ψ is a feedforward neural network receiving the belief state b_t as an input and can be efficiently trained using model-free policy gradient algorithms [50]. Note the policy gradient from RL loss is only used to train the policy network but not the RNN encoder. The entire model is trained in a self-supervised, end-to-end manner using the standard meta-learning paradigm.

Our parameterization is similar in principle to previous work [9, 10, 41], with designs tailored for POMDPs without strict assumptions over stationarity [9] and structures [10] of the hidden states, facilitating a direct interpretation of b_t as the belief states in POMDPs. Furthermore, the end-to-end approach neither relies on privileged information [44], nor requires a separate random policy to generate samples for training predictive models [47] as exploited in previous work.

4 Experiment

We designed the following experiments to answer the central question: Does meta-RL with self-supervised predictive modules enhance learning representations that more closely match the Bayes-optimal belief states than black-box meta-RL? Specifically, we adopt the *state machine simulation* analysis used in Mikulik et al. [13] to examine the equivalence of representation and computation between meta-RL agents and Bayes-optimal solutions. In essence, two state machines can be considered computationally equivalent if they can *simulate* each other—that is, if for any given state in one machine, we can find a mapping onto the other machine such that both their *state transitions* and *outputs* are the same [51]. This analysis examines whether there exists a consistent way of interpreting every state in one machine as a state in the other. Although decoding is commonly used to evaluate representation similarity [10, 33, 34], it only measures correlations between representations. In contrast, state machine simulation thoroughly assesses representational equivalence based on structural and computational relevance, providing a rigorous analysis for model interpretability.

State machine simulation In tasks where Bayes-optimal states are computationally tractable, we can compare meta-RL representations with Bayes-optimal belief states using state machine simulation [13]. The goal is to measure the *state* and *output dissimilarities* after mapping the states from one machine to the other. If both state and output dissimilarities are low in *both* mapping directions, then the two representations can be considered computationally equivalent. Briefly, the

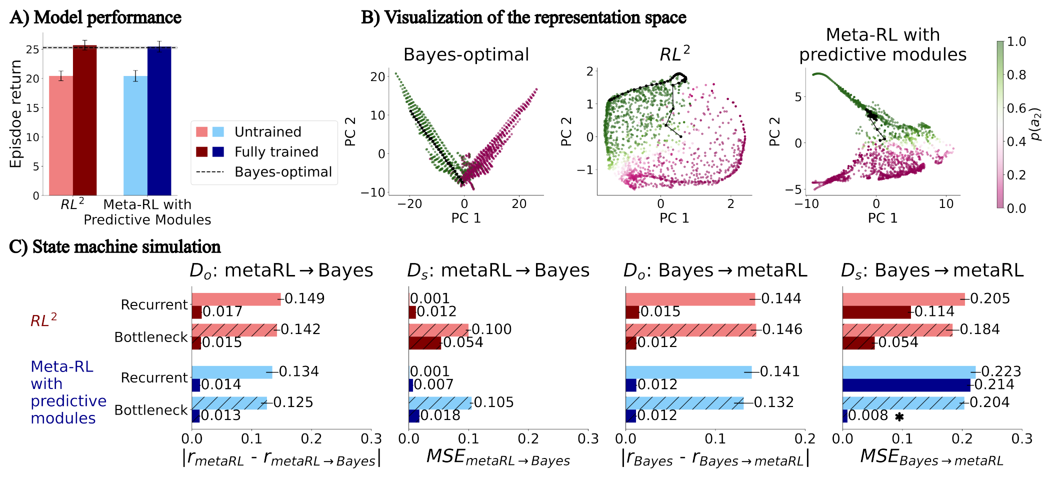

Figure 2: **Meta-RL with predictive modules better approximates Bayes-optimal states in bandit task.** A) Model performance against the Bayes-optimal policy. Light and dark colors denote untrained and fully trained models, respectively. B) Visualization of Bayes-optimal belief states, learned representation in the bottleneck layer of RL^2, and that of meta-RL with predictive modules. States are colored by the probability of choosing a_2. Black curves show one example trajectory. C) State machine simulation. *Recurrent* and *bottleneck* denote the layers used for analysis. The bottleneck layer of meta-RL with predictive modules achieves the lowest state dissimilarity D_s and output dissimilarity D_o in both mapping directions, indicating highest equivalence to Bayes-optimal states. (For all figures error bars denote the s.e.m. across at least 5 training runs, and asterisks denote statistical significance ($p < 0.05$, t-test) among the trained models.)

procedures are as follows (and see A.4 for details): (i) two mapping functions (parameterized as multi-layered perceptrons) are trained to map from meta-learned states to Bayes-optimal states and from Bayes-optimal states to meta-learned states; (ii) The *state dissimilarity* D_s is measured as the mean square error (MSE) of the mapped states against the target states; (iii) The *output dissimilarity* D_o is measured as the difference in return as generated by the output of the original states and the output of the mapped states. Here we compare D_s and D_o of meta-RL with predictive modules against those of black-box meta-RL (RL^2) to evaluate the quality of their learned representations.

Tasks To enable rigorous comparison between meta-RL agents and Bayes-optimal solutions, we select the following exemplar tasks for which Bayes-optimal solutions are computationally tractable. While previous work has only evaluated stationary bandits [13], our task suite covers diverse POMDP challenges: explore-exploit tradeoff, sequential decision-making, dynamic belief tracking, information gathering, and continuous control. To our knowledge, this is the first systematic evaluation of meta-RL representations using ground-truth comparison across diverse POMDPs. Following are the six classes of tasks evaluated (details of each task are described in A.2):

- **Classic two-armed Bernoulli bandit**: The first task is the classic multi-armed bandit also considered in Mikulik et al. [13], where the meta-learned representations in RL^2 models were found to deviate from the minimally sufficient Bayes-optimal states even though their policy converge to the Bayes-optimal one. We will examine if the proposed meta-RL with predictive modules can effectively learn Bayes-optimal belief states in two-armed bandits.
- **Dynamic two-armed bandit**: To evaluate dynamic belief tracking, we next consider a dynamic extension of the two-armed bandit. Each arm is in one of n possible discrete states (for simplicity we choose $n = 2$) with various reward probabilities. The state of each arm evolves according to an independent Markovian transition process. The task is designed such that the belief state update is analytically tractable, and that the Bayes-optimal policy can be derived using the value iteration algorithm over the belief space [1]. To examine different structural and dynamical configurations, we consider three parameter settings: (i) two arms of symmetric reward probability states and transition dynamics, (ii) two arms of asymmetric reward probability states, and (iii) two arms of asymmetric transition dynamics.

- **Stationary Tiger**: To exemplify sequential decision-making under uncertainty, we consider the classic POMDP Tiger task [3], where an agent chooses between two doors—one hiding a tiger (penalty=-100) and the other hiding a treasure (reward=10). The agent may additionally choose to pay a small penalty=-1 to "listen" to acquire noisy observations about the tiger's location. This simple yet powerful paradigm tests an agent's ability to balance information gathering with reward seeking. We consider two difficulty levels by varying the observation accuracy for the "listen" action.
- **Dynamic Tiger**: We extend the Tiger task to a dynamic version by allowing the tiger's location to change over time, following Markov transition dynamics. As information reliability and relevance are further corrupted by the dynamic nature of the hidden state, the agent has to balance listening more to increase confidence against making decisions earlier in case the information gathered becomes obsolete. This task design permits tractable belief updates and the Bayes-optimal policy can be derived using value iteration [1]. Similarly, we consider two difficulty levels by changing the observation accuracy.
- **Oracle bandit**: Black-box meta-RL often struggles when exploration and information seeking are required [11]. We hypothesize this is due to ineffective representation learning. To exemplify this point, we consider an illustrative *oracle bandit*: in an 11-arm bandit environment, one of the first ten arms a_{1-10} is the target arm that gives a payout of 5, whereas the remaining nine are non-target arms with a payout of 1. The last arm a_{11} is the "oracle" arm whose payout is ≤ 1 but informs the target arm in the form of $\frac{1}{10}$ of the target arm index (e.g. a reward of 0.3 from a_{11} indicates a_3 is the target arm). This is similar to tasks discussed in Wang et al. [7] but differs in that here no structured feedback is given, which makes representation learning more challenging and critical. A successful policy requires paying an immediate exploration cost to acquire information for long-term gain.
- **Latent goal cart**: Deriving Bayes-optimal solutions in continuous POMDPs are usually intractable, hindering rigorous representational equivalence analysis. To enable evaluation in continuous observation and action spaces, we design an exemplar continuous control task which still permits tractable Bayes-optimal belief inference and policy. In this task, an agent controls a continuous action, the velocity of a cart, to move along a 1-dimensional track to a hidden goal ($+1$ or -1) which needs to be inferred from the continuous observation (current position) and reward (noisified distance from the hidden goal) it receives.

Agents For each POMDP environment, two types of agents—the proposed meta-RL with predictive modules and the black-box meta-RL (i.e. RL^2)—were trained on the target task distribution. To facilitate comparison, the baseline RL^2 model is designed to be architecturally identical to meta-RL with predictive modules except for the decoder networks. Specifically, the RL^2 baseline consists of an RNN followed by a fully connected bottleneck layer which is the counterpart to the latent belief layer b_t in Fig.1C, and finally with a readout layer that generates action logits and value estimates (details see A.3). Identical layer sizes are used across both models throughout each experiment.

While the latent belief layer (i.e. the bottleneck layer) for meta-RL with predictive modules is the natural target to be interpreted as belief states, Mikulik et al. [13] evaluates the representations learned in the recurrent layers in RL^2 models. In what follows, we systematically analyze the representations learned in both the *recurrent* and *bottleneck* layers in each model using state machine simulation.

5 Results

Two-armed Bernoulli bandit The Bayes-optimal solution can be derived using the Gittins index method [52]. After training, both RL^2 and meta-RL with predictive modules approach Bayes-optimal return (Fig. 2A). Visualization of the bottleneck layers shows meta-RL with predictive modules learns a low-dimensional representation structurally more similar to the Bayes-optimal states than RL^2 (Fig. 2B). This observation is corroborated by the state machine simulation analysis results (Fig. 2C). Before training, both models have high D_s and D_o, indicating the untrained representations are far from Bayes-optimal. One notable exception is D_s of the recurrent layers for the meta-RL$\rightarrow$Bayes mapping direction, which further verifies that using decoding alone is not enough for evaluating the equivalence of two representations, as the untrained recurrent layers may maintain a verbose representation of the history and can be mapped arbitrarily close to the belief states if powerful enough mapping functions are used (see A.4 for detailed discussions).

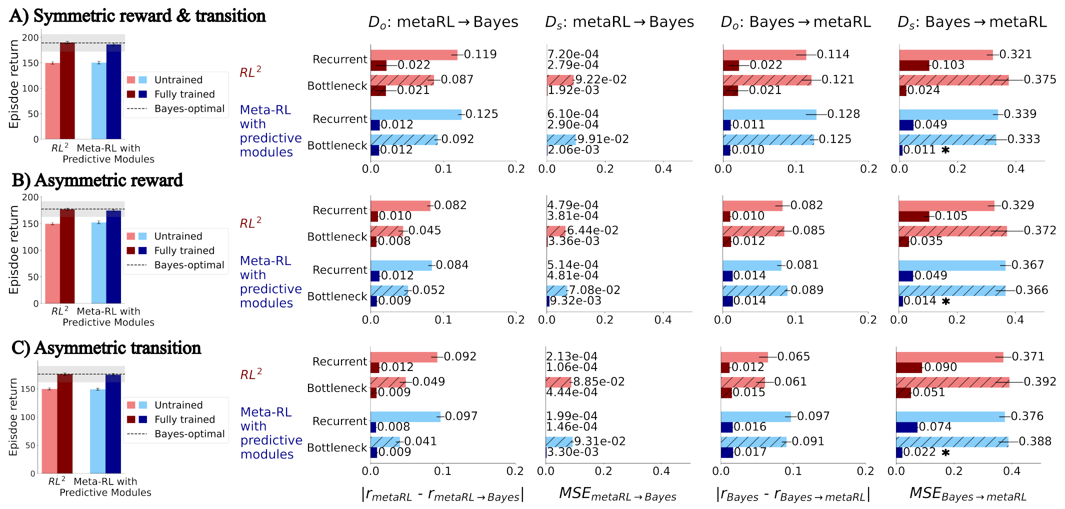

Figure 3: **Dynamic bandits.** Performance (left) and state machine simulation analysis (right) for A) the symmetric task, B) the asymmetric reward task, and C) the asymmetric transition task. Shaded areas in the left column denote the standard deviation of episodic return under Bayes-optimal policy. The bottleneck layer of meta-RL with predictive modules attains the lowest state dissimilarity D_s and output dissimilarity D_o in both mapping directions across tasks.

After training, for the recurrent layers in both models, the state dissimilarity D_s remains high for the Bayes→meta-RL direction, highlighting the representations in the recurrent layer deviate from the minimally-sufficient Bayes-optimal belief states, similar to the findings in Mikulik et al. [13]. In contrast, considering all four measures, the bottleneck layer in meta-RL with predictive modules achieves the lowest D_s and D_o in both mapping directions, indicating that its learned representations best approximate the Bayes-optimal belief states (Fig. 2C). Remarkably, the representation learning capacity does not merely arise from having a low-dimensional bottleneck, as D_s for the bottleneck layer in RL^2 remains significantly higher than that in meta-RL with predictive modules, suggesting that predictive coding encourages more compact, efficient representations in meta-RL agents.

Dynamic bandits Across the three dynamic bandit tasks—symmetric reward and transition, asymmetric reward, and asymmetric transition (Fig. 3A, B, and, C, respectively), both models approach the optimal return. Meta-RL with predictive modules learns representations structurally more similar to the Bayes-optimal states (see A.5.1). State machine simulation reveals that the recurrent layers in both models present with large D_s for the Bayes→meta-RL direction even after training. While the bottleneck layers in both models achieve comparably low output dissimilarities, meta-RL with predictive modules consistently achieve significantly lower state dissimilarity D_s in the Bayes→meta-RL mapping direction, indicating a higher equivalence to the Bayes-optimal belief states.

Stationary Tiger and Dynamic Tiger Difficulty of the Tiger tasks increases when the observation accuracy decreases (information becomes less reliable) and when the tiger's location is dynamic (information may become obsolete if waiting for too long). After training, both models approach the Bayes-optimal return in easier tasks, while for the most challenging dynamic Tiger with low observation accuracy, only meta-RL with predictive modules approach Bayes-optimal return (Fig. 4D). Meta-RL with predictive modules learns representations better capturing the task structure (see A.5.1). State machine simulation shows that the recurrent layers in both models fail to approximate Bayes-optimal belief states (large D_s after training in Fig. 4). Comparing the bottleneck layers, meta-RL with predictive modules consistently achieves higher equivalence to Bayes-optimal belief states, as indicated by the significantly lower D_s for the Bayes→meta-RL direction across all tasks.

Oracle bandit After training, only meta-RL with predictive modules learns the Bayes-optimal policy (Fig. 5A), paying an immediate cost to sample the oracle arm and utilize the information for long-term gain. In contrast, RL^2 converges to a suboptimal policy where it learns to sample the oracle arm upfront, but fails to use the attained information consistently. Visually, the bottleneck layer in

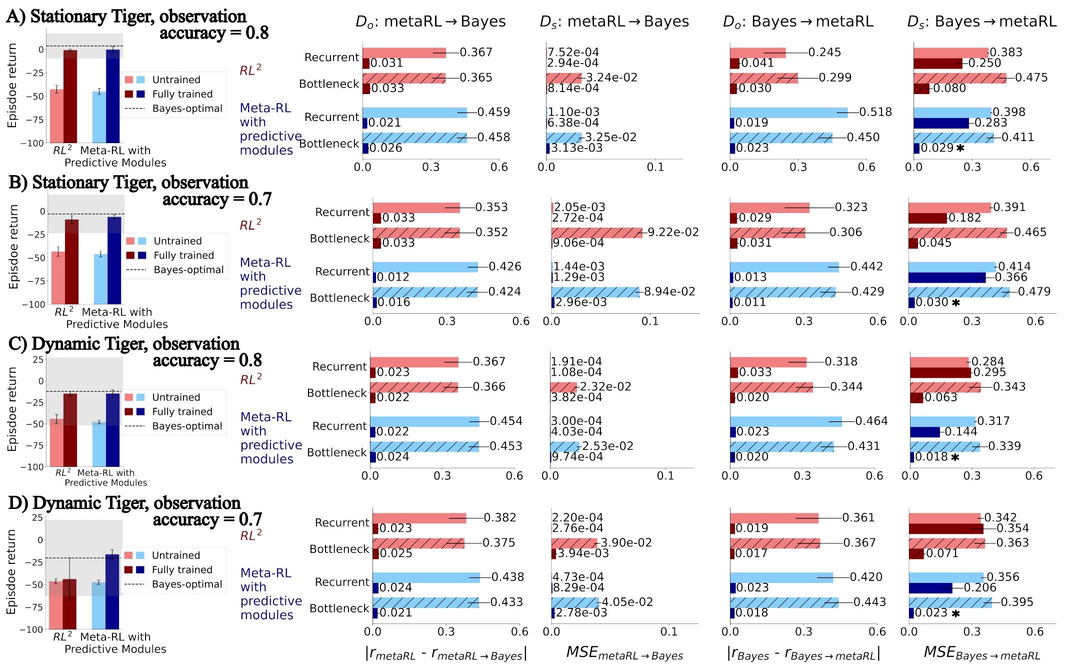
Figure 4: **Stationary and dynamic Tiger.** Performance (left) and state machine simulation analysis (right) for A–B) stationary Tiger with two levels of observation accuracies, and C–D) dynamic Tiger, where the tiger location changes with a probability of 0.9 at each timestep. Note for the most challenging setting in D, only meta-RL with predictive modules approaches the Bayes-optimal return.

meta-RL with predictive modules shows interpretable representations capturing task structures and dynamics. In contrast, RL^2 fails to learn an interpretable representation (Fig. 5B), likely contributing to its suboptimal behavior. State machine simulation shows that after training, the bottleneck layer in meta-RL with predictive modules achieves significantly lower state and output dissimilarities in both mapping directions, indicating close approximation of Bayes-optimal belief states (Fig. 5C). Dissimilarity measures remain high for RL^2 after training, suggesting that RL^2 may suffer from inadequate representation learning when the task requires exploration and information seeking.

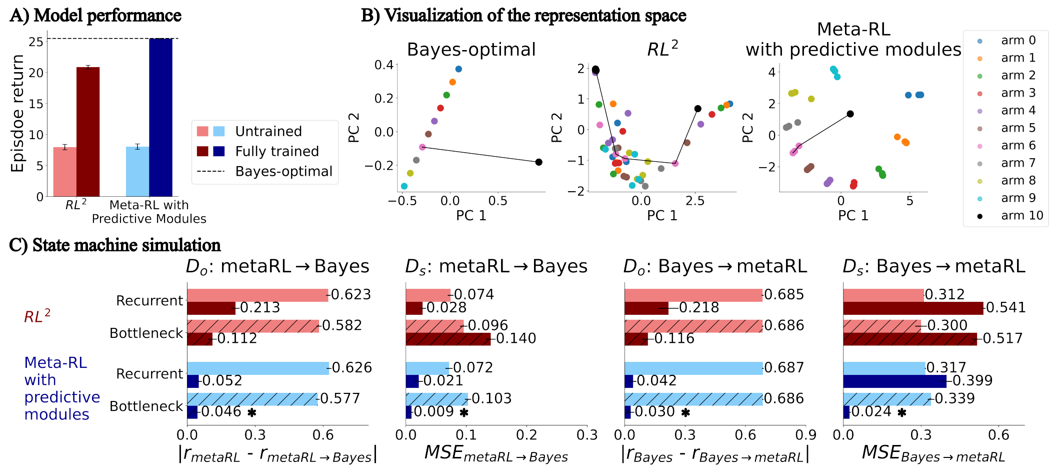
Figure 5: **Meta-RL with predictive modules learns interpretable Bayes-optimal solutions in oracle bandit.** A) Only meta-RL with predictive modules learns the optimal policy. B) Bayes-optimal states (left), the bottleneck layer of RL^2 (middle), and that of meta-RL with predictive modules (right). States are colored by the most likely arm to choose. C) The bottleneck layer in meta-RL with predictive modules attains the lowest state and output dissimilarities in both mapping directions.

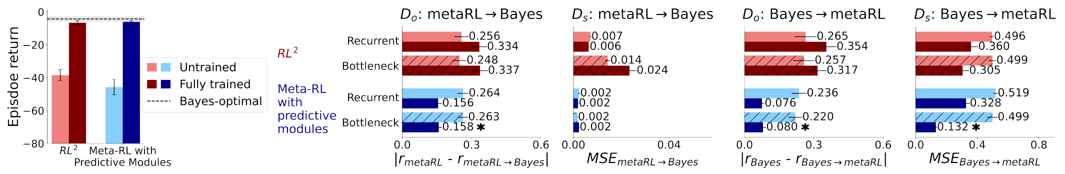
Figure 6: **Latent goal cart.** Performance (left) and state machine simulation analysis (right). The bottleneck layer of meta-RL with predictive modules attains the lowest state and output dissimilarities in both mapping directions, indicating close approximation of Bayes-optimal belief states.

Latent goal cart After training, both models approach the Bayes-optimal return (Fig. 6). Visualization of the states shows meta-RL with predictive modules learns representations better capturing the task structure (see A.5.1). State machine simulation shows that the recurrent layers in both models fail to approximate Bayes-optimal belief states (large D_s after training). In contrast, the bottleneck layers of meta-RL with predictive modules achieve the highest equivalence to Bayes-optimal belief states, as indicated by the significantly lower state and output dissimilarities in both mapping directions.

Improved generalization capability Thus far we present a systematic evaluation of representational equivalence between meta-RL with predictive modules and Bayes-optimal solutions across a diverse array of POMDP tasks. Our state machine simulation results strongly suggest that meta-RL with predictive modules can effectively learn interpretable representations that better approximate Bayes-optimal belief states, which in turn could lead to improved policy learning in challenging tasks. To further illustrate the importance and implications of learning Bayes-optimal belief state representations in partially observable environments, we examine whether learning better representations might lead to improved generalization capability to unseen but related tasks. To this end, we first evaluate zero-shot generalization capacity. As shown in Fig. 7A, when testing models trained on dynamic Tiger with observation accuracy of 0.7 on environments with observation accuracy of 0.8, meta-RL with predictive modules shows near-optimal zero-shot test return, whereas RL^2 has significantly lower test return, suggesting that through learning better representations, meta-RL with predictive modules could capture the underlying belief updates shared across similar tasks to facilitate zero-shot generalization. Next, we evaluate whether representation quality impacts transfer learning efficiency in out-of-distribution (OOD) tasks. As shown in in Fig.7B, when models are first trained on oracle bandits with a pre-training distribution that only contain arms 1-5 before transferring to a second task distribution that contains arms 6-10, meta-RL with predictive modules shows significantly faster transfer learning as compared to RL^2. This implies that with better representation learning, meta-RL with predictive modules could effectively learn the relevant environmental structure and dynamics to facilitate more efficient transfer learning to OOD tasks.

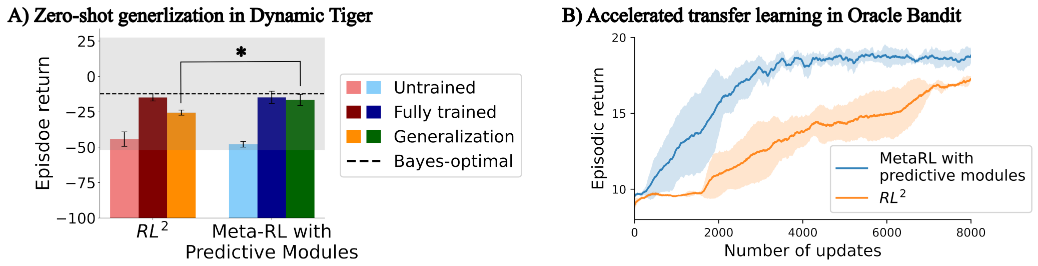

Figure 7: **Better representation leads to improved generalization capability.** A) When testing models trained on dynamic Tiger with observation accuracy of 0.7 on environments with observation accuracy of 0.8, meta-RL with predictive modules shows near-optimal zero-shot test return (-15.94 ± 3.82, green), whereas RL^2 has significantly lower test return (-25.56 ± 1.80, orange), indicating that meta-RL with predictive modules achieves better zero-shot generalization to unseen but relevant tasks. (Red and blue bars are reference model performance from Fig. 4C.) B) When models are first trained on oracle bandits with a task distribution that only contain arms 1-5 before transferring to a second task distribution that contains arms 6-10, meta-RL with predictive modules demonstrates significantly faster transfer learning as compared to RL^2, indicating that meta-RL with predictive modules achieves more efficient transfer learning to out-of-distribution relevant tasks.

Table 1: Ablation studies: changes in bidirectional state and output dissimilarities of the bottleneck layers compared to the proposed model of meta-RL with predictive modules. (M for meta-RL, B for Bayes-optimal solutions. Bold indicates $p < 0.05$.)

	$\Delta D_o^{M\to B}$	$\Delta D_s^{M\to B}$	$\Delta D_o^{B\to M}$	$\Delta D_s^{B\to M}$	$\Delta D_o^{M\to B}$	$\Delta D_s^{M\to B}$	$\Delta D_o^{B\to M}$	$\Delta D_s^{B\to M}$
	Two-arm Bernoulli bandit (unit: 1e-2)				Dynamic bandit (unit: 1e-2)			
No KL	0.12±0.4	**2.44±0.38**	-0.26±0.36	**0.21±0.04**	-0.16±0.53	0.42±0.38	-0.13±0.38	**-0.65±0.15**
joint RL	**-0.41±0.05**	-0.61±0.15	-0.27±0.12	-0.08±0.04	-0.09±0.06	0.08±0.13	0.0±0.04	0.47±0.2
	Dynamic Tiger (unit: 1e-2)				Oracle bandit (unit: 1e-2)			
No KL	1.11±1.35	0.22±0.17	**1.49±0.6**	1.11±1.76	**9.22±4.44**	**23.47±6.94**	**56.3±8.43**	0.24±0.82
joint RL	1.1±0.96	**0.21±0.05**	0.95±1.66	0.62±0.84	**10.91±1.02**	0.24±0.82	5.53±5.0	1.77±3.38

Ablation studies Finally, to pinpoint which algorithmic design choices drive the enhanced representation learning capacity in meta-RL with predictive modules, we performed two targeted ablations: (i) no KL: we remove the VAE's KL regularization to test whether latent regularization is necessary to enforce compact representations, and (ii) joint RL: we allow the policy gradients from RL loss to jointly train the RNN encoder alongside the predictive loss—opposed to training the encoder *solely* with the predictive loss as our proposed model does—to assess whether policy gradients provide additional representational benefits. Table 1 summarizes changes in dissimilarities from state machine simulation analysis on exemplar tasks (see A.5.2 and A.5.3 for full results). First, without KL regularization on the latent space, bottleneck dissimilarities to Bayes-optimal states increase (i.e. worse alignment) on most tasks (except for the minor decrease in dynamic bandits), showing that *suitable* regularization may help to promote compact, belief-like representations. On the other hand, under joint RL training, bottleneck dissimilarities are not further decreased as compared to the predictive loss-only models, indicating that additional policy gradients do not further improve representations and hence are not necessary for learning Bayes-optimal representations. This indicates that the enhanced representation learning capacity can be largely attributed to predictive learning.

6 Discussion and conclusions

In this work we employed rigorous state machine simulation analysis to demonstrate that meta-RL with self-supervised predictive coding modules can effectively learn interpretable, near Bayes-optimal belief state representations across diverse partially observable tasks, whereas conventional black-box meta-RL fails to capture the minimally sufficient representation. We further showed how the enhanced representation learning capacity in meta-RL with predictive modules not only supports effective information gathering and policy learning in challenging tasks, but also leads to improved generalization capability to unseen but relevant tasks. Our results align with predictive coding theories in neuroscience, suggesting that predictive objectives constitute a general computational strategy for achieving Bayesian computation, and highlight predictive learning as a fundamental principle for guiding efficient representation learning in agents navigating partially observable environments.

Scope and limitations We considered POMDPs with tractable Bayes-optimal solutions to enable rigorous analysis. Future work on more scalable analysis methods for tasks with larger state space will be valuable. In addition, we examined next-step future prediction as a general predictive objective. Considering the empirical success of multi-step predictive objective in deep RL, one promising direction is to investigate how incorporating multi-step, different time scales, and temporal abstractions into predictive modules might further enhance representation learning. Finally, we showed representation learning affects an agent's ability for generalization, but a more comprehensive evaluation is needed to explore how to meta-learn representations to support OOD generalization.

Broader Impact Our work advances model interpretability by linking meta-learned representations to Bayes-optimal beliefs, offering a Bayesian account for the success of predictive objectives in deep RL. This understanding is critical for developing safe, adaptable agents that operate under uncertainty and limited information—such as in medical decision support and disaster response. Moreover, our approach shows how insights from neuroscience theories may contribute to opening the black box of modern AI. Conversely, uncovering the computational principles that drive representation learning in artificial agents may in turn shed light on the mechanisms of neural computation in the brain.

Acknowledgements We thank Patrick Zhang and Matthew Storm Bull for their insightful feedback and discussions. We gratefully acknowledge the community support from the University of Washington Computational Neuroscience Center and the Allen Institute for Neural Dynamics. We also thank the anonymous reviewers for their valuable and constructive review.

Code availability The codebase for model training and state machine simulation is available at: https://github.com/walkerlab/metaRL-predictive-representation

References

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[2] Anthony R Cassandra, Leslie Pack Kaelbling, and Michael L Littman. Acting optimally in partially observable stochastic domains. In *Aaai*, volume 94, pages 1023–1028, 1994.

[3] 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.

[4] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

[5] Jayakumar Subramanian, Amit Sinha, Raihan Seraj, and Aditya Mahajan. Approximate information state for approximate planning and reinforcement learning in partially observed systems. *Journal of Machine Learning Research*, 23(12):1–83, 2022.

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

[7] 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. *arXiv preprint arXiv:1611.05763*, 2016.

[8] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In *International conference on machine learning*, pages 2117–2126. PMLR, 2018.

[9] Luisa Zintgraf, Sebastian Schulze, Cong Lu, Leo Feng, Maximilian Igl, Kyriacos Shiarlis, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: Variational bayes-adaptive deep rl via meta-learning. *Journal of Machine Learning Research*, 22(289):1–39, 2021.

[10] Kei Akuzawa, Yusuke Iwasawa, and Yutaka Matsuo. Estimating disentangled belief about hidden state and hidden task for meta-reinforcement learning. *PMLR*, pages 73–86, 2021.

[11] Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. *arXiv preprint arXiv:2301.08028*, 2023.

[12] 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 preprint arXiv:1905.03030*, 2019.

[13] Vladimir Mikulik, Grégoire Delétang, Tom McGrath, Tim Genewein, Miljan Martic, Shane Legg, and Pedro Ortega. Meta-trained agents implement bayes-optimal agents. *Advances in neural information processing systems*, 33:18691–18703, 2020.

[14] Marc O Ernst and Martin S Banks. Humans integrate visual and haptic information in a statistically optimal fashion. *Nature*, 415(6870):429–433, 2002.

[15] Wei Ji Ma, Jeffrey M Beck, Peter E Latham, and Alexandre Pouget. Bayesian inference with probabilistic population codes. *Nature neuroscience*, 9(11):1432–1438, 2006.

[16] Konrad P Körding and Daniel M Wolpert. Bayesian integration in sensorimotor learning. *Nature*, 427(6971):244–247, 2004.

[17] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

[18] Karl Friston. A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, 360(1456):815–836, 2005.

[19] Christopher Summerfield and Floris P De Lange. Expectation in perceptual decision making: neural and computational mechanisms. *Nature Reviews Neuroscience*, 15(11):745–756, 2014.

[20] Rajesh PN Rao. Decision making under uncertainty: a neural model based on partially observable markov decision processes. *Frontiers in computational neuroscience*, 4:146, 2010.

[21] Shohei Furutachi, Alexis D Franklin, Andreea M Aldea, Thomas D Mrsic-Flogel, and Sonja B Hofer. Cooperative thalamocortical circuit mechanism for sensory prediction errors. *Nature*, pages 1–9, 2024.

[22] Chengxu Zhuang, Siming Yan, Aran Nayebi, Martin Schrimpf, Michael C Frank, James J DiCarlo, and Daniel LK Yamins. Unsupervised neural network models of the ventral visual stream. *Proceedings of the National Academy of Sciences*, 118(3):e2014196118, 2021.

[23] Daniel M Wolpert, R Chris Miall, and Mitsuo Kawato. Internal models in the cerebellum. *Trends in cognitive sciences*, 2(9):338–347, 1998.

[24] Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

[25] J O'Keefe. The hippocampus as a cognitive map, 1978.

[26] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[27] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[28] Clare Lyle, Mark Rowland, Georg Ostrovski, and Will Dabney. On the effect of auxiliary tasks on representation dynamics. In *International Conference on Artificial Intelligence and Statistics*, pages 1–9. PMLR, 2021.

[29] Will Dabney, André Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G Bellemare, and David Silver. The value-improvement path: Towards better representations for reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 7160–7168, 2021.

[30] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[31] Bryan Lim, Stefan Zohren, and Stephen Roberts. Recurrent neural filters: Learning independent bayesian filtering steps for time series prediction. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[32] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pages 1–7, 2025.

[33] Gaspard Lambrechts, Adrien Bolland, and Damien Ernst. Recurrent networks, hidden states and beliefs in partially observable environments. *arXiv preprint arXiv:2208.03520*, 2022.

[34] Jay A Hennig, Sandra A Romero Pinto, Takahiro Yamaguchi, Scott W Linderman, Naoshige Uchida, and Samuel J Gershman. Emergence of belief-like representations through reinforcement learning. *PLOS Computational Biology*, 19(9):e1011067, 2023.

[35] Evan Z Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. In *International conference on machine learning*, pages 6925–6935. PMLR, 2021.

[36] Anmo J Kim, Jamie K Fitzgerald, and Gaby Maimon. Cellular evidence for efference copy in drosophila visuomotor processing. *Nature neuroscience*, 18(9):1247–1255, 2015.

[37] Daniela Schiller, Howard Eichenbaum, Elizabeth A Buffalo, Lila Davachi, David J Foster, Stefan Leutgeb, and Charan Ranganath. Memory and space: towards an understanding of the cognitive map. *Journal of Neuroscience*, 35(41):13904–13911, 2015.

[38] James CR Whittington, Timothy H Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy EJ Behrens. The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263, 2020.

[39] Ching Fang and Kimberly L Stachenfeld. Predictive auxiliary objectives in deep rl mimic learning in the brain. *arXiv preprint arXiv:2310.06089*, 2023.

[40] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

[41] Dongqi Han, Kenji Doya, and Jun Tani. Variational recurrent models for solving partially observable control tasks. *arXiv preprint arXiv:1912.10703*, 2019.

[42] Karol Gregor, Danilo Jimenez Rezende, Frederic Besse, Yan Wu, Hamza Merzic, and Aaron van den Oord. Shaping belief states with generative environment models for rl. *Advances in Neural Information Processing Systems*, 32, 2019.

[43] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752, 2020.

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

[45] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

[46] Xiaoyu Chen, Yao Mark Mu, Ping Luo, Shengbo Li, and Jianyu Chen. Flow-based recurrent belief state learning for pomdps. In *International Conference on Machine Learning*, pages 3444–3468. PMLR, 2022.

[47] Andrew Wang, Andrew C Li, Toryn Q Klassen, Rodrigo Toro Icarte, and Sheila A McIlraith. Learning belief representations for partially observable deep rl. In *International Conference on Machine Learning*, pages 35970–35988. PMLR, 2023.

[48] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. Solar: Deep structured representations for model-based reinforcement learning. In *International conference on machine learning*, pages 7444–7453. PMLR, 2019.

[49] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[50] Volodymyr Mnih. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016.

[51] Antoine Girard and George J Pappas. Approximate bisimulations for nonlinear dynamical systems. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 684–689. IEEE, 2005.

[52] John C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 41(2):148–164, 1979.

[53] Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`, 2018.

A Technical Appendices and Supplementary Material

A.1 Evidence lower bound (ELBO)

At a given time step t, the learning objective of the RNN encoder q_ϕ is to maximize

$$\mathbb{E}_{\rho(\tau_{0:t})}[\log p_\theta(o_{t+1}, r_{t+1}|a_{0:t})] \tag{1}$$

where $\rho(\tau_{0:t})$ is the distribution of trajectories generated by the policy π_ψ. Given that Eq. 1 is intractable, we can instead optimize a tractable evidence lower bound (ELBO), computed as:

$$\begin{aligned}\mathcal{L} = \mathbb{E}_{\rho(\tau_{0:t})}\Bigg[\mathbb{E}_{\Pi_{t=0}^{T-1} q_\phi(m_t|\tau_{0:t})} \sum_{t=0}^{T-1} \Big\{ [\log p_\theta(o_{t+1}|m_t, a_t) + \log p_\theta(r_{t+1}|m_t, a_t)] \\ - D_{KL}[q_\phi(m_t|\tau_{0:t})||p_\theta(m_t)] \Big\}\Bigg]\end{aligned} \tag{2}$$

The terms $\mathbb{E}_q[\log p_\theta(o_{t+1}|m_t, a_t) + \log p_\theta(r_{t+1}|m_t, a_t)]$ are the future predictive loss, and the term $D_{KL}[q_\phi(m_t|\tau_{0:t})||p_\theta(m_t)]$ is a regularization term using Kullback-Leibler (KL) divergence between the variational posterior $q_\phi(m_t|\tau_{0:t})$ and the prior over the latent variables $p_\theta(m_t)$. To encourage the variational autoencoder(VAE) to approximate Bayesian filtering for belief update, the prior is set to the previous posterior $q_\phi(m_{t-1}|\tau_{0:t-1})$ with the initial prior $q_\phi(m_0|\tau_0) = \mathcal{N}(0, I)$. To optimize the ELBO (Eq. 2), the expectation is approximated with Monte Carlo sampling as in standard VAE training [49].

A.2 Task details

Two-armed Bernoulli bandit We consider a two-armed Beta-Bernoulli bandit task with an episode length of 40 time steps. At the beginning of each episode, θ_1 and θ_2, the reward probability biases for the two arms, are independently drawn from a fixed Beta distribution, $p(\theta_a) = Beta(1, 1)$. θ_1 and θ_2 are then used to define the Bernoulli reward distributions for arm $a \in \{1, 2\}$, respectively. The reward biases θ_a are hidden from the agent. At each time step, the agent chooses an arm $a \sim \pi$ sampled from its policy, and receives a binary reward sampled from $Ber(\theta_a)$, i.e. $r_t \sim p(r|\theta_a) = Ber(\theta_a)$. The discounted cumulative return is computed with a discount factor $\gamma = 0.95$. For multi-armed bandit tasks, Bayes-optimal policy can be derived using the Gittins index method [52], and the minimally sufficient Bayes-optimal belief state is to keep track of the count of total pulls and the count of rewarded pulls of each arm. In other words, for the two-armed bandit task, the Bayes-optimal belief states are 4-dimensional: $(n_{a_1}, n_{r_{a1}}, n_{a_2}, n_{r_{a2}})$, where n_a and n_{r_a} denote the count of total pulls and the count of rewarded pulls for arm a, respectively.

Dynamic two-armed bandit To evaluate the meta-learned representation and behavior when the underlying environmental state can change over time, we consider a dynamic version of the two-armed bandit. At a given time step, each arm is in one of n possible discrete states. For simplicity we choose $n = 2$. We denote the hidden state of arm a at time t as s_t^a, where $s_t^a \in \{\theta_1^a, \theta_2^a\}, \forall a \in \{1, 2\}$, with each state associated with a different reward probability, i.e., $r_t^a \sim p(r|s_t^a) = Ber(s_t^a)$. The state of each arm evolves according to an independent Markovian transition process independent of the action chosen: $p(s_{t+1}^a|s_t^a, a') = p(s_{t+1}^a|s_t^a) = T^a_{s_t^a, s_{t+1}^a}$. The episode length is set to be 300 time steps. To examine different structural and dynamical configurations, we consider three parameter settings:

- Symmetric reward and transition: the two arms share the same set of discrete states where $s_t^a \in \{0.1, 0.9\}, \forall a \in \{1, 2\}$, and the same transition dynamics where $T^a_{0.1,0.1} = T^a_{0.9,0.9} = 0.9$ and $T^a_{0.1,0.9} = T^a_{0.9,0.1} = 0.1$, $\forall a \in \{1, 2\}$. That is, for each arm, state 1 is highly rewarding whereas state 2 is less rewarding, and the state of each arm stays with a probability of 0.9 and switches with a probability of 0.1.
- Asymmetric reward: two arms share the same transition dynamics, but the reward probability states of the two arms are different. Specifically, the reward probability states for arm a_1

are $s_t^1 \in \{0.1, 0.9\}$, and for arm a_2 are $s_t^2 \in \{0.4, 0.6\}$, respectively. Therefore the two states of arm a_1 are more differentiating than those of arm a_2. The transition dynamics is governed by $T^1_{0.1,0.1} = T^1_{0.9,0.9} = T^2_{0.4,0.4} = T^2_{0.6,0.6} = 0.9$, and $T^1_{0.1,0.9} = T^1_{0.9,0.1} = T^1_{0.4,0.6} = T^2_{0.6,0.4} = 0.1$. In other words, both arms follow the same dynamics that stay with a probability of 0.9 and switch with a probability of 0.1.

- Asymmetric transition: two arms share the same reward probability states $s_t^a \in \{0.1, 0.9\}, \forall a \in \{1, 2\}$, but the transition dynamics are different. For arm a_1: $T^1_{0.1,0.1} = T^1_{0.9,0.9} = 0.9$ and $T^1_{0.1,0.9} = T^1_{0.9,0.1} = 0.1$; for arm a_2, $T^2_{0.1,0.1} = T^2_{0.9,0.9} = T^2_{0.1,0.9} = T^2_{0.9,0.1} = 0.5$. That is, whereas the transition dynamics for arm a_1 is the same as previous scenarios, the transition dynamics for arm a_2 is random with equal stay and switch probabilities of 0.5.

With this task design, the belief state updates in the dynamic two-armed bandit POMDP tasks are analytically tractable by computing the posterior probability of each arm being in state 1 conditioned on the history: $b_t = (p(s_t^1 = \theta_1^1 | h_t), p(s_t^2 = \theta_1^2 | h_t))$. This belief update is tractable using standard Bayesian inference. In turn, the Bayes-optimal policy can be derived using the value iteration algorithm [1] by discretizing the 2-dimensional belief state space. The discounted cumulative return is computed with a discount factor $\gamma = 0.95$.

Stationary Tiger To exemplify sequential decision-making under uncertainty, we consider the classic POMDP Tiger task [3]. In the tiger environment, an agent chooses between two doors—one hiding a tiger (penalty=-100) and the other hiding a treasure (reward=10). The agent may additionally choose to pay a small penalty=-1 for the "listen" action to acquire noisy observations about the tiger's location, The reliability of the noisy observation is controlled by the parameter of observation accuracy. To succeed, the agent must maintain a belief about the tiger's location to decide which door to open. This simple yet powerful paradigm tests an agent's ability to balance information gathering with reward-seeking, making it an ideal benchmark for evaluating RL algorithms in POMDP. We consider two difficulty levels by varying the observation accuracy for the "listen" action. Specifically, we consider a simpler task variant where the observation accuracy is 0.8 and a harder task variant where the observation accuracy is 0.7. For the Tiger tasks, the Bayes-optimal agent tracks the belief of the tiger's location, for instance, by computing the posterior probability of the tiger being on the left given the history, $b_t = p(\text{tiger at the left}|h_t)$, using standard Bayesian inference. The Bayes-optimal policy can be derived using value iteration [1] by discretizing the 1-dimensional belief state space. The discounted cumulative return is computed with a discount factor $\gamma = 0.95$. Typically, as the observation accuracy decreases, the Bayes-optimal solution will require listening for more times as observations are noisier before the belief states crosses the decision threshold for the optimal agent to decide on which door to open.

Dynamic Tiger We extend the classic Tiger task to a dynamic version by allowing the tiger's location to change over time, following Markov transition dynamics. We denote the tiger location s to be in one of the 2 states $s \in \{L, R\}$, and the tiger location evolves according to an independent Markovian transition process independent of the action chosen: $p(s_{t+1}|s_t, a) = p(s_{t+1}|s_t) = T_{s_t,s_{t+1}}$. Specifically, we choose $T_{L,L} = T_{R,R} = 0.9$ and $T_{L,R} = T_{R,L} = 0.1$. In other words, at each time step, the tiger stays with a probability of 0.9 and switches its location with a probability of 0.1. As the information reliability and relevance are further corrupted by the dynamic nature of the hidden state, the task becomes more challenging because the agent has to balance listening more to increase its confidence against making decisions earlier in case the information gathered so far becomes obsolete. This task design permits tractable belief updates by tracking the posterior probability of the tiger being on the left given the history, $b_t = p(\text{tiger at the left}|h_t)$, using standard Bayesian inference incorporating the Markovian transition matrix. The Bayes-optimal policy can be derived using value iteration [1] by discretizing the 1-dimensional belief state space. The discounted cumulative return is computed with a discount factor $\gamma = 0.95$. Similarly to the stationary Tiger tasks, we consider two difficulty levels by varing the observation accuracy to be 0.8 or 0.7.

Oracle bandit task The oracle bandit task is designed to exemplify an environment where a successful policy requires paying an immediate exploration cost and utilize the information acquired to improve long-term return. In an 11-arm bandit environment with an episode length of 6 time steps, one of the first ten arms a_{1-10} is selected uniformly randomly as the target arm a^*, which will give a payout of 5 upon choosing. The other nine arms out of a_{1-10} are non-target arms, each of which will give a payout of 1 upon choosing. The last arm, a_{11}, is the oracle arm whose payout informs the index of the target arm in the form of $1/10$ of the target arm index a^*, i.e. $r(a_{11}) = 0.1 * a^*$ (e.g. a reward of 0.3 from a_{11} indicates that a_3 is the target arm). The average reward from the oracle arm is 0.55 which is smaller than the payout from the non-target arms. As a result, choosing the oracle arm is less favorable in terms of the immediate reward but provides useful information if an agent knows how to utilize to improve long-term return. The discounted cumulative return is computed with a discount factor $\gamma = 0.95$. Note that this setting is similar to a task considered in Wang et al. [7] but differs in that in their setup the oracle information was provided using a structured one-hot encoding format, but in our formulation no other feedback than the reward itself is given, which makes learning a good representation of the history more challenging and critical. With knowledge of the task structure, the Bayes-optimal belief state is to track and update the posterior probability of each of the first ten arms being the true target arm conditioned on the history, i.e. $b_t = (p(a^* = a_1|h_t), p(a^* = a_2|h_t), ..., p(a^* = a_{10}|h_t))$. The Bayes-optimal policy is to pull the oracle arm at the beginning and continue pulling the target arm as informed by the reward information from the oracle arm until the end of the episode.

Latent goal cart Deriving Bayes-optimal solutions in continuous POMDPs are usually intractable, hindering rigorous representational equivalence analysis. To enable evaluation in continuous observation and action spaces, we design an exemplar continuous control task which still permits tractable Bayes-optimal belief inference and policy. In this task, an agent controls a continuous action, the velocity of a cart, to move along a 1-dimensional track to a hidden goal ($+1$ or -1) which needs to be inferred from the continuous observation (current position) and reward (negative noisified distance from the hidden goal, with the noise following a Gaussian distribution) it receives. We consider an episode length of 30 time steps. This task design allows for tractable belief updates by tracking the posterior probability of the hidden goal being at $+1$ given the history, $b_t = p(\text{goal at } +1|h_t)$, using standard Bayesian inference incorporating the knowledge that the noise in reward follows a Gaussian distribution. The Bayes-optimal policy can be derived using value iteration [1] by discretizing the 1-dimensional belief state space. The discounted cumulative return is computed with a discount factor $\gamma = 0.95$. This task is motivated by the Half-Cheetah-Dir task in MuJoCo, and can be seen as a simplified version to allow for tractable belief updates and value iteration to derive the Bayes-optimal solution.

A.3 Agent details

RL^2 The implementation of the baseline black-box memory-based meta-RL models, RL^2, follows an actor-critic architecture as in previous literature [6, 7]. Here we use RNNs that function as a memory module, consisting of 256 hidden units and hyperbolic tangent activation functions. As shown in Fig. 1B, the input includes the current observation o_t, the previous action in one-hot format a_{t-1}, and the associated reward in scalar format r_t. The recurrent layer is followed by a fully connected bottleneck layer designed to be the counterpart to the latent belief layer b_t in Fig.1C. After the bottleneck layer is a hidden layer of 32 units and a readout layer that generates a vector of logits for each action a_t for the actor (or a vector that defines the mean and the standard deviation of a continuous Gaussian policy if solving a continuous task such as the Latent goal cart) and a scalar value baseline V_t for the critic. Actions are then sampled from the softmax distribution defined by the action logits. The network is trained end-to-end to maximize the discounted cumulative reward with the Advantage Actor Critic algorithm [50] (Parts of the implementation code are based on [53], under MIT license). The gradient of the objective function is given by:

$$\begin{aligned}\nabla\mathcal{L}_{A2C} &= \nabla\mathcal{L}_{\pi} + \nabla\mathcal{L}_{V} + \nabla\mathcal{L}_{entropy}\\ &= \frac{\partial \log \pi(a_t|\tau_{:t};\psi)}{\partial\psi}\delta_t(\tau_{:t};\psi_V) + \beta_V\delta_t(\tau_{:t};\psi_V)\frac{\partial V}{\partial\psi_V} + \beta_e\frac{\partial H(\pi(a_t|\tau_{:t};\psi))}{\partial\psi}\end{aligned} \tag{3}$$

where

$$\begin{aligned}\delta_t(\tau_{:t};\psi_V) &= R_t - V(\tau_{:t};\psi_V)\\ R_t &= \sum_{i=0}^{k-1}[\gamma^i r_{t+i} + \gamma^k V(\tau_{:t+k};\psi_V)]\end{aligned} \tag{4}$$

defines the n-step temporal difference error advantage function δ_t, the discounted n-step bootstrapped return R_t with discount factor γ, k the number of remaining time steps in the current episode,

Table 2: **Hyperparameters**

	episode length	β_e	β_V	n_updates	bottleneck size
Two-armed bandit	40	$0.01 - 0.05$	$0.01 - 0.05$	$3e5$	8
Dynamic bandit	300	0.01	0.05	$1e5$	8
Stationary Tiger	$20 - 30$	0.3	0.1	$3e5$	4
Dynamic Tiger	$30 - 40$	0.3	0.1	$3e5$	4
Oracle bandit	6	$0.3 - 0.5$	$0.01 - 0.05$	$3e6$	16
Latent goal cart	30	$0.005 - 0.01$	0.01	$3e5$	8

and V the value function parametrized by ψ_V. The neural network policy is denoted as π and parametrized by ψ, and H_π is the entropy of the policy. Finally, β_V and β_e are hyperparemeters for controlling the relative weighting of value estimation loss and entropy regularization. The choice of hyperparameters for each task is summarized in Table. 2. The neural network parameters are trained via backpropagation through time using the Adam Optimizer with a learning rate of 5e-5.

Meta-RL with predictive modules The self-supervised predictive modules are formulated as a VAE. For the encoder q_ϕ we use an RNN with 256 hidden units and hyperbolic tangent activation functions. The output of the encoder RNN, i.e. the bottleneck layer, is treated as estimating the mean and the variance of the latents m_t, as standard in VAE. Therefore, the latent dimension is half of the bottleneck layer size. The decoders R_θ and T_θ are multi-layered perceptrons (MLPs) with one hidden layer of 32 units and ReLU activation functions. As shown in Fig. 1C, the encoder-decoder framework takes as input the current observation o_t, the previous action in one-hot format a_{t-1}, and the associated scalar reward r_t, and is set up to make prediction of the upcoming observations o_{t+1} and rewards r_{t+1}, conditioned on the trajectory as incurred by the policy network π_ψ described below. The entire VAE is trained to maximize the ELBO (Eq. 2) as derived in A.1. A coefficient of 0.01 is used for the relative contribution of the KL-term for training the VAE. We use the Adam Optimizer with a learning rate of 7e-5 to train the VAE using backpropagation through time.

The policy network π_ψ is parametrized as an MLP with one hidden layer of 32 units and hyperbolic tangent activation functions. Similar to the previous paragraph on RL^2, the policy network is trained with the Advantage Actor Critic algorithm to optimize the same loss function as described in Eq. 3. The choice of hyperparameters for each task is summarized in Table. 2. An Adam Optimizer with a learning rate of 5e-5 is used to optimize the policy network. Note although the policy loss depends on the parameters of the encoder q_ϕ, we do not backpropagate the policy loss gradient through the encoder as the goal of the encoder is to learn a belief b_t over the latent states predictive of the future such that the belief alone should be a sufficient representation for policy learning (similar to Zintgraf et al. [9]). Parts of the implementation code are based on [9] (under MIT license).

Model training The above meta-RL models (RL^2 and ours, meta-RL with predictive modules) are trained on internal GPU clusters (NVIDIA GeForce RTX 4090), which takes less than 1G GPU memory and between 10-48 hours for training per model, depending on the task. Compared to RL^2, training time of meta-RL with predictive modules increases by $\sim$ 30-40% with the overhead coming from two additional decoders and VAE training, whereas inference time is comparable.

Ablation study To pinpoint which algorithmic design choices drive enhanced representation learning in our proposed meta-RL with predictive modules, we performed two targeted ablations:

- (i) no KL: where we remove the VAE's KL regularization to test whether latent regularization is necessary to enforce compact representations.
- (ii) joint RL: where we allow the policy gradients from RL loss to jointly train the RNN encoder alongside the predictive loss—rather than training the encoder *solely* with the predictive loss—to assess whether policy gradients provide additional representational benefits.

A.4 State machine simulation

Following the procedure of state machine simulation introduced in Mikulik et al. [13], we consider whether a meta-RL system (RL^2 or our proposed approach, meta-RL with predictive modules) can both *simulate* and *be simulated by*, a Bayes-optimal agent for a given POMDP task.

To evaluate how well a state machine M *simulates* another machine N, a function ϕ is first learned to map the states S_N in N into the state space S_M of M. As enumeration over all possible trajectories are not practical if not possible, quality of a simulation is measured along trajectories sampled from some reference distribution. Given trajectories from a reference distribution, quality of the simulation is then measured by (i) the *state-transition dissimilarity* D_s, measured as the mean-squared error (MSE) between the embedded states $\phi(S_N)$ and the target states S_M, and (ii) the *output dissimilarity* D_o, measured as the difference in the expected return generated from the states S_N using the machine N and those generated from the states $\phi(S_N)$ using the machine M. If both the state-transition and output dissimilarities D_s and D_o are low/ negligible, then we establish that M simulates N. If both M simulates N and N simulates M, then we can say M and N are computationally equivalent, and their states S_N and S_M are equivalent.

In practice, the mapping function ϕ is implemented as an MLP with ReLU activations and three hidden layers of 64, 128, and 64 units, respectively. The MLP is trained with the Adam Optimizer with learning rate 0.001 and batch size 64. The training set is consisted of $300 - 1000$ trajectories depending on the variability of each task distribution, and the results reported are from another test set of $300 - 1000$ trajectories. Compared with Mikulik et al. [13], where the reference distribution is generated by the meta-RL agent, here we modify the procedure by generating the reference distribution using the Bayes-optimal agent, as this provides an even more stringent condition where the simulation is evaluated in the regime of Bayes-optimal solutions.

When evaluating how well a Bayes-optimal agent *simulates* a meta-RL system, we first train an MLP mapping meta-RL states into Bayes-optimal states by minimizing the MSE between the mapped states and the target Bayes-optimal states. After training, quality of the simulation is measured on a test set by evaluating the state-transition dissimilarity (D_s: metaRL$\rightarrow$Bayes) and the output dissimilarity (D_o: metaRL$\rightarrow$Bayes). If both D_s: metaRL$\rightarrow$Bayes and D_o: metaRL$\rightarrow$Bayes are low after training, then the Bayes-optimal agent *simulates* the meta-RL agent.

On the other hand, to evaluate how well a Bayes-optimal agent *is simulated by* a meta-RL one, an MLP is trained to map Bayes-optimal states into meta-RL states, and the state-transition dissimilarity is denoted D_s: Bayes$\rightarrow$metaRL and the output dissimilarity denoted D_o: Bayes$\rightarrow$metaRL. If both D_s: Bayes$\rightarrow$metaRL and D_o: Bayes$\rightarrow$metaRL are low after training, then the Bayes-optimal agent *is simulated by* the meta-RL agent.

If all the above four dissimilarity measures are low/ negligible, then we can say that the meta-RL agent and the Bayes-optimal agent are computationally equivalent and their representations are equivalent. Our results in this paper demonstrate that after training the proposed meta-RL with predictive coding modules can attain much lower dissimilarities than conventional RL^2, indicating that meta-RL with predictive modules can more closely approximate the Bayes-optimal belief states.

Note that before training, the state dissimilarity D_s can be low for the recurrent layers in all models, similar to what Mikulik et al. [13] reported and discussed—untrained RNNs may maintain a verbose representation of the history by embedding each trajectory into a unique hidden state, which can be subsequently mapped to minimally sufficient Bayes-optimal statistic with low error using expressive enough functions, like the MLPs used in the above state machine simulation procedure. This observation also highlights that using decoding alone may not provide a thorough assessment of representation equivalence, and we need to consider their structural and computational relevance when comparing representations.

A.5 Additional results

A.5.1 State space visualization

To understand the structure of the learned representations in meta-RL agents and qualitatively compare those to the Bayes-optimal belief states, visualization of example state spaces for the stationary and dynamic bandit tasks, the stationary and dynamic Tiger tasks, the oracle bandit tasks, and the latent goal cart tasks are presented in Figures. 8, 9, 10, and 11 respectively.

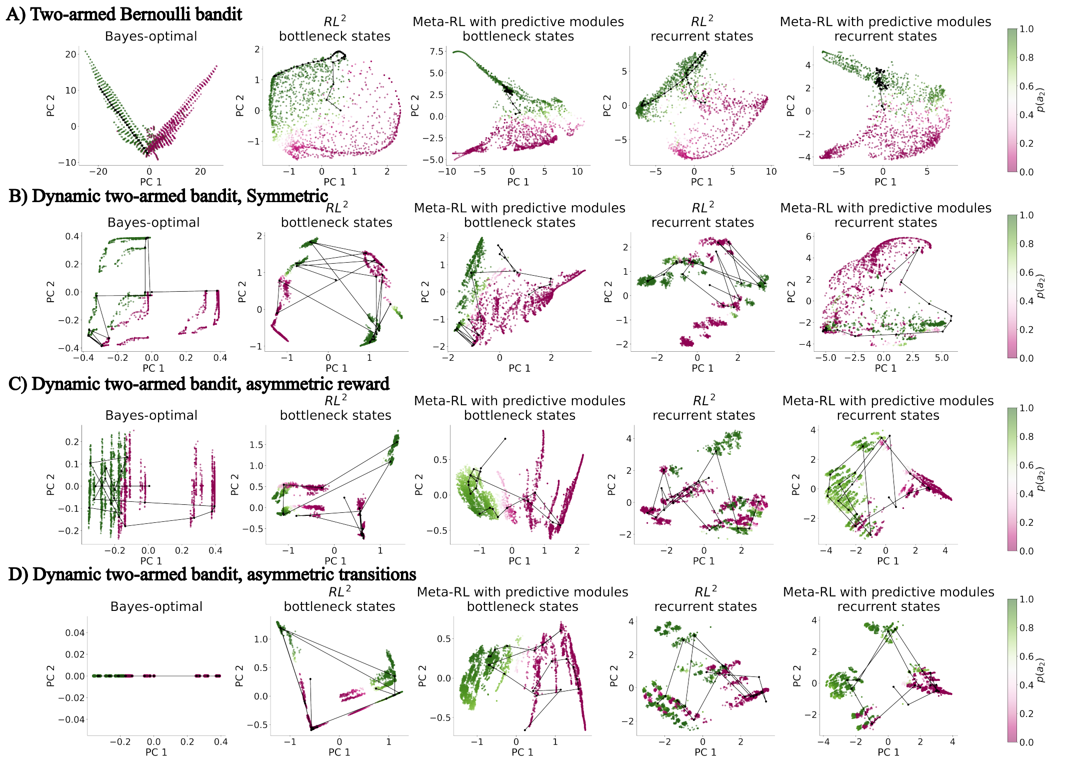

Figure 8: **Visualization of state space in the stationary and dynamic two-armed bandit tasks**. Example state space for A) Two-armed Bernoulli bandit task, B) Dynamic two-armed bandit task with symmetric reward and transition, C) Dynamic two-armed bandit task with asymmetric reward, and D) Dynamic two-armed bandit task with asymmetric transition. For each panel, the first two principal components are plotted. States are colored by the corresponding policy, i.e. probability of choosing a_2. Black curves show one example trajectory. For each task, from the left to the right are the state space of the Bayes-optimal agent (i.e. the belief states), the bottleneck layer in the RL^2 model, the bottleneck layer in the meta-RL with predictive modules, the recurrent layer in the RL^2 model, and the recurrent layer in the meta-RL with predictive modules, respectively.

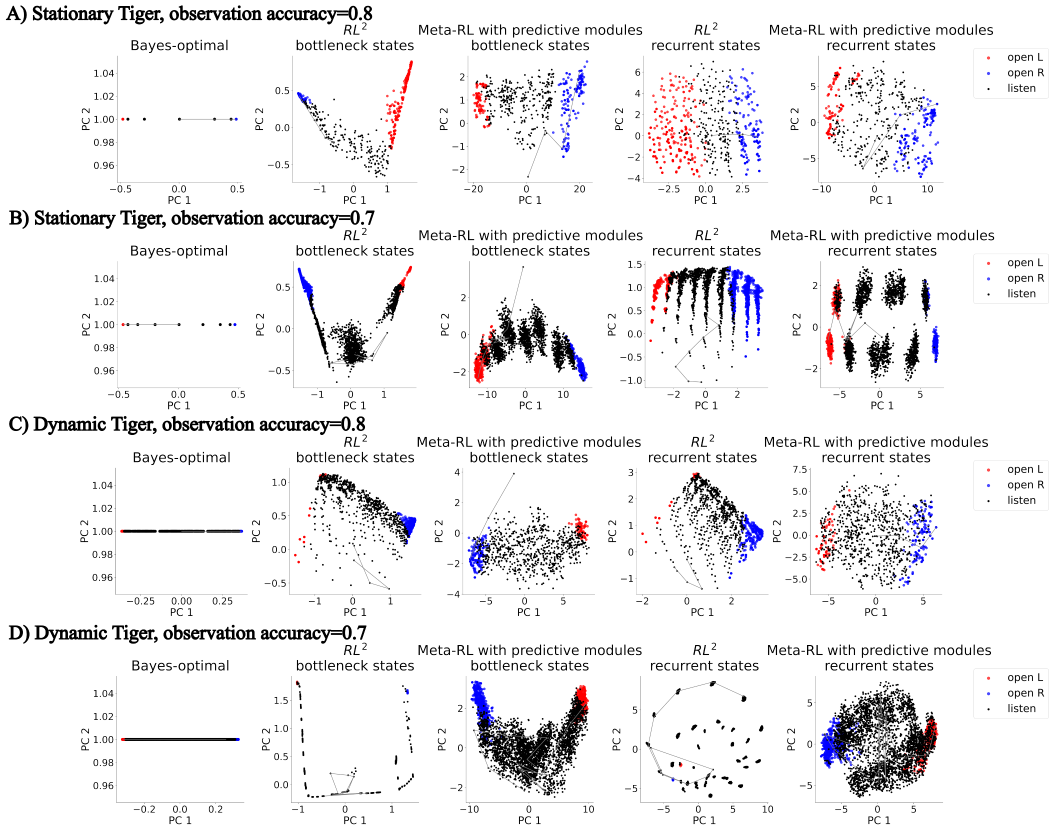

Figure 9: **Visualization of state space in the stationary and dynamic Tiger tasks**. Example state space for A) Stationary Tiger task with observation accuracy of 0.8, B) Stationary Tiger task with observation accuracy of 0.7, C) Dynamic Tiger task with observation accuracy of 0.8, and D) Dynamic Tiger task with observation accuracy of 0.7. For each panel, the first two principal components are plotted. States are colored by the corresponding policy, with black denoting choosing listen, red choosing to open the left door, and blue choosing to open the right door. Black curves show one example trajectory. For each task, from the left to the right are the state space of the Bayes-optimal agent (i.e. the belief states), the bottleneck layer in the RL^2 model, the bottleneck layer in the meta-RL with predictive modules, the recurrent layer in the RL^2 model, and the recurrent layer in the meta-RL with predictive modules, respectively.

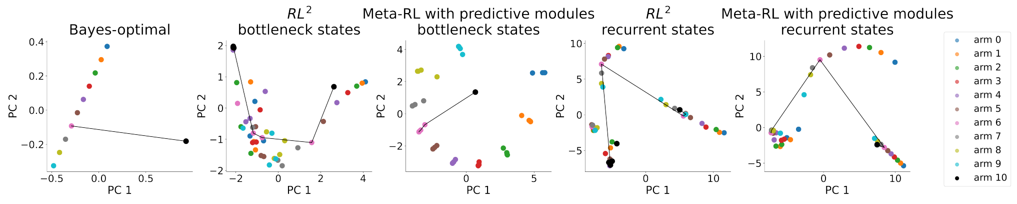

Figure 10: **Visualization of state space in the oracle bandit task**. Example state space for the oracle bandit task. For each panel, the first two principal components are plotted. States are colored by the corresponding policy, with black denoting choosing the oracle arm a_{11} and other colors denoting choosing one of the first ten arms a_{1-10}. Black curves show one example trajectory. From the left to the right are the state space of the Bayes-optimal agent (i.e. the belief states), the bottleneck layer in the RL^2 model, the bottleneck layer in the meta-RL with predictive modules, the recurrent layer in the RL^2 model, and the recurrent layer in the meta-RL with predictive modules, respectively.

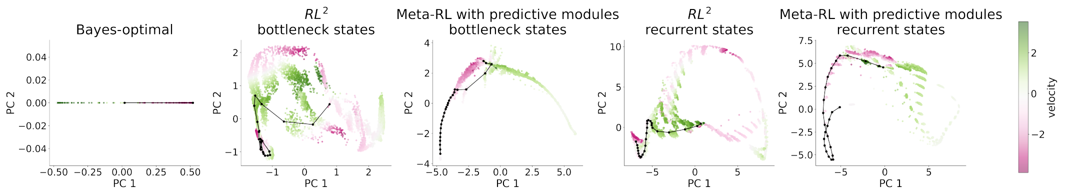

Figure 11: **Visualization of state space in the latent goal cart task**. Example state space for the oracle bandit task. For each panel, the first two principal components are plotted. States are colored by the corresponding policy, i.e. the velocity of the cart. Black curves show one example trajectory. From the left to the right are the state space of the Bayes-optimal agent (i.e. the belief states), the bottleneck layer in the RL^2 model, the bottleneck layer in the meta-RL with predictive modules, the recurrent layer in the RL^2 model, and the recurrent layer in the meta-RL with predictive modules, respectively.

A.5.2 Full state machine simulation results

Full results of state machine simulation analysis on the trained models of RL^2 and the proposed meta-RL with predictive modules (ours), together with models considered in the ablation studies—no KL and joint RL, are summarized in Fig. 12 and Tables 3, 4, 5, and 6.

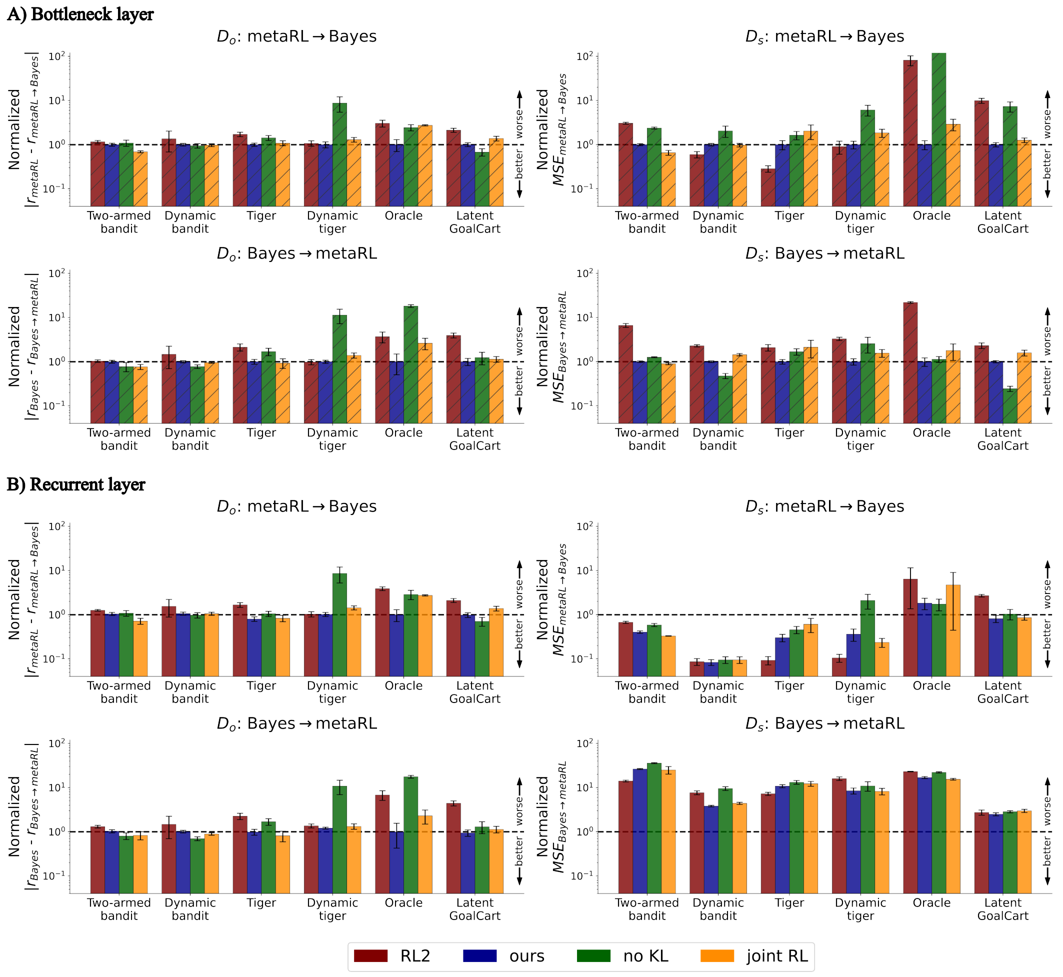

Figure 12: **Results of state machine simulation**. State and output dissimilarities (D_o and D_s) for both mapping directions (meta-RL→Bayes and Bayes→meta-RL) across all tasks considered, using A) the bottleneck layer and B) the recurrent layer of the fully-trained models of RL^2 (red), meta-RL with predictive modules (ours, blue), no KL (where KL regularization is ablated for VAE training, green), and joint RL (where the policy gradient of RL loss is used to jointly train the RNN encoder with the predictive loss, orange). To pool across different variants of the same task type, all dissimilarity measures are normalized by the mean of the corresponding dissimilarities of the bottleneck layer in the meta-RL with predictive modules models (denoted as *ours* in the legend.) In other words, when normalized with the mean bottleneck dissimilarity of our model, values significantly greater than 1 indicate that the representation is worse than the bottleneck layer in our model (i.e. deviating more from the Bayes-optimal belief states). In contrast, values significantly smaller than 1 indicate the representation is better than the bottleneck layer in our model (i.e. more closely approximating the Bayes-optimal belief states). Error bars denote s.e.m. across at least five random seeds per model type.

Table 3: **State machine simulation results for the stationary and dynamic bandit tasks**: State and output dissimilarities of the fully-trained models for both mapping directions. M for meta-RL, B for Bayes-optimal solutions. Values indicate mean±s.t.d. Bold indicates the best dissimilarities among the trained models and better than untrained models ($p < 0.05$).

Two-armed Bernoulli bandit (unit: 1e-2)								
	RL^2	ours	no KL	joint RL	RL^2	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M\to B}$	1.52±0.36	1.31±0.37	1.44±0.4	**0.91±0.05**	1.65±0.24	1.37±0.38	1.42±0.36	**0.94±0.15**
$D_s^{M\to B}$	5.41±0.96	1.77±0.33	4.21±0.38	1.16±0.15	1.19±0.23	0.71±0.16	1.04±0.14	0.58±0.01
$D_o^{B\to M}$	**1.19±0.25**	**1.15±0.31**	**0.89±0.36**	**0.88±0.12**	**1.51±0.35**	**1.19±0.34**	**0.93±0.27**	**0.95±0.2**
$D_s^{B\to M}$	5.38±1.81	**0.81±0.11**	1.02±0.04	**0.73±0.04**	11.43±1.83	21.37±2.12	29.05±1.43	20.33±3.96

Dynamic two-armed bandit: symmetric reward and transition (unit: 1e-2)								
	RL^2	ours	no KL	joint RL	RL^2	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M\to B}$	**2.09±3.95**	**1.17±0.18**	**0.86±0.48**	**1.06±0.1**	**2.18±3.92**	**1.25±0.21**	**0.86±0.43**	**1.08±0.12**
$D_s^{M\to B}$	0.19±0.02	0.21±0.05	0.49±0.36	0.22±0.1	0.03±0.01	0.03±0.01	0.03±0.02	0.04±0.01
$D_o^{B\to M}$	**2.06±3.81**	**0.98±0.36**	**0.79±0.33**	**0.89±0.19**	**2.18±3.75**	**1.05±0.44**	**0.72±0.3**	**0.82±0.3**
$D_s^{B\to M}$	2.38±0.6	1.09±0.22	**0.45±0.13**	1.43±0.23	10.26±2.01	4.86±0.44	10.25±2.37	6.11±0.69

Dynamic two-armed bandit: asymmetric reward (unit: 1e-2)								
	RL^2	ours	no KL	joint RL	RL^2	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M\to B}$	**0.8±0.28**	**0.86±0.27**	**0.76±0.19**	**0.82±0.24**	**1.01±0.14**	**1.22±0.34**	**0.59±0.13**	**1.04±0.34**
$D_s^{M\to B}$	0.34±0.06	0.93±0.11	0.4±0.17	1.04±0.16	0.04±0.01	0.05±0.0	0.03±0.01	0.04±0.01
$D_o^{B\to M}$	**1.15±0.27**	**1.39±0.03**	**0.99±0.37**	**1.37±0.16**	**1.03±0.31**	**1.37±0.03**	**0.97±0.34**	**1.28±0.35**
$D_s^{B\to M}$	3.49±0.51	**1.45±0.1**	**0.99±0.44**	1.98±0.33	10.47±3.6	4.91±0.75	18.2±5.4	5.56±0.6

Dynamic two-armed bandit: asymmetric transition (unit: 1e-2)								
	RL^2	ours	no KL	joint RL	RL^2	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M\to B}$	**0.85±0.23**	**0.93±0.33**	**1.11±0.47**	**0.98±0.39**	**1.22±0.33**	**0.78±0.29**	**1.39±0.56**	**0.95±0.4**
$D_s^{M\to B}$	0.04±0.02	0.33±0.11	1.1±1.06	0.24±0.09	0.01±0.0	0.01±0.01	0.03±0.01	0.02±0.01
$D_o^{B\to M}$	**1.52±0.1**	**1.7±0.17**	**1.36±0.46**	**1.67±0.12**	**1.21±0.39**	**1.64±0.19**	**1.12±0.5**	**1.55±0.16**
$D_s^{B\to M}$	5.14±0.75	2.15±0.45	**0.7±0.39**	3.56±1.08	9.0±1.06	7.41±0.91	14.35±5.85	8.16±1.69

Table 4: **State machine simulation results for the stationary and dynamic Tiger tasks**.

Stationary Tiger: observation accuracy 0.8 (unit: 1e-2)

	RL^2	ours	no KL	joint RL	RL^2	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M\to B}$	**3.33±0.91**	**2.57±1.01**	**2.93±1.09**	**2.97±0.87**	**3.05±0.64**	**2.07±1.03**	**2.13±0.37**	**2.6±1.22**
$D_s^{M\to B}$	0.08±0.05	0.31±0.33	0.46±0.3	0.67±0.87	0.03±0.03	0.06±0.04	0.11±0.04	0.23±0.27
$D_o^{B\to M}$	**3.03±1.54**	**2.26±0.88**	**2.98±1.7**	**1.88±1.24**	**4.15±1.72**	**1.85±1.22**	**2.98±1.7**	**1.34±0.92**
$D_s^{B\to M}$	8.04±4.38	**2.92±1.84**	4.1±1.3	9.07±10.54	24.95±7.07	28.31±8.99	35.61±8.74	35.95±12.55

Stationary Tiger: observation accuracy 0.7 (unit: 1e-2)

	RL^2	ours	no KL	joint RL	RL^2	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M\to B}$	3.25±1.13	**1.57±0.49**	2.68±1.0	**1.58±0.74**	3.27±1.14	**1.24±0.62**	2.0±0.9	**1.02±0.36**
$D_s^{M\to B}$	0.09±0.05	0.3±0.27	0.53±0.39	0.58±0.44	0.03±0.01	0.13±0.09	0.17±0.1	0.15±0.08
$D_o^{B\to M}$	**3.06±1.54**	**1.09±0.73**	**2.25±1.34**	**1.13±0.85**	**2.86±1.67**	**1.33±0.76**	**2.25±1.1**	**1.13±0.85**
$D_s^{B\to M}$	4.51±0.96	**2.97±0.78**	5.81±3.41	**3.46±1.53**	18.18±1.76	36.56±11.68	42.07±15.13	36.21±14.37

Dynamic Tiger: observation accuracy 0.8 (unit: 1e-2)

	RL^2	ours	no KL	joint RL	RL^2	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M\to B}$	**2.19±0.96**	**2.39±0.41**	**3.5±1.35**	**3.48±1.18**	**2.25±0.95**	**2.16±0.7**	**3.1±0.22**	**3.93±1.03**
$D_s^{M\to B}$	0.04±0.01	0.1±0.05	0.31±0.17	0.29±0.06	0.01±0.0	0.04±0.04	0.12±0.09	0.02±0.02
$D_o^{B\to M}$	**2.01±0.93**	**1.99±0.45**	3.48±0.6	3.28±1.44	3.28±0.59	**2.34±0.42**	3.21±0.76	3.28±1.26
$D_s^{B\to M}$	6.27±1.15	**1.78±0.98**	**2.9±1.76**	**2.03±0.88**	29.5±2.07	14.39±4.93	24.2±10.58	11.82±5.04

Dynamic Tiger: observation accuracy 0.7 (unit: 1e-2)

	RL^2	ours	no KL	joint RL	RL^2	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M\to B}$	**2.51±0.66**	**2.05±1.28**	27.92±21.09	**2.33±1.01**	**2.3±0.67**	**2.37±0.53**	27.75±21.84	**2.51±0.95**
$D_s^{M\to B}$	0.39±0.23	0.28±0.17	2.23±1.53	0.2±0.18	0.03±0.02	0.08±0.06	0.74±0.77	0.07±0.06
$D_o^{B\to M}$	**1.73±0.32**	**1.8±0.47**	32.11±22.34	**2.05±0.57**	**1.95±0.31**	**2.27±0.17**	30.65±20.94	**1.8±0.65**
$D_s^{B\to M}$	7.1±1.89	**2.3±0.72**	7.37±8.41	4.62±2.85	35.44±12.21	20.65±10.63	21.05±19.62	22.64±12.53

Table 5: **State machine simulation results for the oracle bandit tasks**.

Oracle bandit (unit: 1e-2)

	RL^2	ours	no KL	joint RL	RL^2	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M\to B}$	17.21±9.73	**5.64±4.5**	13.79±4.44	15.48±1.02	21.96±6.65	**5.64±4.5**	16.26±7.76	15.48±1.02
$D_s^{M\to B}$	6.45±5.14	**0.08±0.05**	24.4±6.94	0.23±0.14	0.51±1.27	0.14±0.11	0.14±0.08	0.37±0.68
$D_o^{B\to M}$	12.01±10.45	**3.26±4.26**	59.34±8.43	8.57±5.0	22.18±17.23	**3.24±4.9**	57.34±8.39	7.49±5.26
$D_s^{B\to M}$	51.29±6.73	**2.33±1.34**	2.64±0.82	4.18±3.38	53.72±2.63	39.33±5.68	51.5±3.42	36.02±3.26

Table 6: **State machine simulation results for the latent goal cart tasks**.

Latent goal cart (unit: 1e-2)

	RL^2	ours	no KL	joint RL	RL^2	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M\to B}$	33.68±11.84	**15.83±3.80**	**10.73±4.94**	21.86±8.88	33.37±11.44	**15.56±4.61**	**11.20±6.19**	21.93±9.05
$D_s^{M\to B}$	2.35±1.02	**0.24±0.07**	1.75±1.10	**0.30±0.12**	0.64±0.14	**0.19±0.09**	**0.25±0.16**	**0.20±0.08**
$D_o^{B\to M}$	31.67±13.36	**8.01±3.62**	**9.96±7.73**	**9.14±4.47**	35.42±15.36	**7.56±3.07**	**10.39±7.69**	**9.08±5.10**
$D_s^{B\to M}$	30.51±14.64	13.16±1.84	**3.22±1.09**	20.90±10.58	36.02±16.02	32.78±6.28	37.38±5.29	39.16±10.85

A.5.3 Sensitivity analysis

To evaluate whether representational equivalence between Bayes-optimal solutions and meta-RL with predictive modules is sensitive to the choice of bottleneck dimensions, using the oracle bandit task as an example, we performed systematic evaluation of state machine simulation while varying the bottleneck dimension ranging from 1 to 32. As shown in Fig. 13, when the dimension of the bottleneck is greater than task complexity (i.e. the belief state dimension for the task, e.g. > 2 for the oracle bandit task), the bottleneck of the meta-RL with predictive modules achieve similarly low dissimilarities, indicating that regularization helps to maintain interpretable belief representation even when the bottleneck is more than expressive enough. In contrast, when the dimension of the bottleneck is smaller than task complexity, both performance (suboptimal return) and representational equivalence (high dissimilarity) significantly degrade, indicating that the bottleneck dimension should be selected to be at least surpassing the task complexity in order to learn the task effectively.

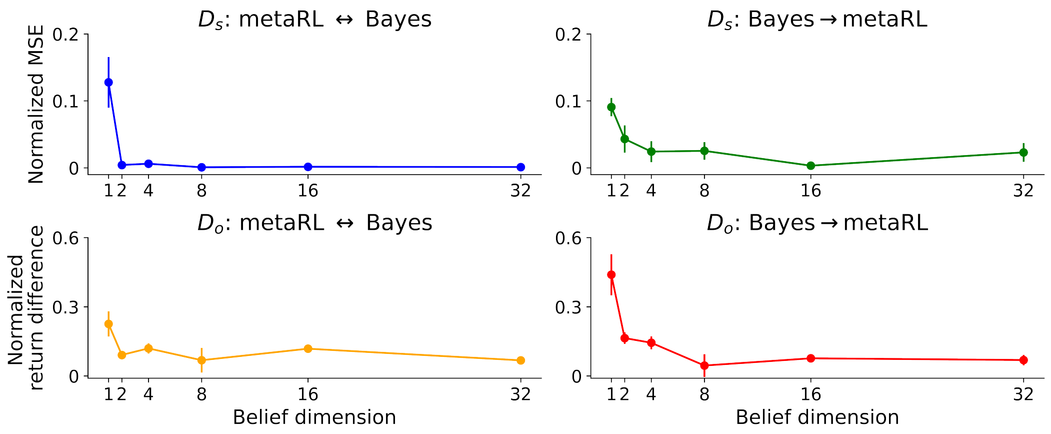

Figure 13: **Sensitivity analysis**. State machine simulation results of the bottleneck layers in meta-RL with predictive modules in the Oracle bandit task, as the belief dimension varies from 1 to 32. When the belief dimension is smaller than task complexity (e.g. $<= 2$), the models present with high dissimilarities in all four metrics. In contrast, as long as the belief dimension is greater than task complexity, all four dissimilarities remain consistently low even when the belief dimension is large, indicating that regularization is useful for maintaining compact, interpretable belief representation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claim in this study is that augmenting meta-RL agents with predictive coding modules can learn representations that better approximate Bayes-optimal belief states in partially observable environments than conventional meta-RL models. This claim is examined both qualitatively and quantitatively across several POMDP tasks and presented in the Results section (Section 5) and appendix (A.5.2).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The scope and limitations of this study is discussed in Section 6. As the scope of the study is to investigate whether meta-RL with predictive modules can learn representations with high equivalence to Bayes-optimal states, we focus on tasks where Bayes-optimal solutions are derivable. Additional work is needed to further evaluate tasks with larger state space. We also discuss other limitations in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The main theoretical result involved in the paper is the derivation of the evidence lower bound used for optimizing the predictive variational autoencoder, which is presented in Appendix A.1.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Models and experiments are introduced in Sections 3 and 4. Implementation details of the tasks, model architectures, and the analysis procedures considered in the study are further presented in Appendix A.2, A.3, and A.4, respectively.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Open access to the code for training the models and running experimental analysis is publicly available at https://github.com/walkerlab/metaRL-predictive-representation. We also provided detailed implementation details in the Appendix A.2, A.3, and A.4 to reproduce all the experimental results.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Models and experiments are introduced in Sections 3 and 4. All relevant details of model training, evaluation, and analysis are presented in A.2, A.3, and A.4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The quantitative results using state machine analysis are presented with error bars indicating the standard errors of the mean across different models with different random seeds. Statistical significance tests comparing the difference between difference models are also performed for the analysis, with results indicated in the figures.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Information on the computer resources is provided in Appendix A.3, including compute worker type, memory, and time of training.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The study in this paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This study is foundational research and not tied to particular applications. But we have a paragraph discussing how our work is broadly contributing to model interpretability and the connection between neuroscience and AI research.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The source codes that parts of our implementation is based on are cited and their license mentioned and respected.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets, but we have plans to release the source code once the paper is accepted.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs are used only for writing, editing, formatting, and implementing standard analysis. They are not used for the core methodology.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.