Ph.D. Defense

Meta-Reinforcement Learning with Imaginary Tasks

Suyoung Lee

Advisor: Youngchul Sung

Smart Information Systems Research Lab (SISReL)
School of Electrical Engineering, KAIST

Introduction

- Deep reinforcement learning (RL) faces critical challenges, including the issues of overfitting and a limited generalization capability when encountering unseen test tasks.
- Meta-reinforcement learning (meta-RL) aims to solve these problems by training agents across a variety of tasks, enabling them to learn to infer the underlying dynamics of new tasks and to rapidly adapt their policies accordingly.
  - Nonetheless, conventional meta-RL methods rely on a restricted training task distribution, which limits adaptability to out-of-distribution (OOD) test tasks.
- This thesis introduces two meta-RL algorithms that train the policy on imaginary tasks generated by a learned dynamics model, enhancing the ability to generalize to unseen distribution of tasks.

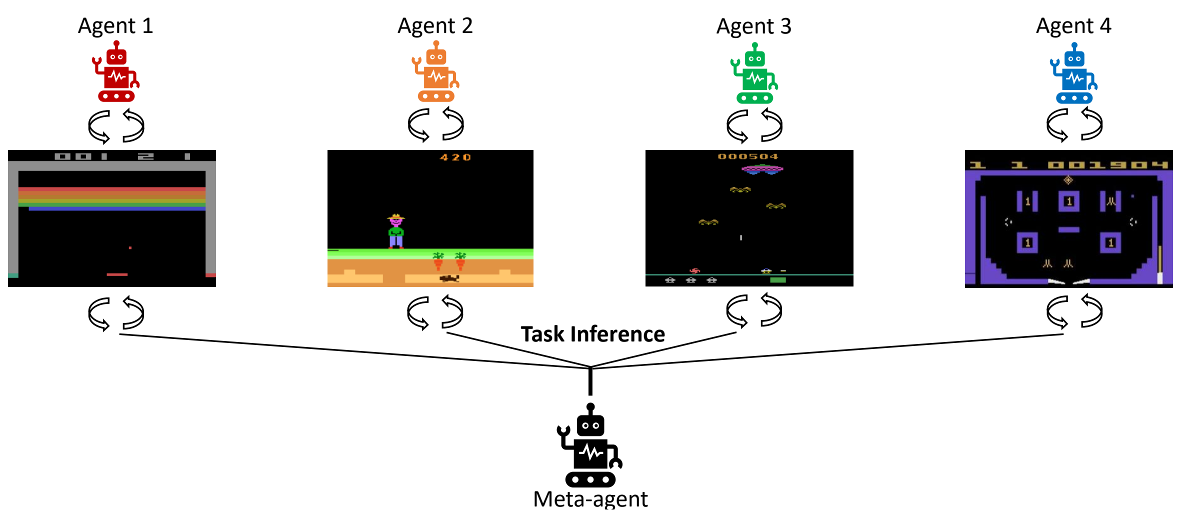


Figure 1: Comparison of standard RL and meta-RL.

Outline and Contributions

Meta-Reinforcement Learning with Imaginary Tasks

1. Introduction to Meta-Reinforcement Learning: A running example on Gridworld

2. Latent Dynamics Mixture (LDM)[1]
   - Addresses parametric task variability (Gridworld, MuJoCo).
   - Trains a policy on imaginary tasks generated from interpolations of latent beliefs.
   - Enhances generalization to unseen out-of-distribution (OOD) tasks with simple parametric variations.

3. Subtask Decomposition and Virtual Training (SDVT)[2]
   - Addresses non-parametric task variability (Meta-World).
   - Learns to decompose each non-parametric task into a set of shared elementary subtasks.
   - Trains a policy on imaginary tasks composed of imaginary compositions of learned elementary subtasks.

[1] **Suyoung Lee** and Sae-Young Chung, "Improving Generalization in Meta-RL with Imaginary Tasks from Latent Dynamics Mixture," NeurIPS 2021.
[2] **Suyoung Lee**, Myungsik Cho, and Youngchul Sung, "Parametrizing Non-Parametric Meta-Reinforcement Learning Tasks via Subtask Decomposition," NeurIPS 2023.

Meta-Reinforcement Learning

- Assume a finite task space: $\mathcal{M} = \left\{M^{(1)}, \ldots, M^{(m)}\right\}$.
- Meta-reinforcement learning (Meta-RL) aims to learn to adapt to a set of Markov Decision Processes (MDPs) with diverse reward and transition dynamics.

$$M^{(i)} = (\mathcal{S}, \mathcal{A}, R^{(i)}, T^{(i)}, T_0^{(i)}, \gamma, H). \tag{1}$$

- At the start of a meta-episode, an MDP $M^{(i)} \sim \mathbb{P}(\mathcal{M})$ is sampled from the pool of tasks $\mathcal{M}$.
- Each MDP $M^{(i)}$ has unique dynamics:
  - Reward dynamics: $R^{(i)}\left(r_{t+1} \mid s_t, a_t, s_{t+1}\right)$ or $r_{t+1} = R^{(i)}\left(s_t, a_t, s_{t+1}\right)$ (deterministic case)
  - (State) Transition dynamics: $T^{(i)}\left(s_{t+1} \mid s_t, a_t\right)$
  - Initial state distribution: $T_0^{(i)}\left(s_0\right)$
- A meta-episode consists of N rollout episodes, each with a horizon H.
  - A new task $M^{(i)}$ is sampled only at the start of each meta-episode (every $H^+ := NH$ steps).
  - After each rollout episode, the state is reset to an initial state (every H steps).

$$s_{kH} \sim T_0^{(i)}(\cdot), \quad k = 0, 1, \ldots, N-1. \tag{2}$$

Meta-Reinforcement Learning

- A belief is defined as a posterior distribution over the reward and state transition functions given the current meta-episode's trajectory $\tau_{:t} = (s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{t-1}, a_{t-1}, r_t, s_t)$.

$$b_t\left(R^{(i)}, T^{(i)}\right) := \mathbb{P}_{R,T}\left(R^{(i)}, T^{(i)} \mid \tau_{:t}\right), \quad i = 1, \dots, m. \tag{3}$$

- The optimization of meta-RL typically consists of two stages.
  - Task Inference: Learn to infer the current task by constructing a belief.
  - Policy Optimization: Learn the optimal policy conditioned on the inferred belief.

- The objective is to optimize the belief encoder $b_t = q_\phi(\tau_{:t})$ and the policy $\pi_\psi\left(a_t \mid s_t, b_t\right)$ that maximize the expected return across all MDPs.

$$J\left(\phi, \psi\right) = \mathbb{E}_{M^{(i)} \sim \mathbb{P}(\mathcal{M})}\left[\mathbb{E}_{T_0^{(i)}, T^{(i)}, \pi}\left[\sum_{t=0}^{H^+-1} \gamma^t R^{(i)}(s_t, a_t, s_{t+1})\right]\right]. \tag{4}$$

Running Example

- Gridworld navigation task (3 × 3) without obstacles.

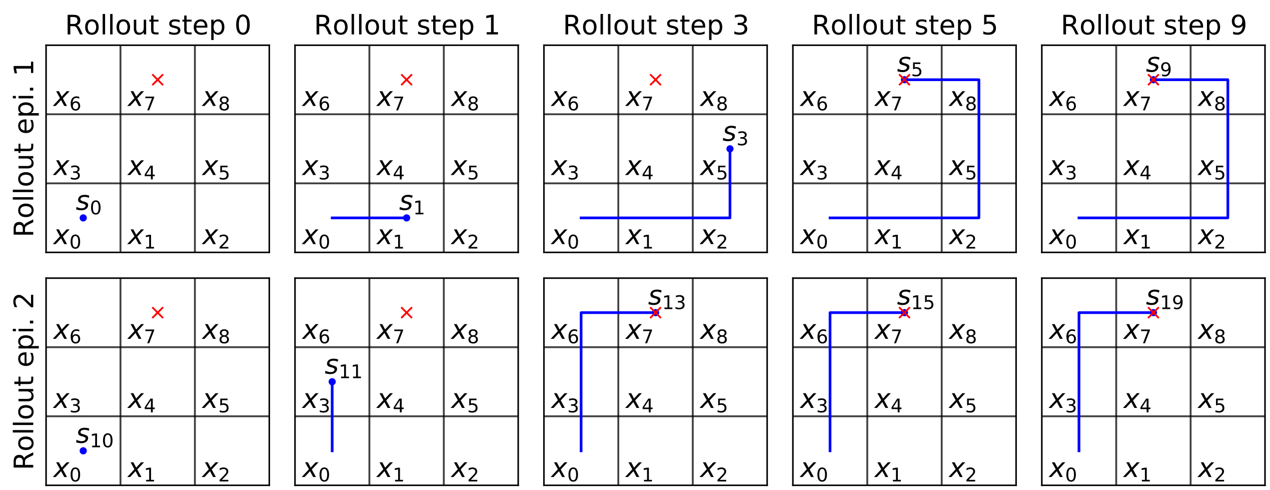


Figure 2: 3×3 Gridworld navigation task $M^{(7)}$ with an unknown goal position ×.

  - $\mathcal{M} = \{M^{(1)}, \ldots, M^{(8)}\}$.
  - $\mathcal{S} = \{x_0, \ldots x_8\}$ and $\mathcal{A} = \{$'up,' 'down,' 'left,' 'right,' 'stay'$\}$.
  - All $M^{(i)}$'s share the same standard Gridworld transition $T^{(i)} = T$.
  - All $M^{(i)}$'s start from the same initial state x_0. i.e., $T_0^{(i)}(x_0) = T_0(x_0) = 1$.
  - Each task $M^{(i)}$ assigns a reward of 1 only when the agent reaches x_i.

$$R^{(i)}(s_t, a_t, s_{t+1}) = \begin{cases} 1 & \text{if } s_{t+1} = x_i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

  - $N = 2$ rollout episodes, $H = 10$ steps.

Running Example

- Meta-episode 1: Suppose $M^{(7)}$ is sampled for the first meta-episode.

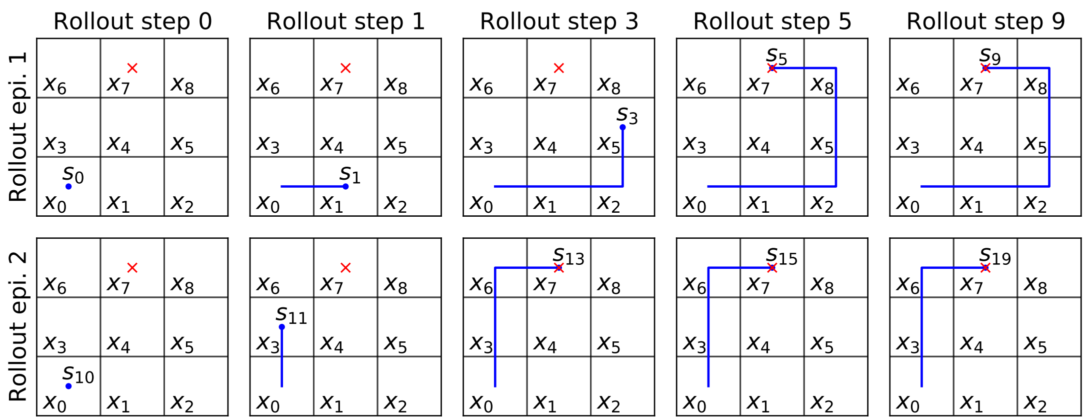


Figure 3: Example trajectory in meta-episode 1 within $M^{(7)}$.

- Meta-episode 2: After $H^+ = NH = 2 \times 10$ steps, a new task $M^{(5)}$ is sampled.

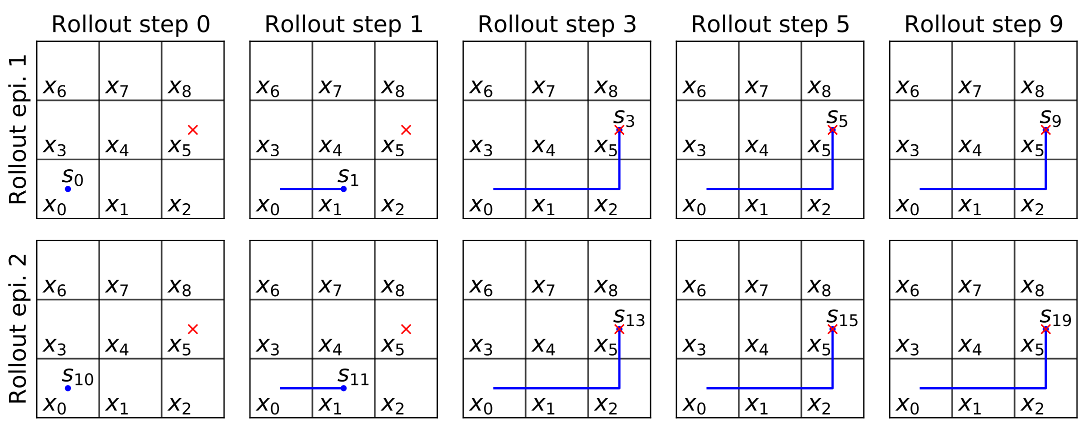


Figure 4: Example trajectory in meta-episode 2 within $M^{(5)}$.

Belief Update Example

- Belief Update: The process of refining the agent's understanding of the current task using observed transitions and rewards.

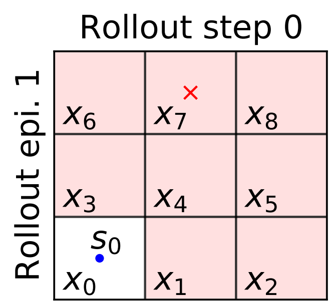


Figure 5: Initial belief state in a meta-episode on task $M^{(7)}$, $t = 0$.

- Assume that we are inferring the current task's information as a posterior belief given the current meta-episode's trajectory $\tau_{:t} = (s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{t-1}, a_{t-1}, r_t, s_t)$.

$$b_t\left(R^{(i)}\right) := \mathbb{P}_R\left(R^{(i)} \mid \tau_{:t}\right) \quad \text{for } i = 1, \ldots, 8. \tag{6}$$

- For example, assume that the current meta-episode is on the task $M^{(7)}$.
- $(t = 0)$ We start with a uniform prior, which we have to learn as well to match $\mathbb{P}(\mathcal{M})$.

$$\left\{b_0\left(R^{(i)}\right)\right\}_{i=1}^{8} = \left\{\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}\right\}. \tag{7}$$

Belief Update Example

- Belief update example of a perfectly trained meta-RL agent.

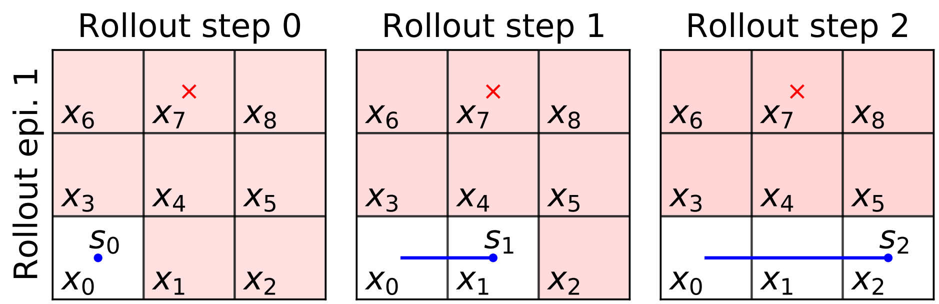


Figure 6: A meta-episode on task $M^{(7)}$. The first rollout episode until $t = 2$.

- $(t = 1)$ Assume we take action $a_0 =$ 'right' at s_0. We observe $s_1 = x_1$ and since the current meta-episode's task is $M^{(7)}$, we observe $r_1 = 0$, implying $\mathbb{P}_R(R^{(1)}|\tau_{:1}) = 0$.

$$\left\{b_1\left(R^{(i)}\right)\right\}_{i=1}^{8} = \left\{0, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}\right\}. \tag{8}$$

- $(t = 2)$ Continuing with action $a_1 =$ 'right' at s_1, observing $s_2 = x_2$ and $r_2 = 0$ refines the belief further.

$$\left\{b_2\left(R^{(i)}\right)\right\}_{i=1}^{8} = \left\{0, 0, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right\}. \tag{9}$$

Belief Update Example

- Belief update example of a perfectly trained meta-RL agent.

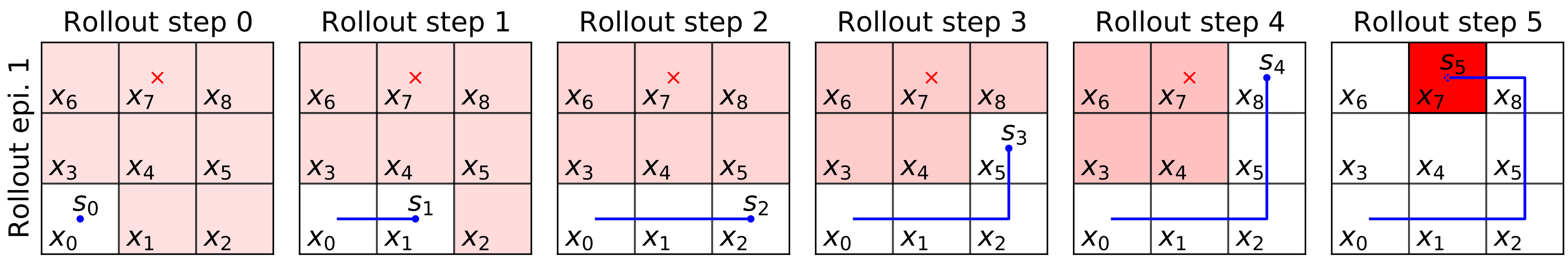


Figure 7: A meta-episode on task $M^{(7)}$. The first rollout episode until $t = 5$.

- $(t = 4)$ Taking action a_3 = 'up' at s_3 leads to $s_4 = x_8$ and $r_4 = 0$, refining the belief.

$$\left\{b_4\left(R^{(i)}\right)\right\}_{i=1}^{8} = \left\{0, 0, \frac{1}{4}, \frac{1}{4}, 0, \frac{1}{4}, \frac{1}{4}, 0\right\}. \tag{10}$$

- $(t = 5)$ Finally, the action a_4 = 'left' at s_4 leads to the goal, observing $s_5 = x_7$ and $r_5 = 1$.

$$\left\{b_5\left(R^{(i)}\right)\right\}_{i=1}^{8} = \{0, 0, 0, 0, 0, 0, 1, 0\}. \tag{11}$$

- We confirm that the current task of this meta-episode is certainly $M^{(7)}$.
- This conclusive belief remains for the rest of the meta-episode until a new task is sampled.

Belief Update Example

- Belief update example of a perfectly trained meta-RL agent.

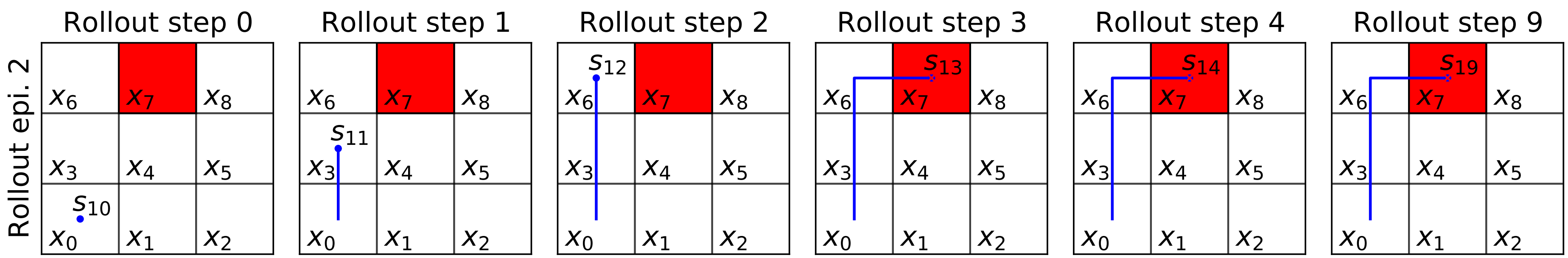


Figure 8: A meta-episode on task $M^{(7)}$. The second rollout episode.

- After the end of the first rollout episode ($H = 10$ steps), the state is reset to the initial state $s_H = x_0$.

- However, since the agent is in the same meta-episode, it is still in $M^{(7)}$, keeping the belief.

$$\left\{b_t\left(R^{(i)}\right)\right\}_{i=1}^{8} = \{0,0,0,0,0,0,1,0\}\,, \quad \text{for } 5 \leq t < H^+. \tag{12}$$

- The policy can exploit the belief inferred from the first rollout episode.

- After the meta-episode terminates (i.e., $H^+ = 20$ steps), a new task is sampled for the next-meta-episode. Also, the belief is reset to the prior b_0.

Bayes-Adaptive Meta-RL

- To learn $b_t\left(R^{(i)}, T^{(i)}\right) := \mathbb{P}_{R,T}\left(R^{(i)}, T^{(i)} \mid \tau_{:t}\right)$, we need to learn the prior and posterior update.
  - Prior: $b_0\left(R^{(i)}, T^{(i)}\right) := \mathbb{P}_{R,T}\left(R^{(i)}, T^{(i)} \mid s_0\right)$.
  - Posterior update upon taking a_t from s_t and observing s_{t+1} and r_{t+1} for all $t = 0, \ldots, H^+ - 1$.

$$b_{t+1}\left(R^{(i)}, T^{(i)}\right) = \frac{\mathbb{P}\left(s_{t+1}, r_{t+1} \mid s_t, a_t, R^{(i)}, T^{(i)}\right)}{\sum_{k=1}^{m} \mathbb{P}\left(s_{t+1}, r_{t+1} \mid s_t, a_t, R^{(k)}, T^{(k)}\right) b_t\left(R^{(k)}, T^{(k)}\right)} b_t\left(R^{(i)}, T^{(i)}\right), \tag{13}$$

where the likelihood $\mathbb{P}(s_{t+1}, r_{t+1} \mid s_t, a_t, R^{(i)}, T^{(i)}) = R^{(i)}(r_{t+1} \mid s_t, a_t, s_{t+1}) T^{(i)}(s_{t+1} \mid s_t, a_t)$.

Details in the Appendix (Eq. 27)

- The Bayes-adaptive policy $\pi\left(a_t \mid s_t, b_t\right)$ is trained to maximize the return conditioned on the belief.
- The posterior update requires a tractable task space and knowledge of the reward and transition functions.

Motivation

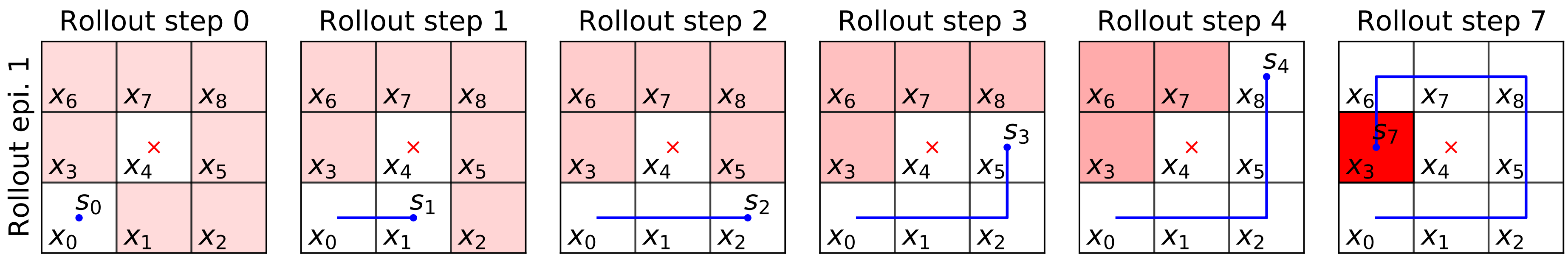


Figure 9: An example of test trajectory that is trained on $\mathcal{M}_{\text{train}} = \{M^{(i)}\}_{i=1}^{8} - M^{(4)}$ and tested on $\mathcal{M}_{\text{test}} = M^{(4)}$.

- Traditional meta-RL methods operate under the assumption that training and test tasks are drawn from the same distribution.

$$\mathcal{M} = \mathcal{M}_{\text{train}} = \mathcal{M}_{\text{test}}.$$

- This research is prompted by the following motivating question.

What if we evaluate the agent on out-of-distribution (OOD) test tasks?

$$\mathcal{M} = \mathcal{M}_{\text{train}} \cup \mathcal{M}_{\text{test}} \quad \text{and} \quad \mathcal{M}_{\text{train}} \cap \mathcal{M}_{\text{test}} = \emptyset.$$

Is it possible for the agent to effectively explore and exploit these previously unseen tasks?

Improving Generalization in Meta-RL with Imaginary Tasks from Latent Dynamics Mixture

Suyoung Lee and Sae-Young Chung

Presented at NeurIPS 2021

Introduction

- Similar to the Gridworld example, most conventional meta-RL methods assume the same distribution of training and test tasks: $(\mathcal{M} = \mathcal{M}_{\text{train}} = \mathcal{M}_{\text{test}})$.
- This research delves into OOD scenarios where the training and test tasks are made completely disjoint: $\mathcal{M} = \mathcal{M}_{\text{train}} \cup \mathcal{M}_{\text{test}}$ and $\mathcal{M}_{\text{train}} \cap \mathcal{M}_{\text{test}} = \emptyset$.

Table 1: Set of training and test parameters for OOD MuJoCo tasks. Here, $k \in \{0, 1, 2, 3\}$.

	Ant-direction	Ant-goal		Half-cheetah-velocity
	θ	r	θ	v
$\mathcal{M}_{\text{train}}$	$90^\circ \times k$	$[0.0, 1.0) \cup [2.5, 3.0)$	$[0^\circ, 360^\circ)$	$[0.0, 0.5) \cup [3.0, 3.5)$
$\mathcal{M}_{\text{test}}$	$90^\circ \times k + 45^\circ$	$[1.0, 2.5)$	$[0^\circ, 360^\circ)$	$[0.5, 3.0)$

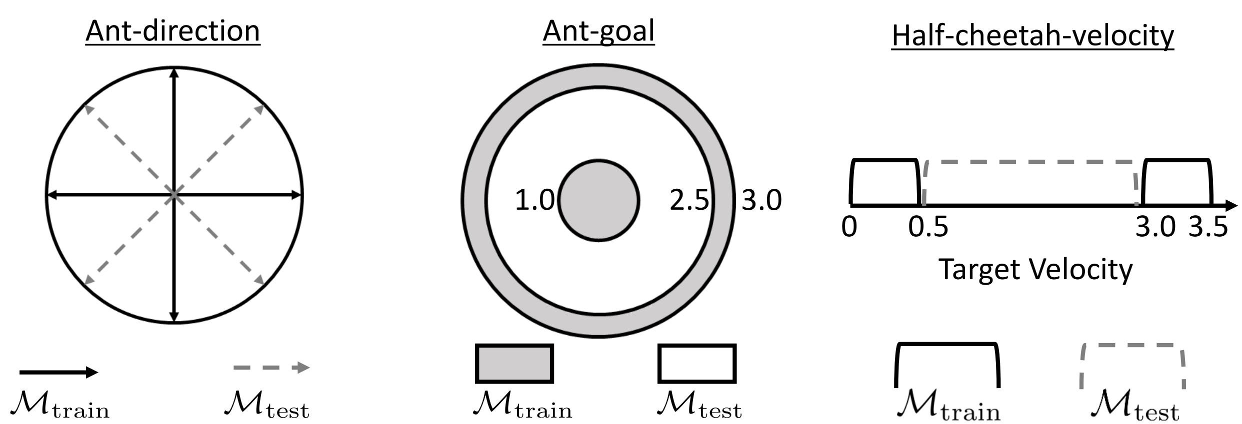


Figure 10: Illustrative examples of OOD MuJoCo tasks.

Latent Dynamics Mixture: Key Idea

- **Step 1**. Encode the dynamics of each MDP into a latent space representation, z.
- **Step 2**. Mix these latent embeddings from multiple training tasks to form a mixture embedding $\tilde{z}$.
- **Step 3**. Decode this combined latent vector to synthesize an *imaginary* MDP, $\tilde{M}$, which is then used for training the policy network.

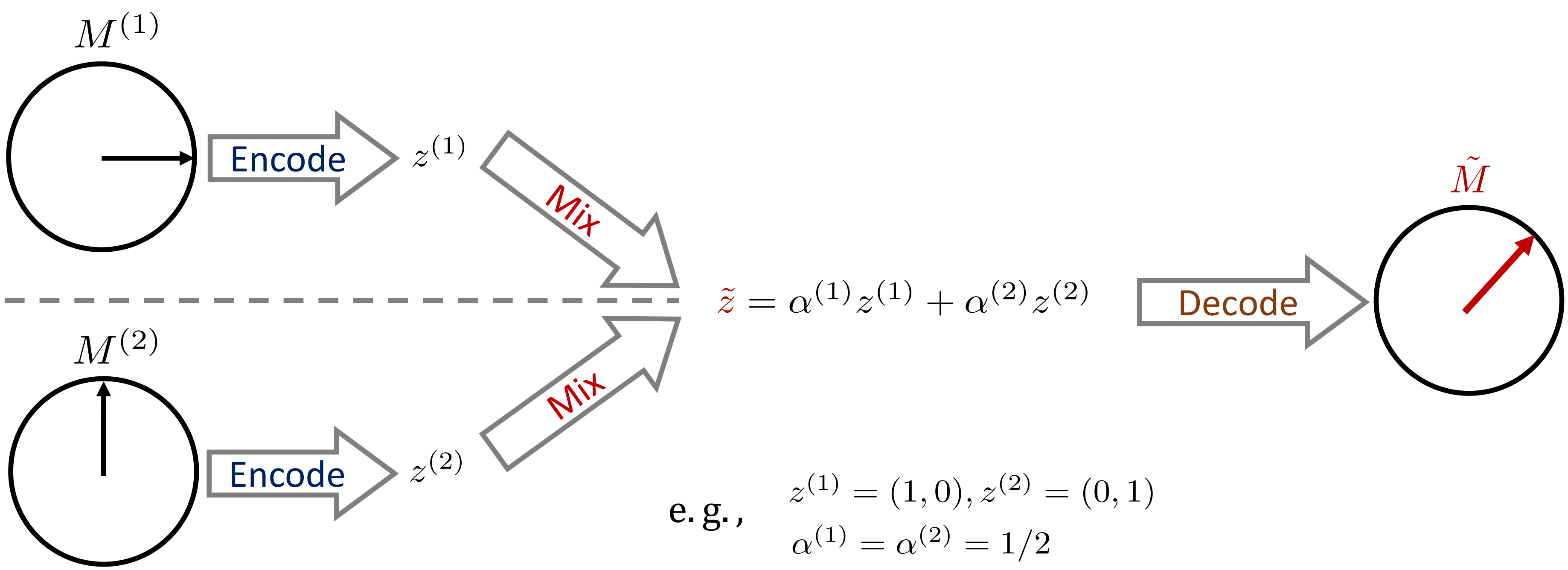


Figure 11: Conceptual flow of the Latent Dynamics Mixture (LDM) algorithm.

Variational Bayes-Adaptive Deep RL

- We need an encoder and decoder to infer (parameterize) the task dynamics and to learn the dynamics model to generate an imaginary MDP.
- VariBAD[3] addresses the challenge of posterior updates, which are often computationally infeasible, by leveraging variational inference within meta-RL. The framework is structured as follows:
  - A Variational AutoEncoder (VAE) with a recurrent network $q_\phi(\tau_{:t})$ that encodes the trajectory $\tau_{:t}$ into a latent representation.
  - The decoders p_{θ_R} and p_{θ_T} that reconstruct the reward and transition dynamics.
  - A separate policy network $\pi_\psi(a_t|s_t, b_t)$ conditioned on the inferred belief from the encoder.

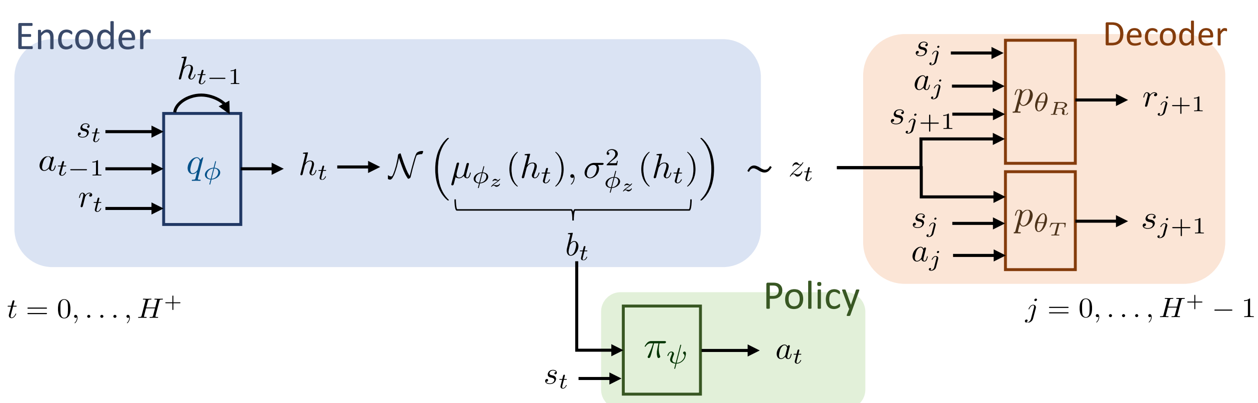


Figure 12: VariBAD architecture.

[3]Zintgraf, et al., "VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning," ICLR 2020.

Variational Bayes-Adaptive Deep RL

- The latent mean $\mu_{\phi_z}(h_t)$ and variance $\sigma^2_{\phi_z}(h_t)$ of the VAE are neural network outputs given the context of the current meta-episode, $h_t = q_\phi(\tau_{:t})$.
- The belief $b_t = \left(\mu_{\phi_z}(h_t), \sigma^2_{\phi_z}(h_t)\right)$ is expected to contain the inferred task dynamics until time t.
- As t grows and the agent explores the task space,
  - the uncertainty $\sigma^2_{\phi_z}(h_t)$ about the current meta-episode's task diminishes.
  - the converged mean $\mu_{\phi_z}(h_t)$ represents the inferred parametrization of the current task.

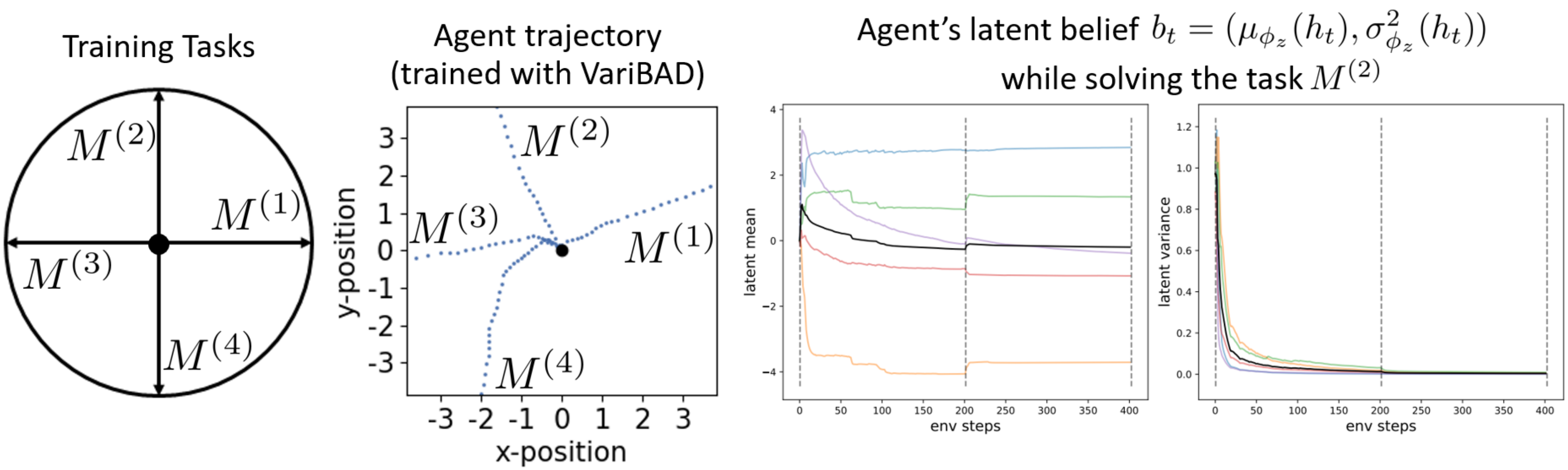


Figure 13: VariBAD's belief update example on the Ant-direction task. The latent beliefs are 5-dimensional.

Latent Dynamics Mixture

- Multiple workers with shared networks.
  - All workers jointly train a shared policy network and a latent dynamics network.
  - Unlike VariBAD, we use two separate encoders q_{ϕ_p} and q_{ϕ_v}.
  - The policy part is optimized with PPO, and the VAE is trained to maximize ELBO.

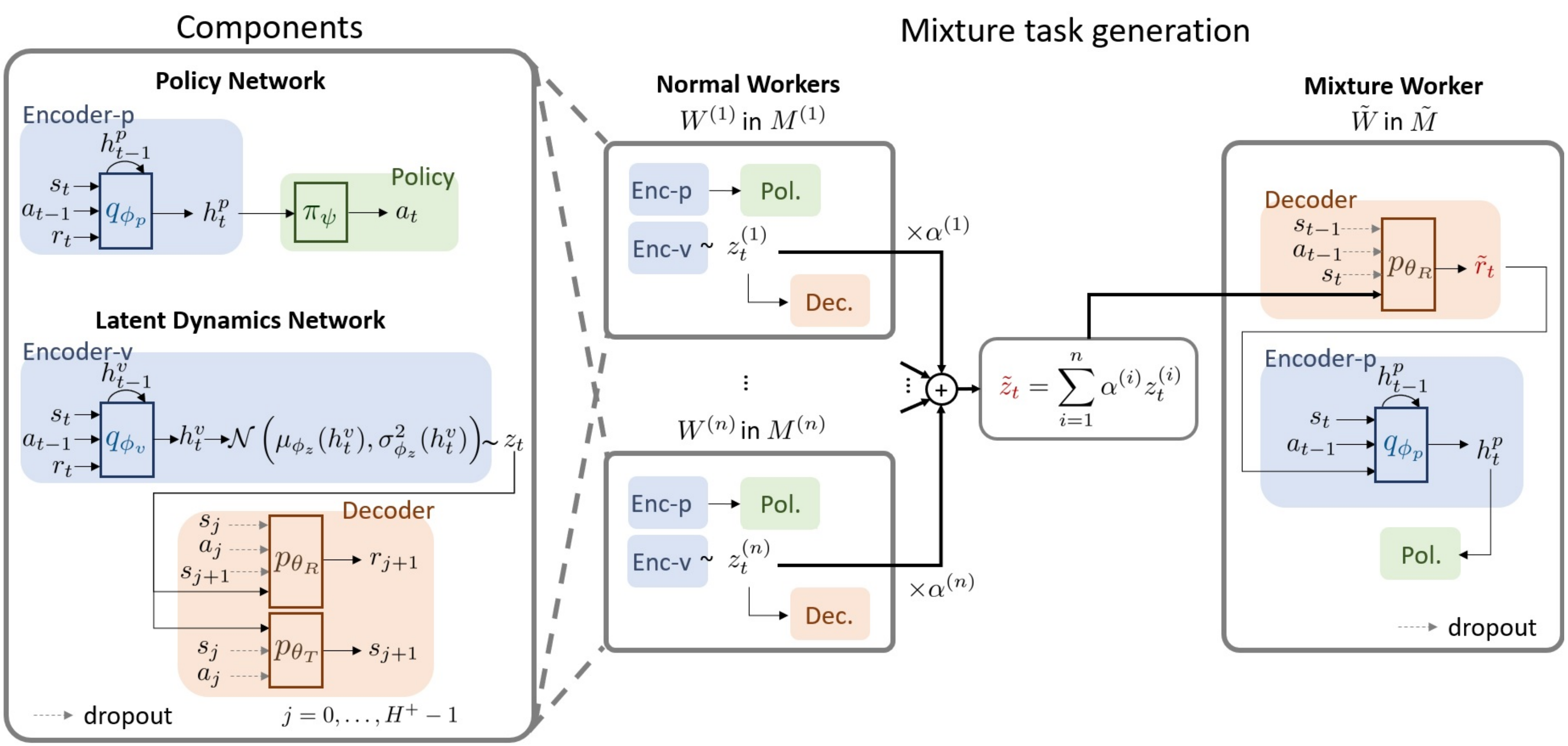
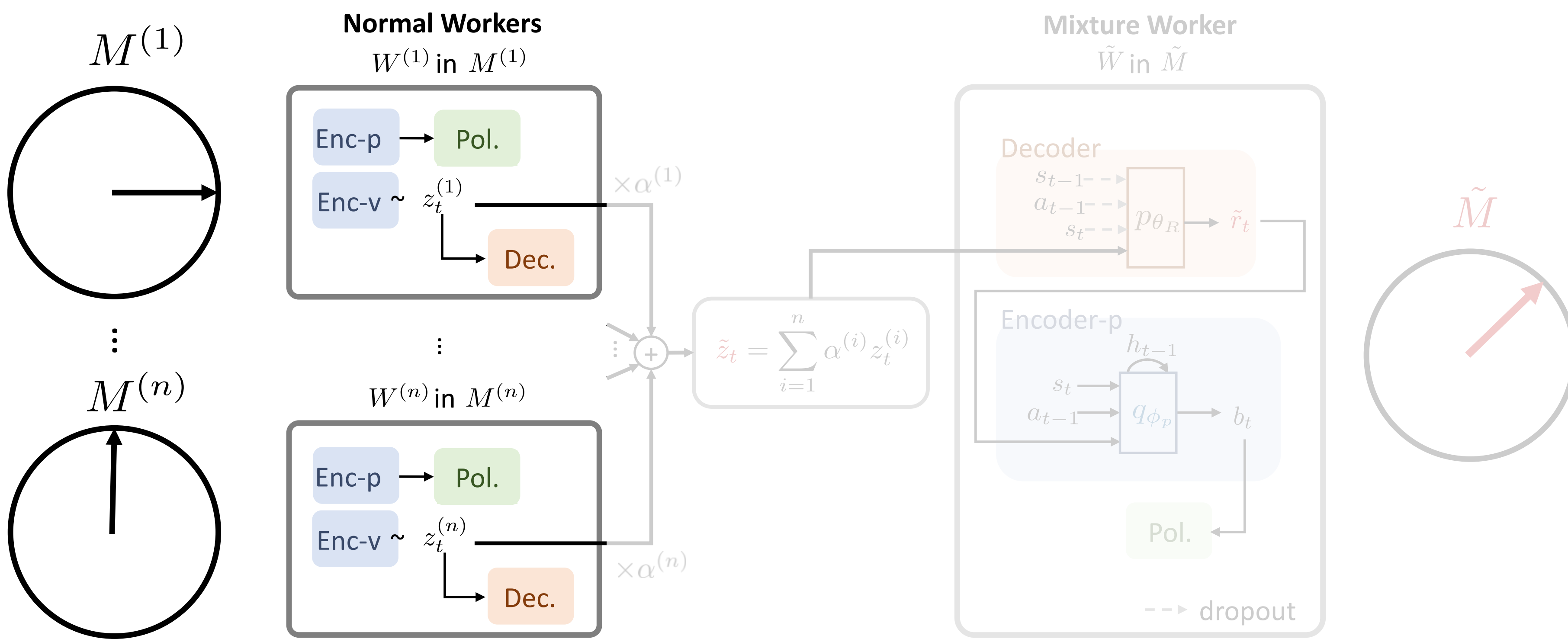


Figure 14: Imaginary task generation from Latent Dynamics Mixture (LDM).

Latent Dynamics Mixture

- **Step 1**. Each normal worker $W^{(i)}$ encodes its belief for $M^{(i)}$ sampled from $\mathbb{P}(\mathcal{M}_{\text{train}})$

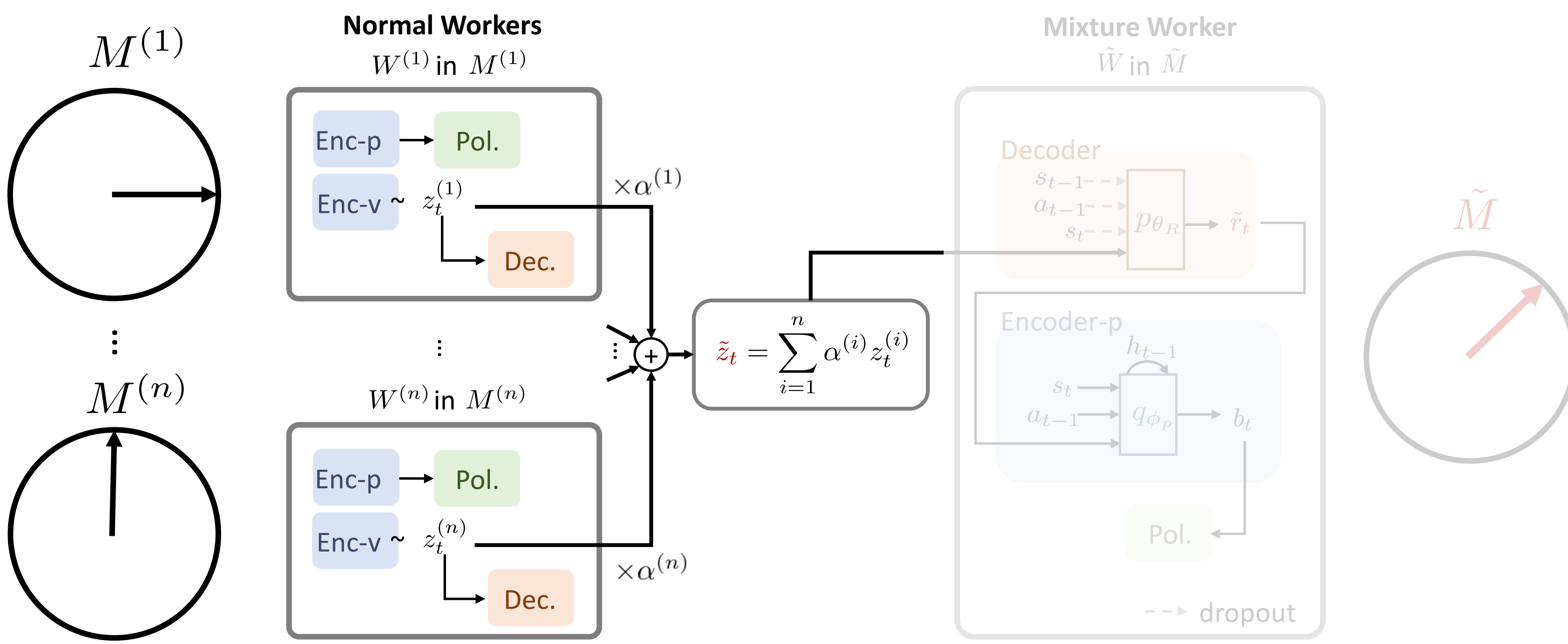


Latent Dynamics Mixture

- **Step 2.** A mixture worker computes a mixture latent belief $\tilde{z}_t = \sum_{i=1}^{n} \alpha^{(i)} z_t^{(i)}$, where

$$\left(\alpha^{(1)}, \ldots, \alpha^{(n)}\right) \sim \beta \cdot \mathrm{Dirichlet}(\mathbb{1}_n) - \frac{\beta - 1}{n}. \tag{14}$$

  - The weights satisfy $\sum_{i=1}^{n} \alpha^{(i)} = 1, \quad \mathbb{E}\left[\alpha^{(i)}\right] = 1/n$.
  - The hyperparameter β controls the extrapolation level. If $\beta = 1$, $0 \le \alpha^{(i)} \le 1$.

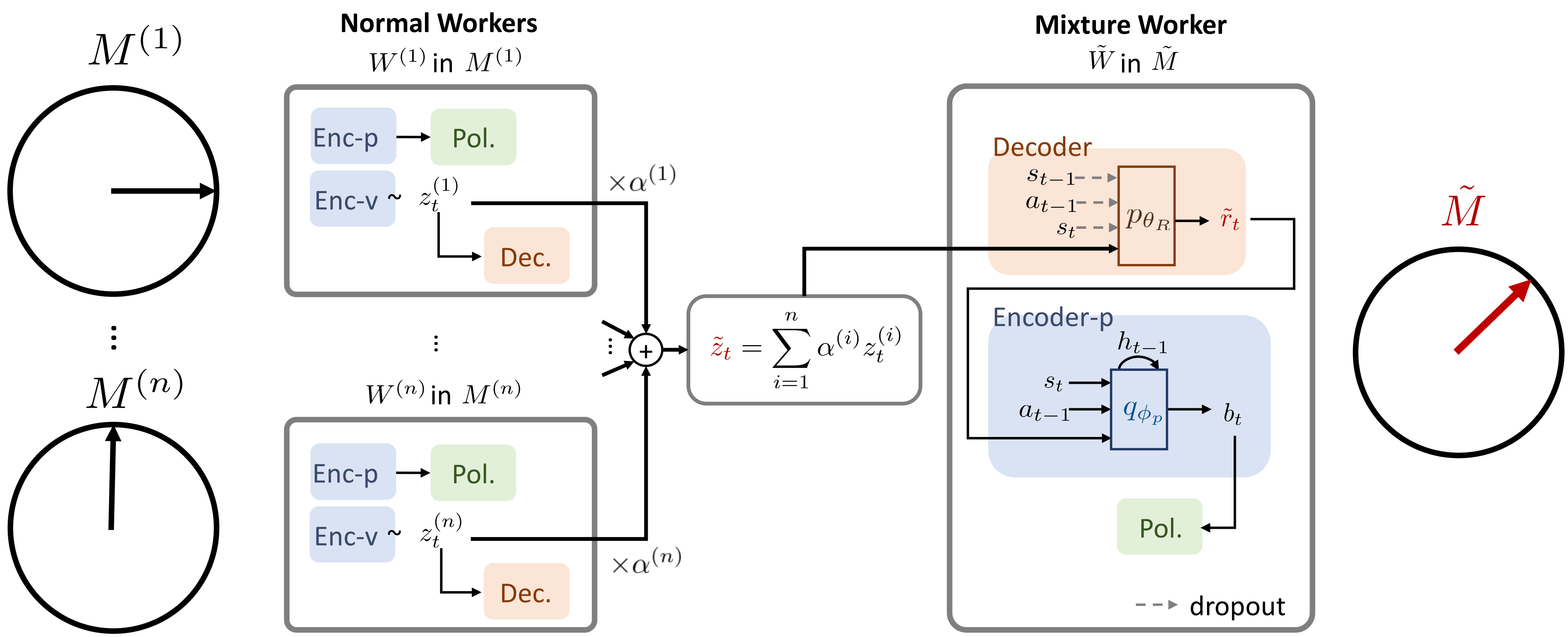


Latent Dynamics Mixture

- **Step 3.** Given $\tilde{z}_t$, the mixture worker $\tilde{W}$ generates an imaginary MDP $\tilde{M}$ with imaginary rewards $\tilde{r}_t \sim p_{\theta_R}(\cdot|s_t, a_t, s_{t+1}; \tilde{z}_t)$. But the states are from a real training MDP $M^{(k)} \sim \mathbb{P}(\mathcal{M}_{\text{train}})$.

$$\tilde{M} = \left(\mathcal{S}, \mathcal{A}, \tilde{R}, T^{(k)}, T_0^{(k)}, \gamma, H\right). \tag{15}$$

  – The trajectories from $\tilde{M}$ are only used to train the policy part: q_{ϕ_p} and π_ψ, not the VAE part.



Dropout on Decoder Input

- The decoder easily overfits the state and action observations, ignoring the latent belief z_t.
- States that are unseen during training may be assigned low rewards regardless of the mixture latent belief $\tilde{z}_t$ (e.g., the dotted states in Figure 15).
- Therefore we apply dropout to all inputs of the decoder except the latent belief when training and generating with the decoder.

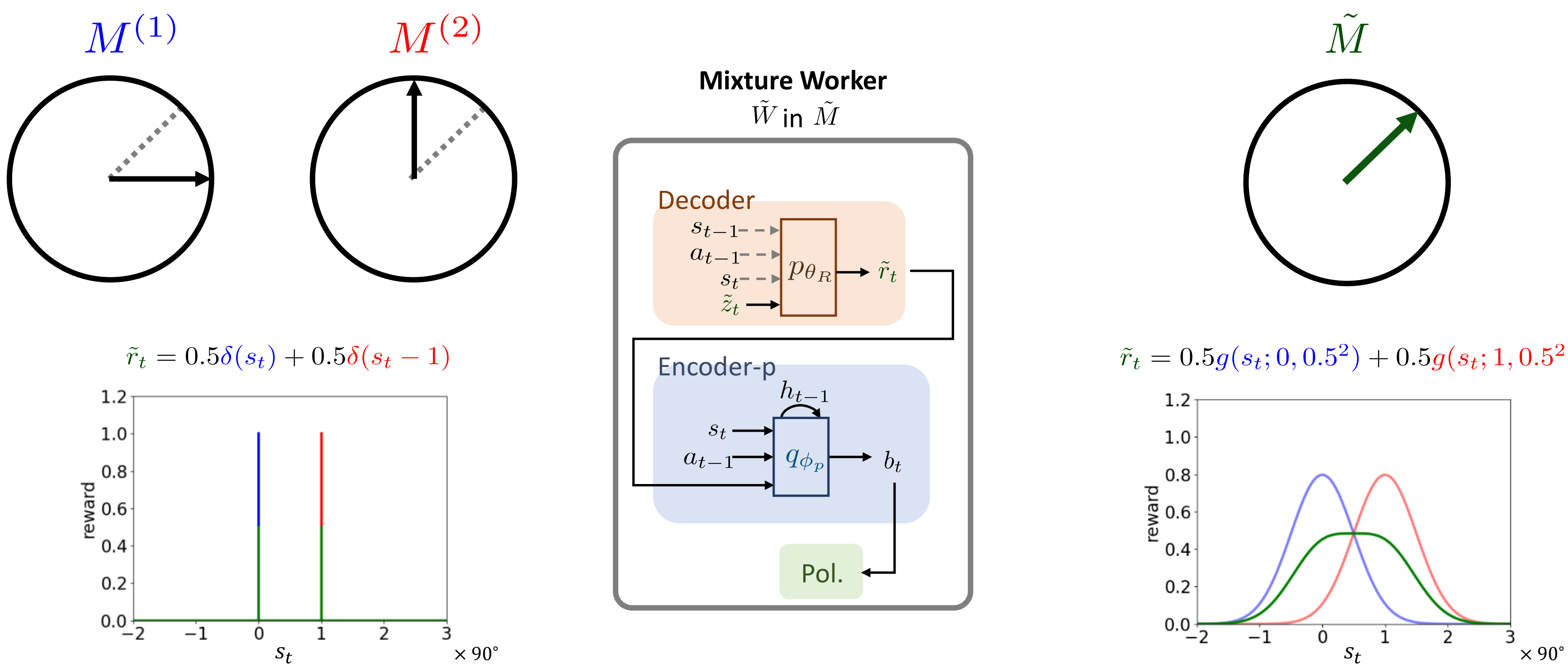
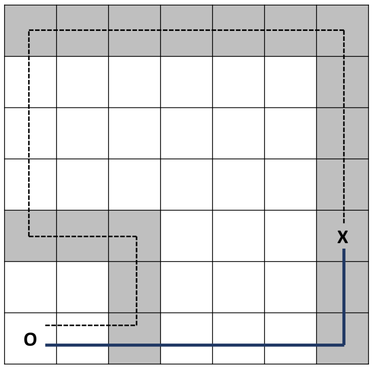


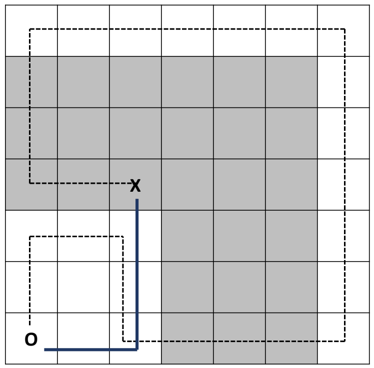
Figure 15: Applying dropout to the decoder input.

Experiments – Setup

- We evaluate LDM and baselines on Gridworld and MuJoCo tasks, where we strictly divide the entire task distribution into disjoint training and test tasks.
- Gridworld
  - 7×7 Gridworld task with $N = 4$ rollout episodes of horizon $H = 30$ each.
  - Number of tasks: $|\mathcal{M}_{\text{train}}| = 18$ and $|\mathcal{M}_{\text{test}}| = 27$.



(a) $\mathcal{M}_{\text{train}}$



(b) $\mathcal{M}_{\text{test}}$

Figure 16: **Gridworld example. (a)** Training MDPs $\mathcal{M}_{\text{train}}$. **(b)** Test MDPs $\mathcal{M}_{\text{test}}$. A goal is located at one of the shaded positions.

- MuJoCo
  - We evaluate over three OOD MuJoCo tasks introduced in Figure 10 with $N = 2$ and $H = 200$.

Experiments – Gridworld

- We report the results at the last rollout episode in terms of
  - the mean returns in $\mathcal{M}_{\text{train}}$ and $\mathcal{M}_{\text{test}}$.
  - the number of tasks in $\mathcal{M}_{\text{test}}$ in which the agent fails to reach the goal (out of 27 tasks).
- The oracle methods, that are trained on the entire task set $\mathcal{M} = \mathcal{M}_{\text{train}} \cup \mathcal{M}_{\text{test}}$ including the test tasks, are evaluated for reference.

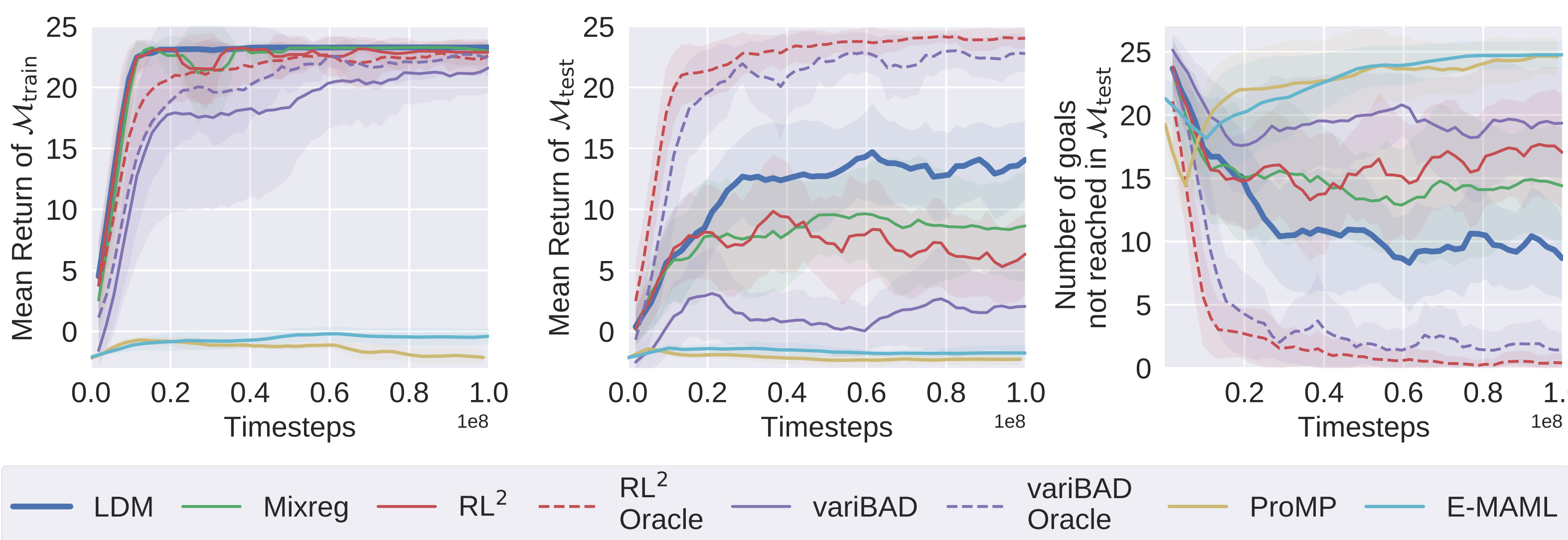


Figure 17: Gridworld results.

Experiments – Gridworld

- We provide an empirical analysis to confirm that LDM successfully generates appropriate imaginary tasks that contribute to solving the test tasks.
- The dropout applied to the next state input of the decoder plays a critical role here.
  - Dropout ($p_{\text{drop}} = 0.7$) leads the decoder to attribute high rewards to certain goals in $\mathcal{M}_{\text{test}}$.

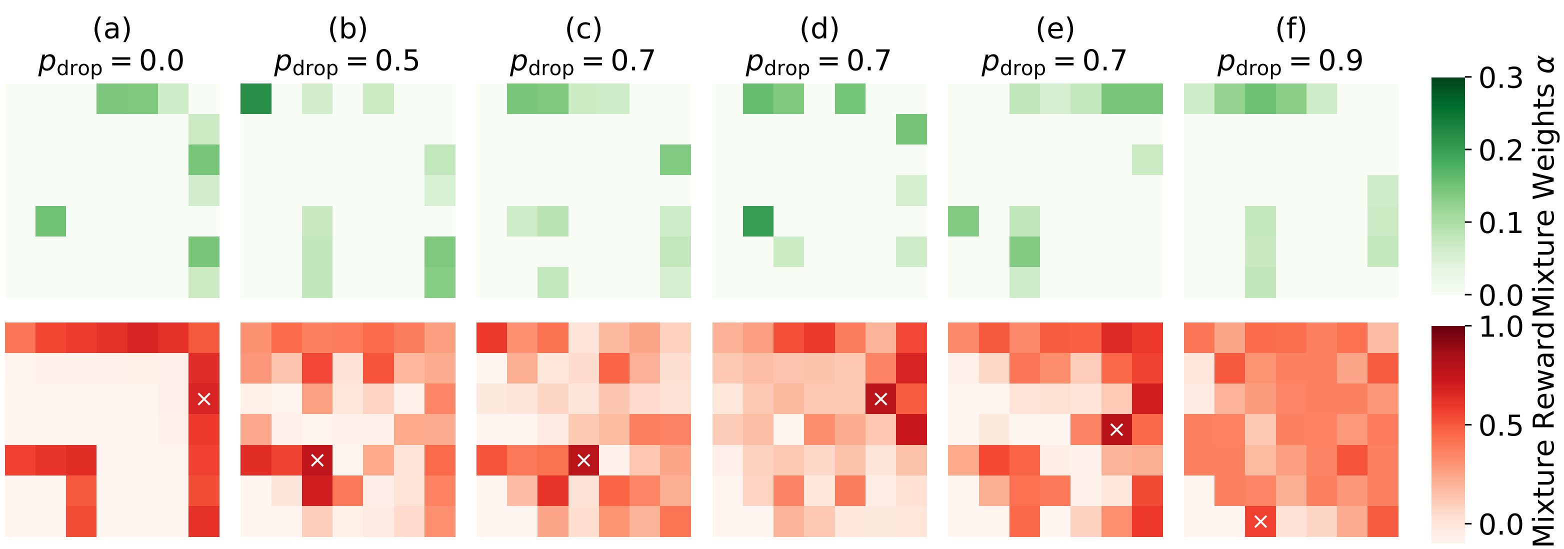
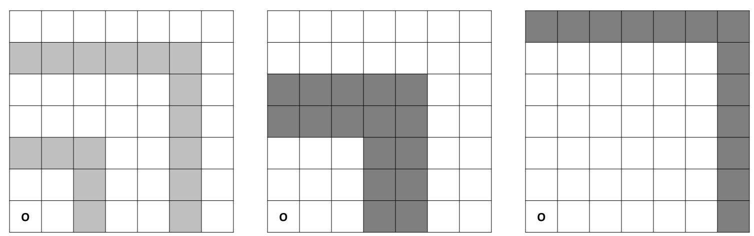


Figure 18: Examples of mixture tasks generated by LDM. **First row**: Mixture weights ($\alpha^{(i)}$) multiplied to the latent beliefs ($z_{H^+}^{(i)}$). **Second row** (reward map): decoder output for each next state conditioned on the mixture weights from the first row. $\times$ denotes the state yielding the maximum reward.

Experiments – Gridworld

- We demonstrate that LDM's applicability is not confined to target tasks within the interpolation of training tasks with the following experiment on Gridworld-extrapolation task.
  - Recall: $\left(\alpha^{(1)}, \ldots, \alpha^{(n)}\right) \sim \beta \cdot \mathrm{Dirichlet}(\mathbb{1}_n) - \frac{\beta-1}{n}$.
  - For lower β settings such as $\beta = 1.0$ and $\beta = 1.5$, LDM concentrates on $\mathcal{M}_{\mathrm{test1}}$.
  - As β increases, we observe a decline in returns for tasks in $\mathcal{M}_{\mathrm{test1}}$, accompanied by an increase for tasks in $\mathcal{M}_{\mathrm{test2}}$.



(a) $\mathcal{M}_{\mathrm{train}}$ (b) $\mathcal{M}_{\mathrm{test1}}$ (c) $\mathcal{M}_{\mathrm{test2}}$

Figure 19: Gridworld-extrapolation task, that consists of two separate task regions, $\mathcal{M}_{\mathrm{test1}}$ and $\mathcal{M}_{\mathrm{test2}}$.

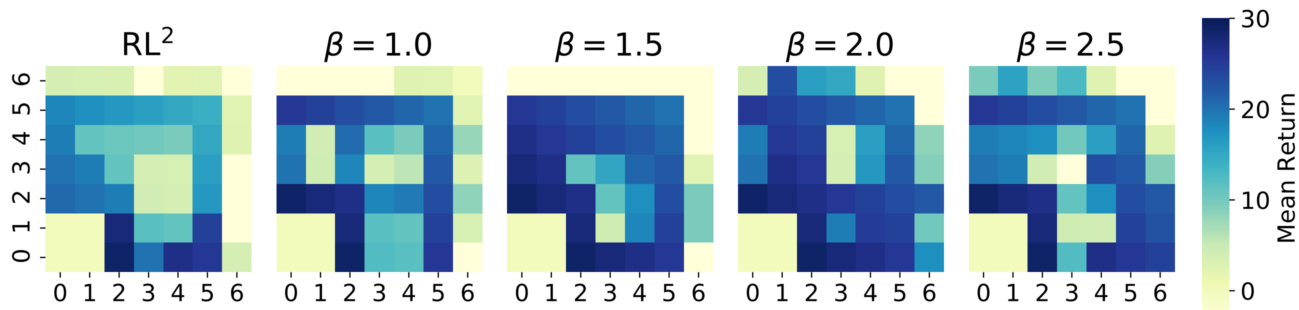


Figure 20: Extrapolation results. Returns of LDM for different extrapolation level β on the Gridworld-extrapolation task.

Experiments – MuJoCo

- We evaluate our method and baselines on three OOD meta-RL benchmarks: Ant-direction, Ant-goal, and Half-cheetah-velocity.
- The mean test returns in $\mathcal{M}_{\text{eval}}$, which is a fixed subset of $\mathcal{M}_{\text{test}}$, are illustrated in Figure 21.
- LDM surpasses the performance of non-oracle baselines.

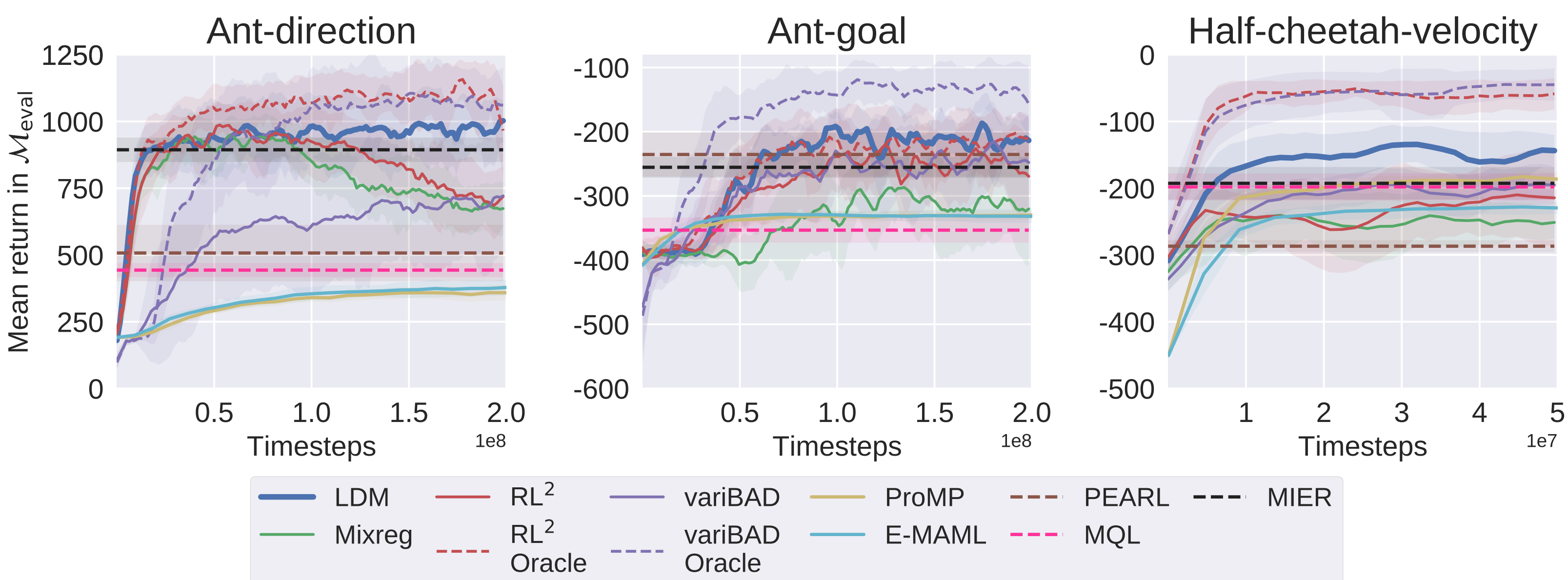
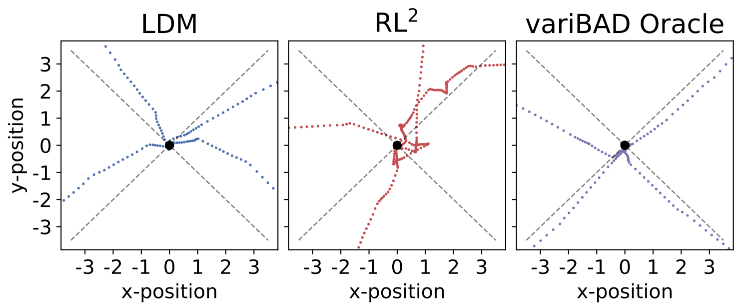


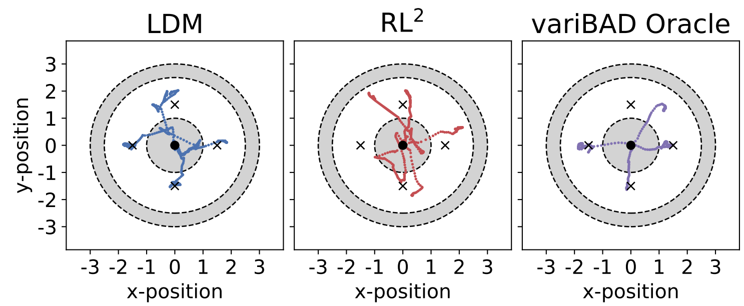
Figure 21: MuJoCo results. Mean returns at the last rollout episode in $\mathcal{M}_{\text{eval}}$.

Experiments – MuJoCo

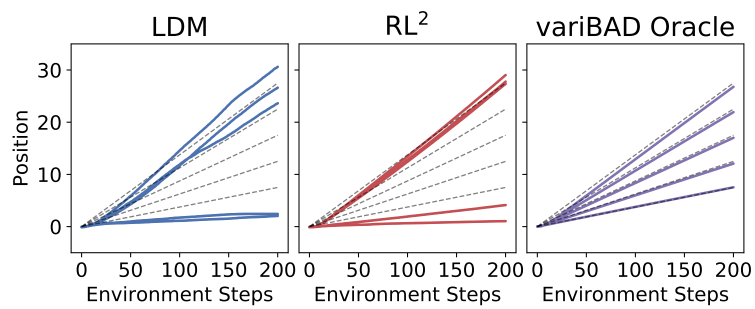
- We demonstrate the sample trajectories of the agents in $\mathcal{M}_{\text{eval}}$ at the last rollout episode.



(a) Ant-drection, 4 trajectories from $\mathcal{M}_{\text{eval}}$.



(b) Ant-goal, 4 trajectories from $\mathcal{M}_{\text{eval}}$.

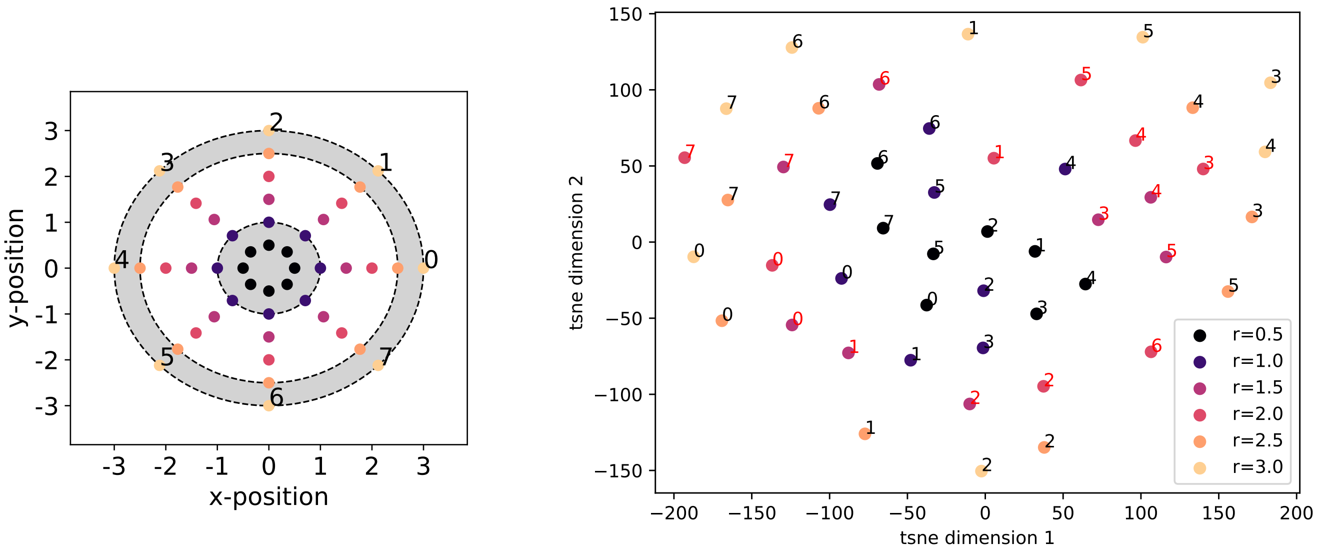


(c) Half-cheetah-velocity, 5 trajectories from $\mathcal{M}_{\text{eval}}$.

Figure 22: Sampled trajectories of the agents in $\mathcal{M}_{\text{eval}}$. The targets of $\mathcal{M}_{\text{eval}}$ are indicated as dashed lines or cross marks

Analysis on Task Embedding

- We have conducted empirical tests that affirm the latent models effectively reflect the structure of the test task, although not trained in $\mathcal{M}_{\text{test}}$, which supports the efficacy of LDM.
  - We sample 48 tasks in Ant-goal as in Figure 23a. 32 tasks from $\mathcal{M}_{\text{train}}$ and 16 from $\mathcal{M}_{\text{test}}$.
  - For each task, We evaluate latent beliefs z_{H^+} at the end of each meta-episode.



(a) Ant-goal: sample tasks.

(b) Ant-goal: t-SNE plot of test-time latent belief z_{H^+}.

Figure 23: Latent belief distributions. Latent dynamics network's learned latent models on Gridworld and Ant-goal. The red numbers denote the tasks that belong to $\mathcal{M}_{\text{test}}$.

Limitation

- Parametric task variability
  - LDM requires the tasks to exhibit simple parametric variations, such as the goal position, target direction, and target velocity, that could be easily modeled by the latent embedding.
  - If the task distribution is more complex, we can not expect the test tasks to be represented as the interpolations of the latent embedding of the training tasks.

- Interpolation
  - LDM can generalize to test tasks that are within the convex hull of the training tasks' latent embeddings.
  - The extrapolation achieved by the hyperparameter β is limited.

- Tasks with varying reward dynamics
  - LDM prepares for test tasks that exhibit unseen reward dynamics.
  - It uses states from the real training tasks, therefore unable to prepare for OOD tasks with varying state transition dynamics.

Parameterizing Non-Parametric Meta-Reinforcement Learning Tasks via Subtask Decomposition

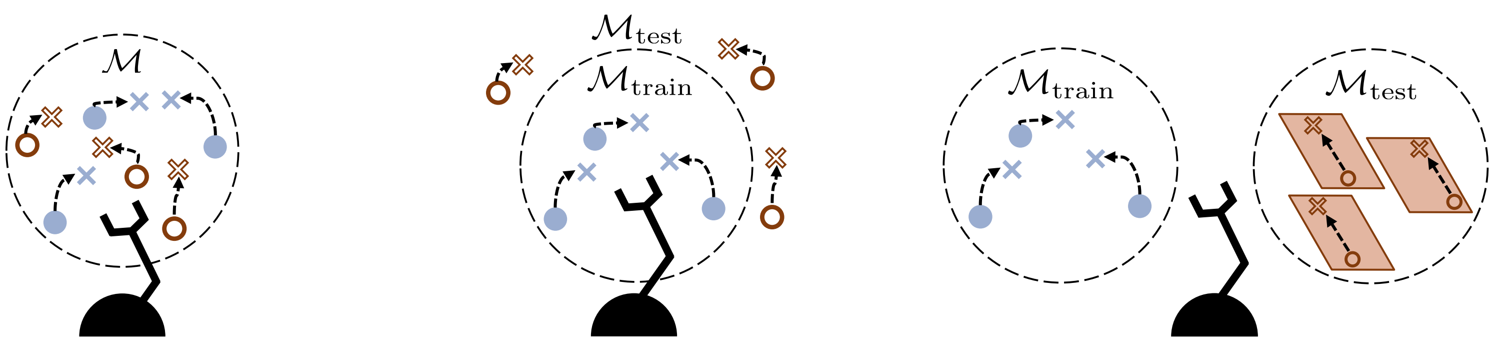
Suyoung Lee, Myungsik Cho, and Youngchul Sung

To be presented at NeurIPS 2023

Introduction

- Conventional Meta-RL methods and LDM are evaluated on tasks with variations that can be expressed in a shared parametric form representing the task dynamics (Figure 24a and 24b).

- Now, we explore a more general meta-RL framework that addresses generalization in more qualitatively distinct tasks, namely with non-parametric task variability[4].
  - For example, training on the task "Pick-place" and testing on "Sweep-into" from Meta-World (Figure 24c).



(a) "Pick-place" in standard in-distribution meta-RL setup.

(b) "Pick-place" in out-of-distribution meta-RL setup.

(c) Non-parametric task variation between "Pick-place" and "Sweep-into."

Figure 24: Illustrating different meta-RL scenarios. ∘: object, ×: goal.

[4] Yu, et al., "Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning," CoRL 2019.

Motivation

- Despite the complex non-parametric task variability, tasks may share common elementary subtasks.
  - **"Pick-place"** = "grip-object" + "place-object"
  - **"Sweep-into"** = "grip-object" + "push-object"
- Our primary strategy is to

  Decompose each non-parametric task into a set of shared elementary subtasks,

  which allows to parameterize a task with embeddings representing the subtask composition along with the conventional parametric variations.

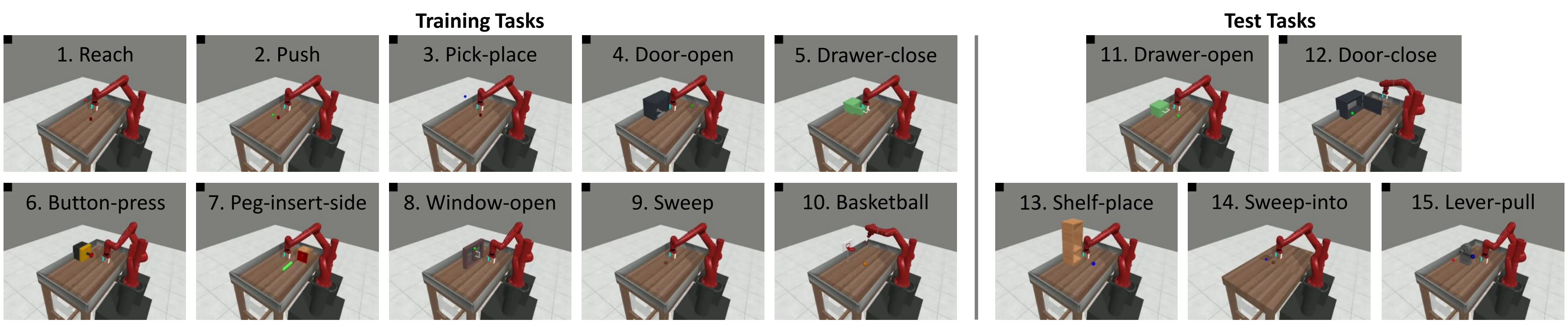


Figure 25: Meta-World ML-10 robotic manipulation benchmark. There are 10 training tasks and 5 test tasks with non-parametric task variability. Within each task, there are 50 parametric variations.

Subtask Decomposition and Virtual Training: Key Idea

- For example, we want
  - Embedding(**"Pick-place"**) $= (0.5, 0.5, 0.0)$
  - Embedding(**"Sweep-into"**) $= (0.0, 0.5, 0.5)$
  - Where each dimension of the embedding corresponds to the weights for the following subtasks: "place-object," "grip-object," and "push-object".
- The key problem is to learn the
  - Basis: the set of elementary subtasks
  - Coefficients: the decomposition of each task given the set of elementary subtasks.
- We employ meta-learning for the subtask decomposition (**SD**) process using a Gaussian mixture variational autoencoder (GMVAE) that encodes the trajectory up to the current timestep into
  - Compositional information: latent categorical contexts
  - Parametric variability information: latent Gaussian contexts.
- To further enhance generalization to unseen compositions of learned elementary subtasks, we propose a virtual training (**VT**) process similar to that of LDM.

Subtask Decomposition

- The proposed architecture of SDVT incorporates three main components: encoder, decoder, and policy, similar to the VariBAD architecture.
- The major difference is that the encoder involves a categorical latent variable y_t.

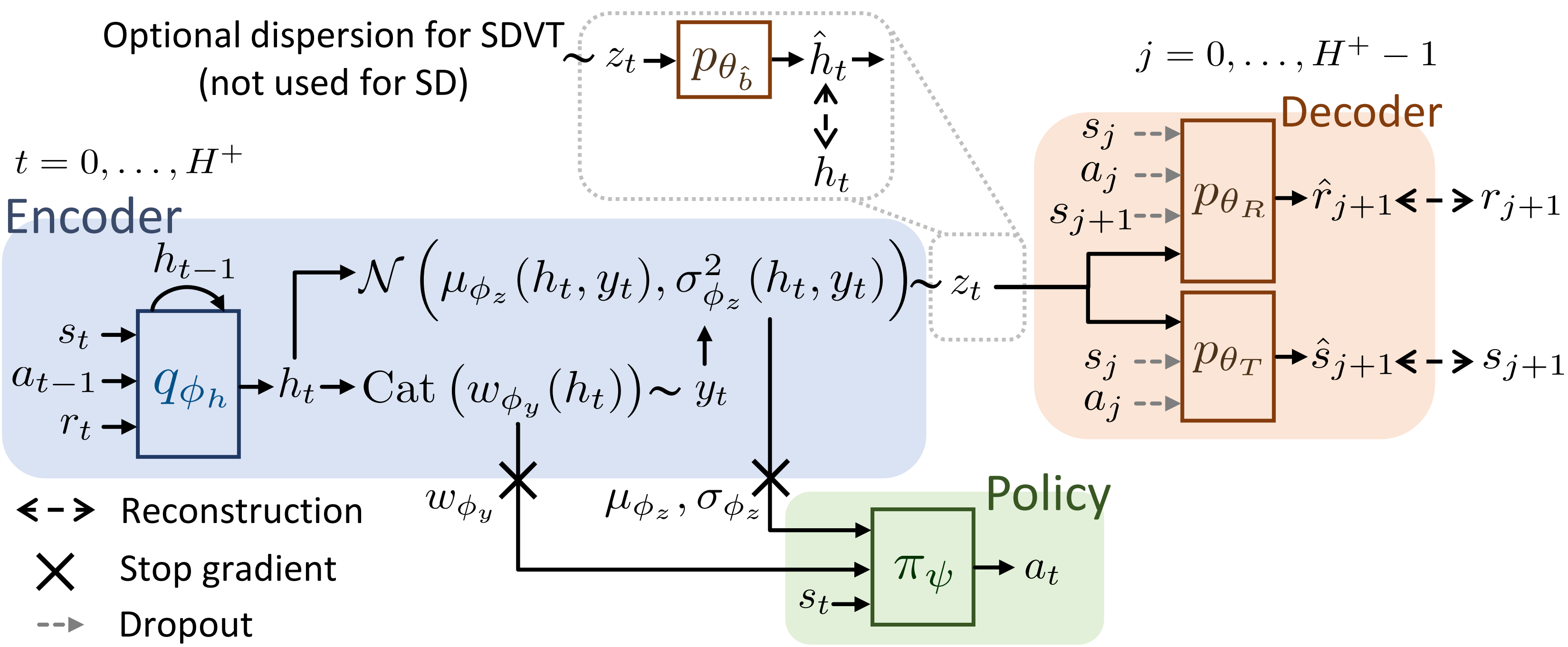


Figure 26: SDVT architecture.

Subtask Decomposition

- Encoder: $q_\phi(y_t, z_t|h_t) = q_{\phi_y}(y_t|h_t) q_{\phi_z}(z_t|h_t, y_t)$
  - A recurrent network encodes the past trajectory $\tau_{:t}$ into a hidden embedding $h_t = q_{\phi_h}(\tau_{:t})$.
  - The categorical encoder $q_{\phi_y}(y_t|h_t)$: $\mathrm{Cat}\left(\omega_{\phi_y}(h_t)\right)$ samples y_t, where $\omega_{\phi_y}(h_t) \in \Delta^K$.
  - Then the multivariate Gaussian encoder $q_{\phi_z}(z_t|h_t, y_t)$: $\mathcal{N}\left(\mu_{\phi_z}(h_t, y_t), \sigma^2_{\phi_z}(h_t, y_t)\right)$ samples a continuous latent context z_t.
  - z_t contains the compositional information of the current task and the parametric information of the subtasks.

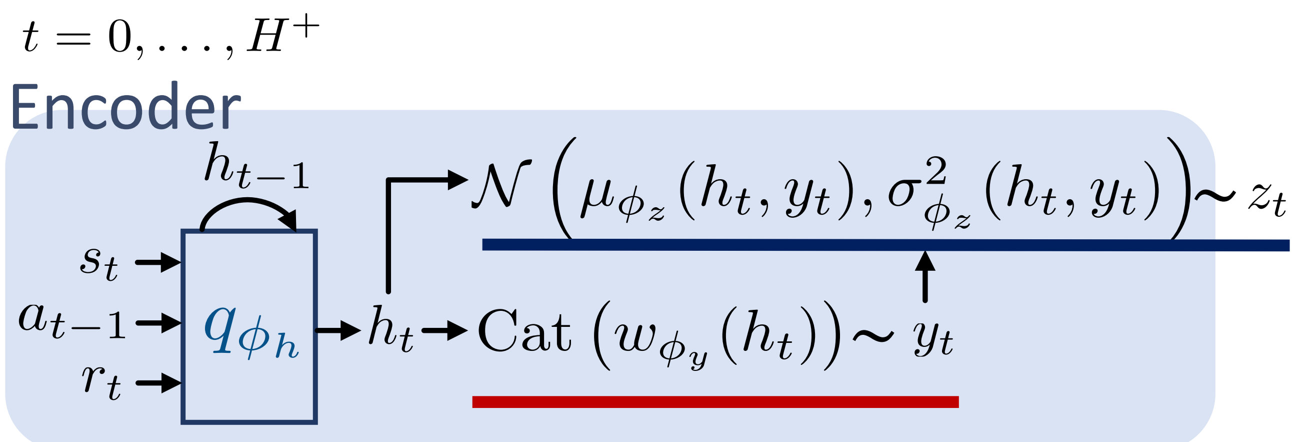


Figure 27: SDVT encoder.

Learned Subtask Compositions

- Learned subtask compositions on ML-10.
  - Each column denotes the tasks in ML-10 ($\mathcal{M}_{\text{train}} : 1 \sim 10$, $\mathcal{M}_{\text{test}} : 11 \sim 15$).
  - Each row number represents the index of the elementary subtask.
- We find that such learned subtask compositions $y_{H^+} \in \triangle^K$ are shared by qualitatively similar tasks.
  - (3) "Pick-place," (7) "Peg-insert-side," (10) "Basketball"
  - (1) "Reach," (5) "Drawer-close," (8) "Window-open"

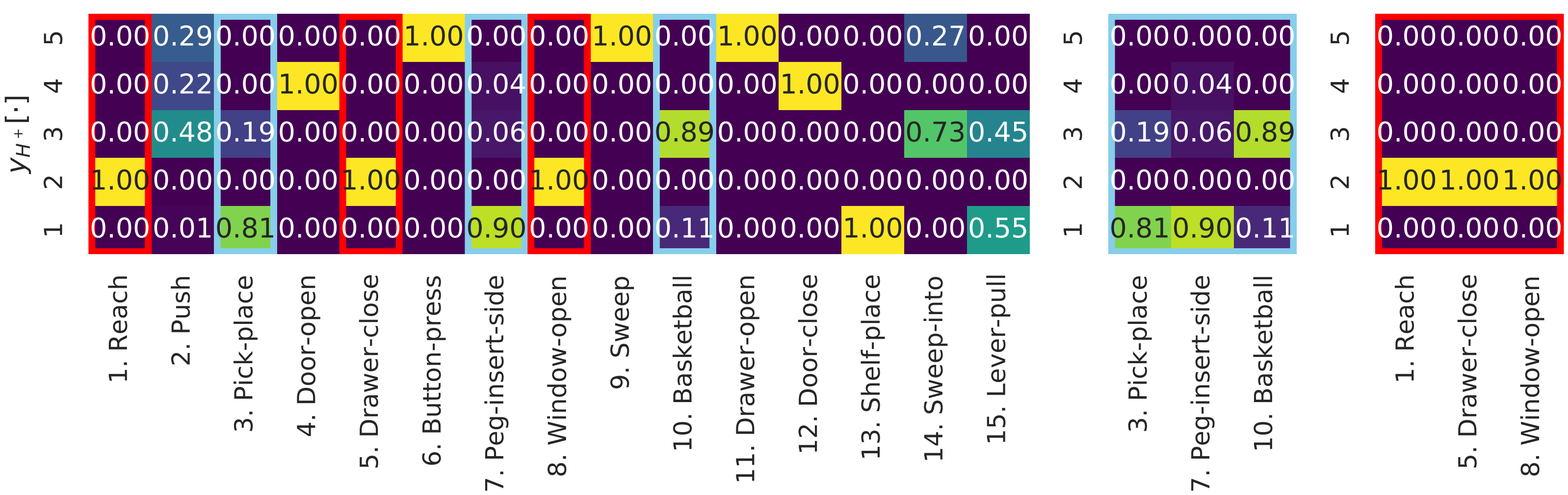


Figure 28: Learned subtask compositions on ML-10 by SDVT-LW with $K = 5$, $\alpha_c = 0.5$.

GMVAE Objectives

- The objective of the GMVAE is to maximize the evidence lower bound (ELBO), for $t = 0, \dots, H^+$ and for the trajectory distribution $d(M^{(k)}, \tau_{:H^+})$ at MDP $M^{(k)}$, induced by the policy π_ψ.

$$\mathrm{ELBO}_t(\phi, \theta) = \mathbb{E}_{d(M^{(k)}, \tau_{:H^+})}\left[\mathbb{E}_{q_\phi(y_t, z_t | h_t)} \mathcal{J}_{\mathrm{GMVAE}}\right], \tag{16}$$

$$\mathcal{J}_{\mathrm{GMVAE}} = \alpha_R \mathcal{J}_{\text{R-rec}} + \alpha_T \mathcal{J}_{\text{T-rec}} + \alpha_g \mathcal{J}_{\mathrm{reg}} + \alpha_c \mathcal{J}_{\mathrm{cat}}. \tag{17}$$

- Reconstruction objectives

$$\mathcal{J}_{\text{R-rec}} = \sum_{j=0}^{H^+-1} \log p_{\theta_R}(r_{j+1} | s_j, a_j, s_{j+1}; z_t), \tag{18}$$

$$\mathcal{J}_{\text{T-rec}} = \sum_{j=0}^{H^+-1} \log p_{\theta_T}(s_{j+1} | s_j, a_j; z_t). \tag{19}$$

  – The reconstruction objectives are computed for all timesteps in the meta-episode, including the future $(j > t)$ and past $(j < t)$ (same as VariBAD and LDM).

GMVAE Objectives

- The objective of the GMVAE is to maximize the evidence lower bound (ELBO), for $t = 0, \ldots, H^+$ and for the trajectory distribution $d(M^{(k)}, \tau_{:H^+})$ at MDP $M^{(k)}$, induced by the policy π_ψ.

$$\mathrm{ELBO}_t(\phi, \theta) = \mathbb{E}_{d(M^{(k)}, \tau_{:H^+})} \left[\mathbb{E}_{q_\phi(y_t, z_t | h_t)} \mathcal{J}_{\mathrm{GMVAE}} \right], \tag{20}$$

$$\mathcal{J}_{\mathrm{GMVAE}} = \alpha_R \mathcal{J}_{\text{R-rec}} + \alpha_T \mathcal{J}_{\text{T-rec}} + \alpha_g \mathcal{J}_{\mathrm{reg}} + \alpha_c \mathcal{J}_{\mathrm{cat}}. \tag{21}$$

- Regularization objective

$$\mathcal{J}_{\mathrm{reg}} = \log \frac{p_{\theta_z}(z_t | y_t)}{q_{\phi_z}(z_t | h_t, y_t)}. \tag{22}$$

  - Unlike the standard VAE that assumes a standard normal prior, we learn K distinct Gaussian priors conditioned on y_t and minimize the KL divergence to the learned posterior $q_{\phi_z}(z_t | h_t, y_t)$.

- Categorical objective

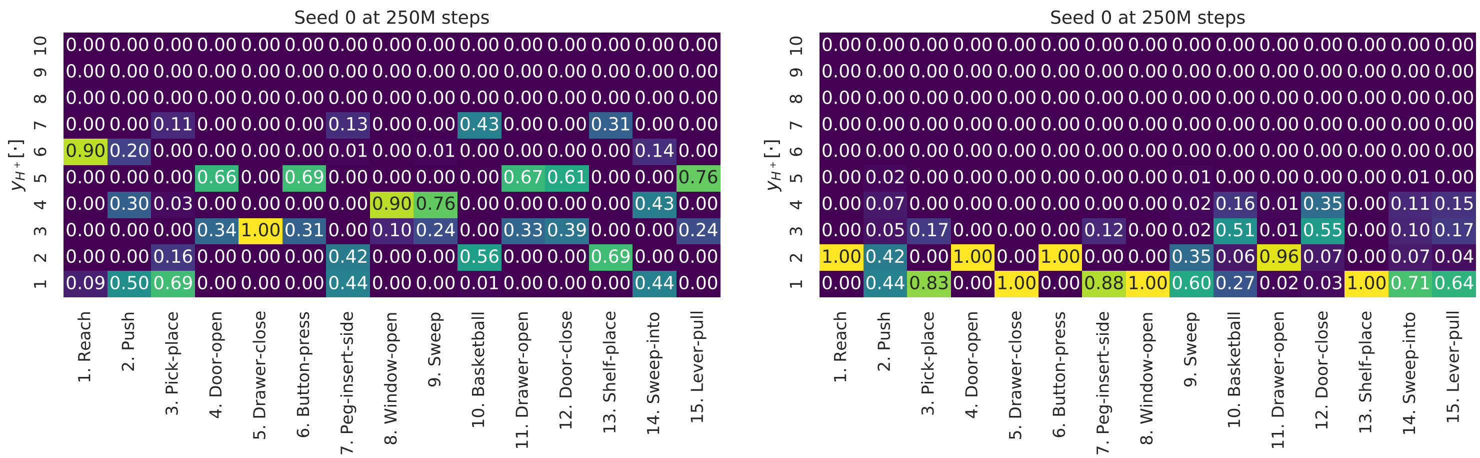
$$\mathcal{J}_{\mathrm{cat}} = \log \frac{p(y_t)}{q_{\phi_y}(y_t | h_t)}. \tag{23}$$

  - The categorical objective $\mathcal{J}_{\mathrm{cat}}$ maximizes the conditional entropy of y_t given h_t.
  - We use a fixed uniform prior $p(y_t)$.
  - We set the coefficient α_c large enough to penalize one-hot subtask compositions.

Occupancy Regularization

- The number of underlying subtasks K (i.e., dimension of y_t) is a crucial hyperparameter that should be determined based on the number of training tasks N_{train} and their similarities.
- We assume $K^* < N_{\text{train}}$, otherwise each task will be classified into a separate subtask with one-hot label, preventing learning shareable subtasks.
- We start with a sufficiently large $K = N_{\text{train}}$ and regularize ELBO objective with the following occupancy regularization to progressively reduce the number of effective subtasks.

$$\mathcal{J}_{\text{occ}} = -\log K \left(e^{-K+1}, e^{-K+2}, \ldots, e^{-1}, e^{0}\right) \cdot y_t. \tag{24}$$



(a) SDVT ($\alpha_o = 5.0$). (b) SDVT ($\alpha_o = 10.0$).

Figure 29: Occupancy ablation. Learned subtask compositions of SDVT $(K = 10, \alpha_c = 1.0)$ for different occupancy coefficients.

Virtual Training

- LDM's task generation is limited to parametric variations.
- Now, using our GMVAE, we can condition the decoder on an imaginary composition of subtasks $\tilde{y}$.

$$\tilde{y} \sim \text{Dirichlet}(\bar{y}), \tag{25}$$

where $\bar{y}$ is the empirical running mean of y_t aggregated during training.

- $\tilde{y}$ is fixed for a meta-episode, while $\tilde{z}_t$ varies over time.
- We train the policy to maximize the sum of generated rewards $\tilde{r}_{t+1} \sim p_{\theta_R}(\cdot|s_t, a_t, s_{t+1}; \tilde{z}_t)$.

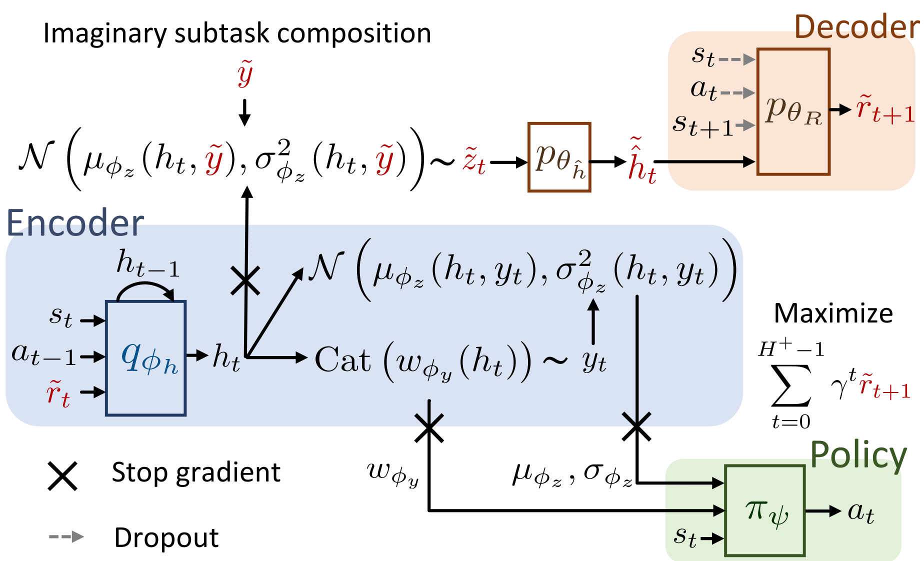
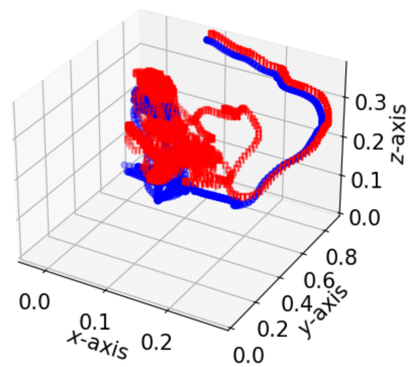


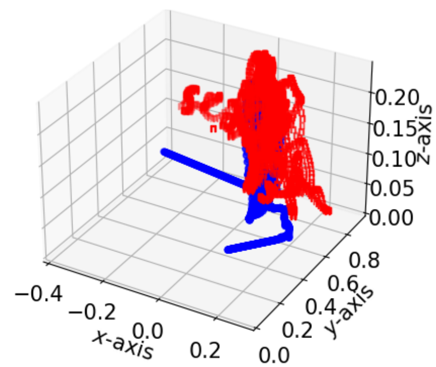
Figure 30: Generation of imaginary rewards with the decoder conditioned on a fixed imaginary subtask composition $\tilde{y}$.

Virtual Training

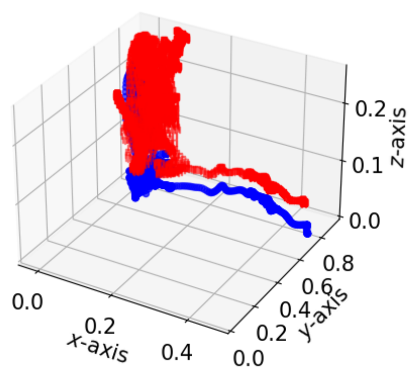
- Generated imaginary tasks result in diverse trajectories.
- We evaluate the same policy conditioned on different subtask compositions $\tilde{y}$.
- All states are from the Meta-World "Reach." Red rods: gripper and blue circles: object.



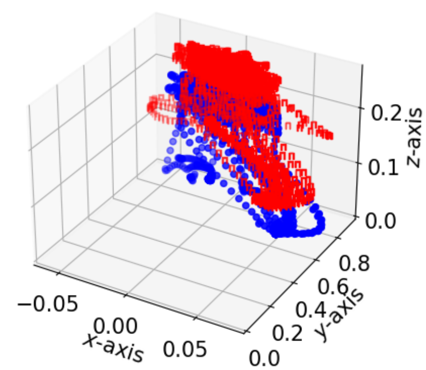
(a) $\tilde{y} = [1, 0, 0, 0, 0]$.



(b) $\tilde{y} = [0, 1, 0, 0, 0]$.



(c) $\tilde{y} = [\frac{1}{2}, \frac{1}{2}, 0, 0, 0]$.



(d) $\tilde{y} = [0, 0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$.

Figure 31: Trajectories on generated tasks.

A Brief Summary

- SDVT = Subtask Decomposition (SD) and Virtual Training (VT).
- SDVT without a GMVAE and with a single Gaussian VAE reduces to LDM.
- LDM without virtual training reduces to VariBAD.
- VariBAD without a dynamics decoder reduces to RL^2.

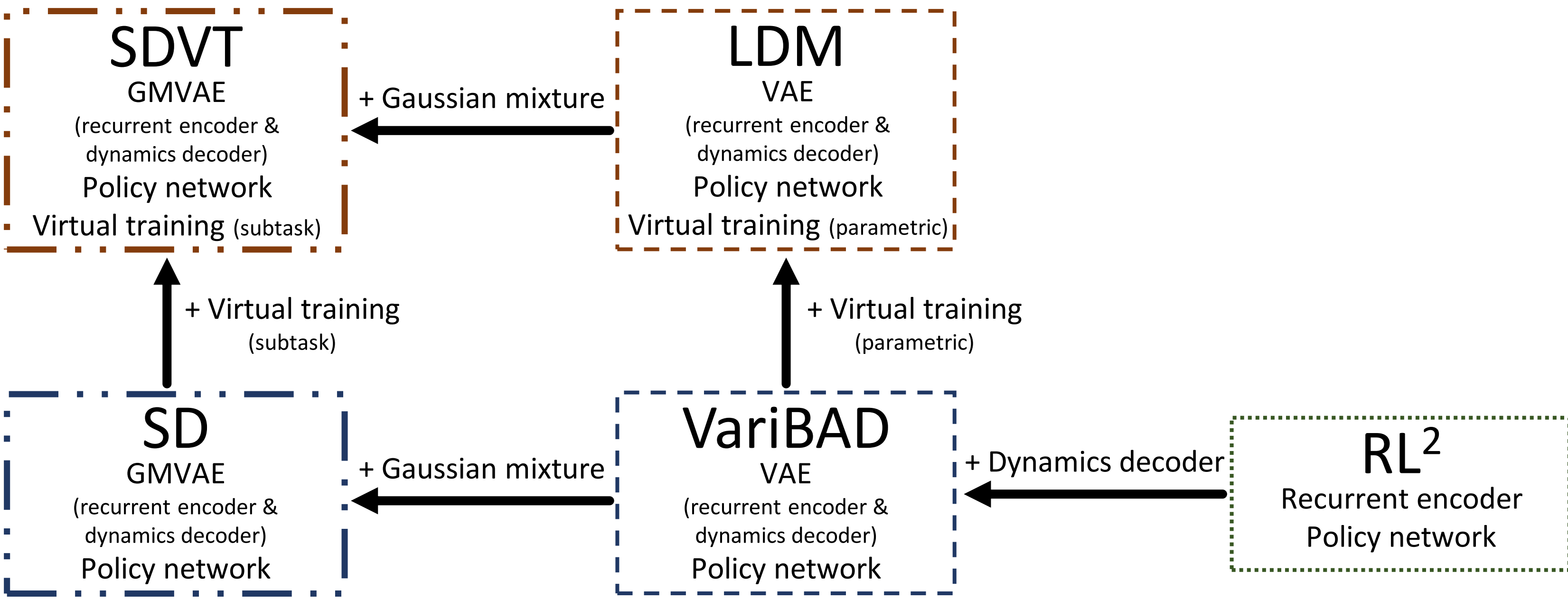


Figure 32: Schematic overview of proposed algorithms and baselines.

Experiments – Setup

- The Meta-World V2 benchmark
  - A benchmark with 50 qualitatively distinct robotic manipulation tasks.
  - **ML-10** benchmark: $N_{\text{train}} = 10$ training tasks and $N_{\text{test}} = 5$ test tasks.
  - **ML-45** benchmark: $N_{\text{train}} = 45$ training tasks and $N_{\text{test}} = 5$ test tasks.

- Our methods
  - SDVT: subtask decomposition and virtual training.
  - SD: subtask decomposition without virtual training.
  - SDVT-LW and SD-LW (Light Weight variants): ours without the occupancy regularization ($\alpha_o = 0$), assuming that we know the optimal number of elementary subtasks $K^* = 5$. The default setup is $K = N_{\text{train}}$ with occupancy regularization ($\alpha_o = 1$).

- Evaluation metric
  - $N = 10$ rollout episodes, $H = 500$ steps per rollout episode.
  - We evaluate the performance at the last rollout episode in terms of the success rate and the episode return.

Experiments – Result

- On the test tasks, SDVT and SDVT-LW substantially outperform all baselines.
- SDVT's gain over LDM is attributed to our virtual training process, which is specifically designed for test tasks involving non-parametric variability.

Table 2: Meta-World V2 success rates and returns. We report the final success rates (%) and returns of our methods and baselines averaged across training tasks and test tasks of the ML-10 and ML-45 benchmarks. All results are reported as the mean success rate ± 95% confidence interval of 8 seeds.

	Success Rate				Return			
	ML-10		ML-45		ML-10		ML-45	
Methods	Train	Test	Train	Test	Train	Test	Train	Test
SDVT	**77.2**±**3.0**	32.8±3.9	55.6±4.2	28.1±3.2	**3656**±**62**	1225±160	2379±214	839±74
SDVT-LW	62.1±4.1	**33.4**±**5.0**	50.4±4.1	**31.2**±**1.2**	3454±137	**1527**±**214**	2294±202	**894**±**27**
SD	77.0±5.9	30.8±7.7	**61.0**±**1.7**	23.0±5.1	3630±241	1112±190	**2672**±**79**	786±69
SD-LW	75.5±5.5	26.2±8.7	56.7±1.5	25.4±2.9	3525±297	1043±234	2578±64	793±49
RL^2	67.4±4.4	15.1±2.7	58.0±0.4	11.8±3.2	1159±83	715±33	1411±22	663±100
MAML	42.2±4.5	3.9±3.7	32.0±1.4	19.8±6.3	1822±136	439±78	1388±104	658±96
PEARL	23.2±1.9	0.8±0.5	10.3±2.4	6.7±3.3	1081±77	340±54	597±121	506±122
VariBAD	58.2±8.9	14.1±6.1	57.0±1.2	22.1±3.5	3055±466	919±143	2492±47	762±40
LDM	56.7±12.3	19.8±6.0	54.1±0.9	24.8±2.9	2963±626	1166±264	2515±67	768±63

Experiments – Result

- Our methods achieve the highest success rates across all test tasks on the ML-10 benchmark.
- However all methods are challenged by the tasks "Shelf-place" and "Lever-pull".
  - They include unseen objects not included in the raw observation.
  - These tasks cannot be decomposed into previously seen subtasks but rather require new elementary subtasks.
- Link to demo videos: `https://sites.google.com/view/sdvt-neurips`.

Table 3: Meta-World V2 ML-10 success rates on test tasks.

Index. Task	SDVT	SDVT-LW	SD	SD-LW	RL^2	MAML	PEARL	VariBAD	LDM
11. Drawer-open	**65.0±19.9**	30.5±12.9	48.8±23.4	45.0±24.0	2.2±1.9	15.8±19.3	1.5±1.1	12.8±12.8	21.8±12.0
12. Door-close	7.5±9.0	**81.2±19.0**	33.8±25.4	18.8±24.1	8.2±7.3	3.2±6.0	1.2±1.5	27.0±21.8	30.2±28.2
13. Shelf-place	0.0±0.0	**1.0±1.2**	0.0±0.0	0.0±0.0	0.2±0.2	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
14. Sweep-into	**90.0±8.5**	51.2±18.9	71.2±14.9	55.0±21.6	64.5±8.3	0.0±0.0	0.8±1.0	30.5±22.0	46.5±21.8
15. Lever-pull	1.2±2.3	3.2±2.7	0.0±0.0	**12.5±10.2**	0.5±0.8	0.5±0.9	0.5±0.9	0.2±0.5	0.5±0.9
Test mean	32.8±3.9	**33.4±5.0**	30.8±7.7	26.2±8.7	15.1±2.7	3.9±3.7	0.8±0.5	14.1±6.1	19.8±6.0

Limitation

- Test tasks involving entirely novel subtasks
  - SDVT can prepare for unseen compositions of seen subtasks.
  - However, SDVT does not prepare for unseen compositions of unseen subtasks.
- Limited to scenarios with varying rewards
  - SDVT does not consider setups where transition dynamics, action, and state space may vary.
- No temporal information of subtasks
  - Currently the subtask composition is a belief about how the current meta-episode's task is decomposed into.
  - It does not consider temporal information of subtasks, such as the ordering or start and termination of subtasks.

Conclusion

1. Latent Dynamics Mixture (LDM): Parametric task variability
   - A robust meta-RL algorithm that effectively prepares for potential out-of-distribution test tasks.
   - LDM pretrains the policy on imaginary tasks generated from mixtures of latent beliefs on training tasks.
   - LDM effectively enhances generalization to unseen tasks without additional test-time training.
2. Subtask Decomposition and Virtual Training (SDVT): Non-parametric task variability
   - SDVT generalizes LDM for scenarios with non-parametric task variability.
   - SDVT employs a Gaussian mixture VAE to meta-learn the set of elementary subtasks and the composition of each task.
   - SDVT extends the idea of virtual training from LDM to generate tasks with unseen compositions of subtasks.
3. Both methods demonstrate the efficacy of virtual training with generated imaginary tasks.
   - We may extend the ideas to more general scenarios with varying transition dynamics, state space, or action space.
   - We may combine it with orthogonal strategies such as offline RL or test-time adaptation techniques.

Thank you!

Appendix

Bayes-Adaptive Belief Update

- Posterior belief update

$$b_{t+1}\left(R^{(i)}, T^{(i)}\right) = \frac{\mathbb{P}\left(s_{t+1}, r_{t+1} \mid s_t, a_t, R^{(i)}, T^{(i)}\right)}{\sum_{k=1}^{m} \mathbb{P}\left(s_{t+1}, r_{t+1} \mid s_t, a_t, R^{(k)}, T^{(k)}\right) b_t\left(R^{(k)}, T^{(k)}\right)} b_t\left(R^{(i)}, T^{(i)}\right). \quad (26)$$

$$\mathbb{P}\left(s_{t+1}, r_{t+1} \mid s_t, a_t, R^{(i)}, T^{(i)}\right) b_t\left(R^{(i)}, T^{(i)}\right) = \mathbb{P}\left(s_{t+1}, r_{t+1} | \tau_{:t}, R^{(i)}, T^{(i)}\right) \mathbb{P}\left(R^{(i)}, T^{(i)} \mid \tau_{:t}\right) \quad (27)$$

$$= \frac{\mathbb{P}\left(s_{t+1}, r_{t+1}, \tau_{:t}, R^{(i)}, T^{(i)}\right)}{\mathbb{P}\left(R^{(i)}, T^{(i)}, \tau_{:t}\right)} \frac{\mathbb{P}\left(R^{(i)}, T^{(i)}, \tau_{:t}\right)}{\mathbb{P}\left(\tau_{:t}\right)} \quad (28)$$

$$= \frac{\mathbb{P}\left(\tau_{:t+1}, R^{(i)}, T^{(i)}\right)}{\mathbb{P}\left(\tau_{:t}\right)} \quad (29)$$

$$= \mathbb{P}\left(R^{(i)}, T^{(i)} \mid \tau_{:t+1}\right) \frac{\mathbb{P}\left(\tau_{:t+1}\right)}{\mathbb{P}\left(\tau_{:t}\right)} \quad (30)$$

$$= b_{t+1}\left(R^{(i)}, T^{(i)}\right) \mathbb{P}\left(s_{t+1}, r_{t+1} \mid \tau_{:t}\right). \quad (31)$$

SDVT ELBO Derivation

- GMVAE ELBO Derivation

$$\log p_\theta(\tau_{:H^+}) = \log \mathbb{E}_{q_\phi(y_t,z_t|h_t)}\left[\frac{p_\theta(\tau_{:H^+}, y_t, z_t)}{q_\phi(y_t, z_t|h_t)}\right] \tag{32}$$

$$\geq \mathbb{E}_{q_\phi(y_t,z_t|h_t)}\left[\log \frac{p_\theta(\tau_{:H^+}, y_t, z_t)}{q_\phi(y_t, z_t|h_t)}\right] \tag{33}$$

$$= \mathbb{E}_{q_\phi(y_t,z_t|h_t)}\left[\log \frac{p_\theta(\tau_{:H^+}|y_t, z_t)p_\theta(z_t|y_t)p(y_t)}{q_\phi(y_t|h_t)q_\phi(z_t|h_t, y_t)}\right] \tag{34}$$

$$= \mathbb{E}_{q_\phi(y_t,z_t|h_t)}\left[\log p_\theta(\tau_{:H^+}|z_t) + \log \frac{p_\theta(z_t|y_t)}{q_\phi(z_t|h_t, y_t)} + \log \frac{p(y_t)}{q_\phi(y_t|h_t)}\right]. \tag{35}$$

Equation (35) is equivalent to the ELBO objective in Equation (17) without weighting coefficients. We assume that the reconstruction $\tau_{:H^+}$ is conditionally independent of the subtask composition y_t given z_t.

Dropout Ablation – LDM

- Dropout rate of LDM
  - Upward trajectory in test performance in correlation with an increase in the dropout rate.
  - However, this trend reverses when the rate approaches its upper limit, specifically at $p_{\text{drop}} = 0.9$.

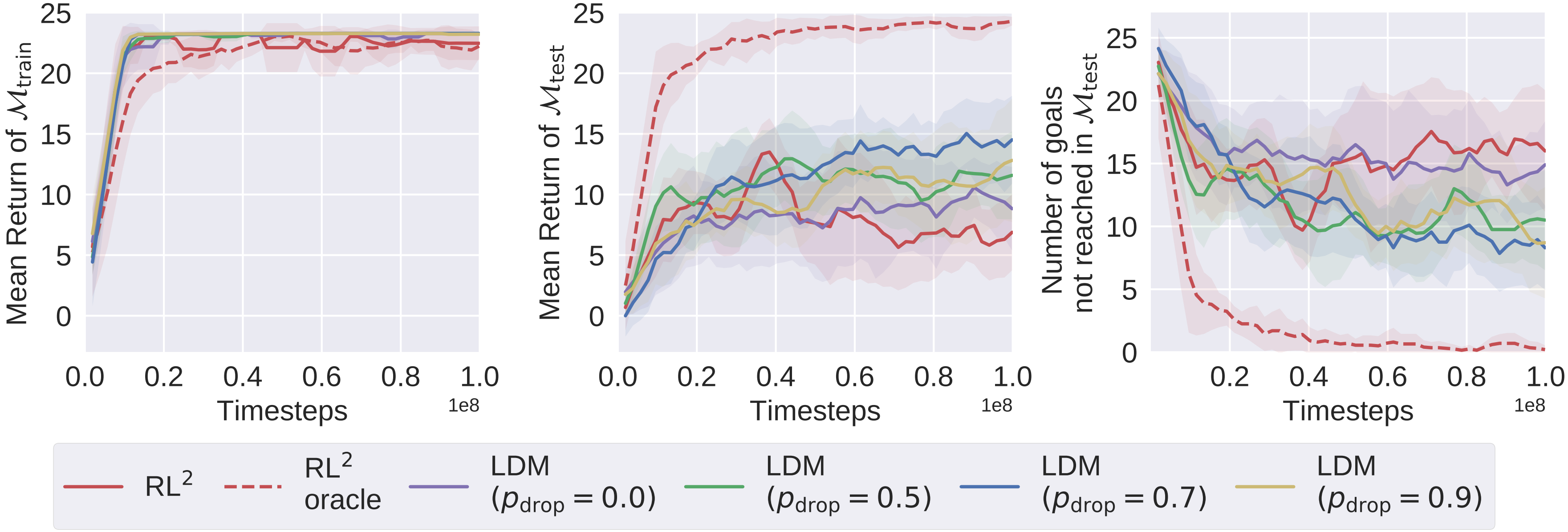


Figure 33: Varying dropout rate. Results of LDM with different dropout rates on Gridworld task.

Dropout Ablation – Baselines

- We assess the performance of both RL^2 dropout and VariBAD dropout,
  - Dropout destabilizes policy training with multi-step policy gradient loss
  - Dropout on the decoder input improves generalization due to the relatively simpler single-step regression loss with the buffer updates.

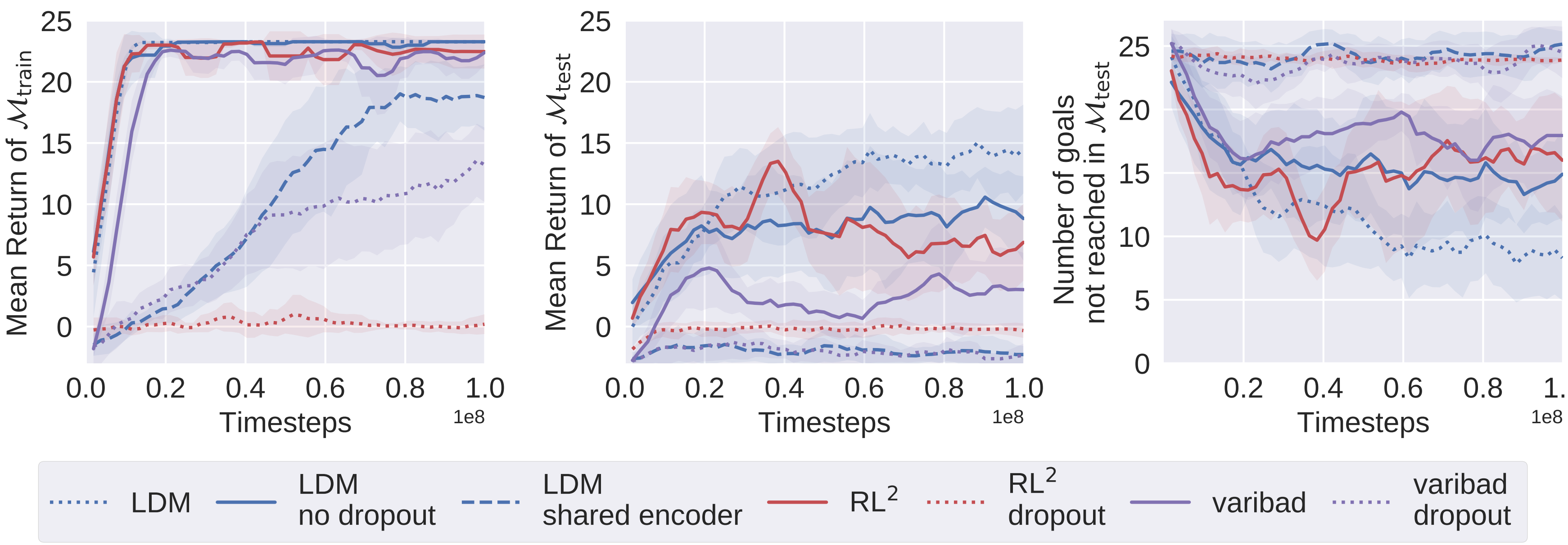
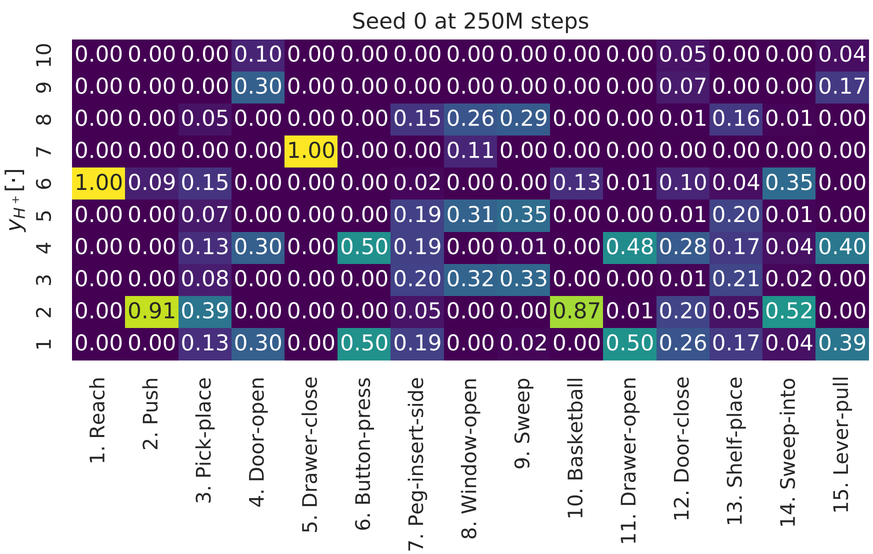


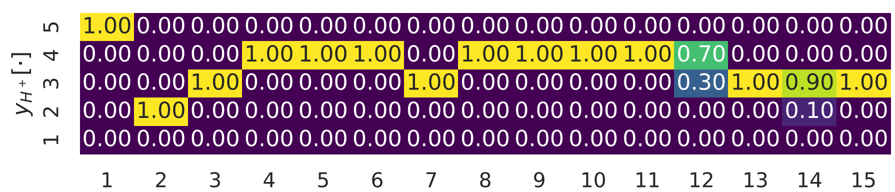
Figure 34: Dropout ablation. Results of LDM, RL^2, and VariBAD trained with and without dropout on Gridworld task.

SDVT Ablation

- The default setup of our methods are
  - SDVT: $K = 10$, $\alpha_c = 1.0$, $\alpha_0 = 1.0$,
  - SDVT-LW: $K = 5$, $\alpha_c = 0.5$, $\alpha_0 = 0.0$.
- When α_c is too small, task classification collapses into a few one-hot subtasks.
- When α_c is too large, all tasks return a uniform probability distribution over subtasks.



(a) SDVT ($K = 10, \alpha_c = 1.0, \alpha_o = 1.0$).



(b) SDVT-LW ($K = 5, \alpha_c = 0.1, \alpha_o = 0.0$).

(c) SDVT-LW ($K = 5, \alpha_c = 2.0, \alpha_o = 0.0$).

Figure 35: Learned subtask compositions on ML-10 with different hyperparameters.

SDVT Ablation

- We report the mean success rates by changing hyperparameters from the default SDVT-LW $(K = 5, \alpha_c = 0.5, \alpha_o = 0.0)$ on ML-10.
- We report the difference caused by the changes in parenthesis.

Table 4: Ablation results of SDVT-LW. Performance measured as test success rate (%) on ML-10.

SDVT-LW	without		K			α_c		
	Dropout	Dispersion	3	7	10	0.1	1.0	2.0
ML-10 Train	65.6 (+3.5)	73.0 (+10.9)	60.1 (-2.0)	69.5 (+7.4)	70.8 (+8.7)	59.8 (-2.3)	70.3 (+8.2)	66.3 (+4.2)
ML-10 Test	16.5 (-16.9)	21.5 (-11.9)	25.0 (-8.4)	22.0 (-11.4)	21.1 (-12.3)	20.5 (-12.9)	16.1 (-17.3)	17.9 (-15.5)

SDVT Learned Compositions – Different Seeds

- Decomposition processes differ among random seeds due to varying initialization and sample tasks during meta-training.
- One subtask can be interpreted as a combination of multiple subtasks on other seeds.
- We rarely have a collapse to a single Gaussian.

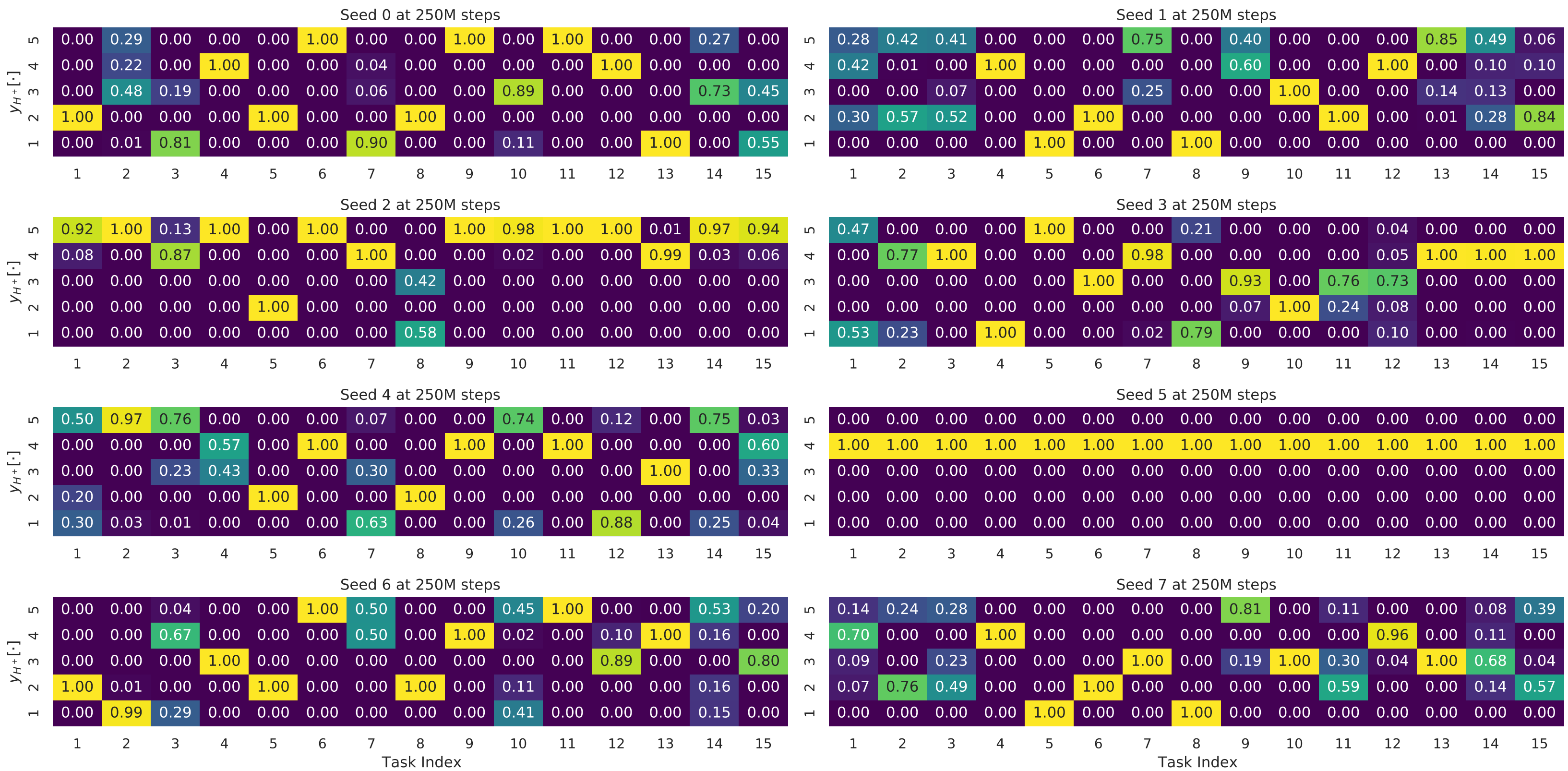


Figure 36: Subtask compositions of all seeds. We visualize the subtask compositions of SDVT-LW on ML-10 after 250M training steps.

Learned Compositions – Over Time

- Learned compositions over the course of training (0M $\sim$ 250M steps).
- The composition starts with a uniform distribution for all tasks.
- As training progresses, the agent learns to merge similar tasks.

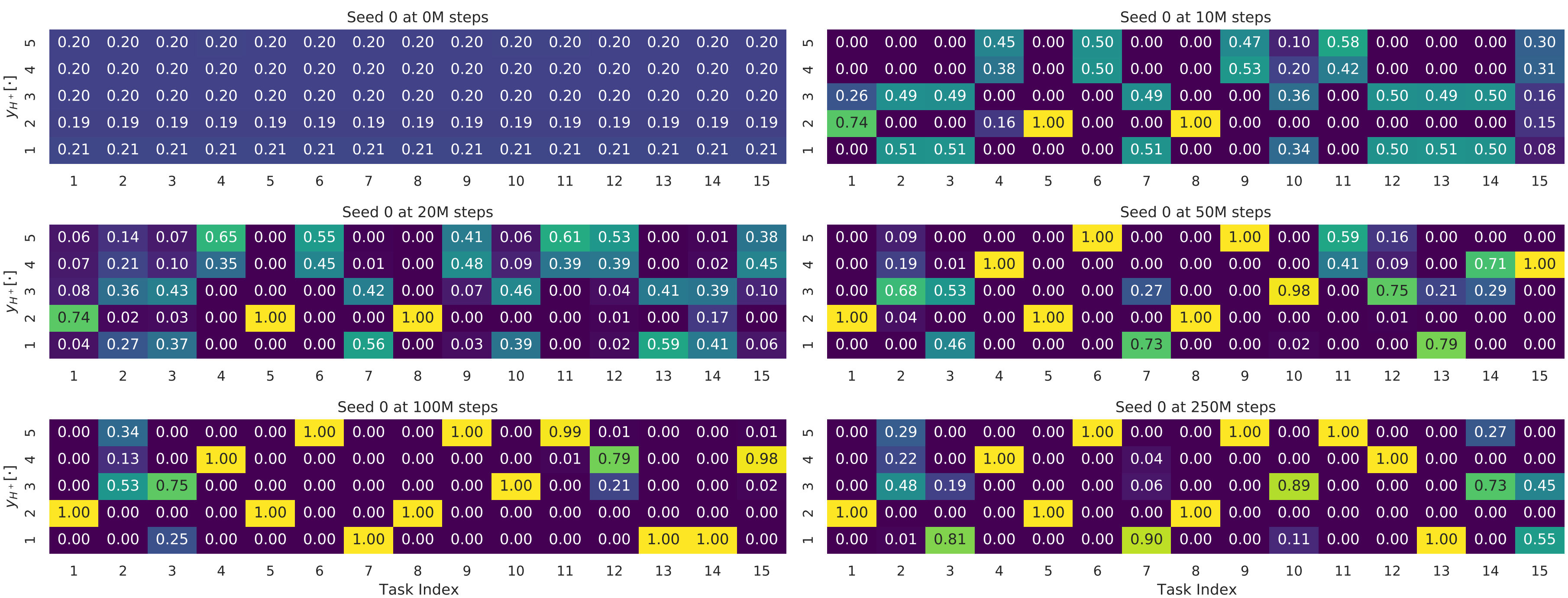


Figure 37: Subtask compositions learned over training. SDVT-LW $(K = 5, \alpha_c = 0.5, \alpha_o = 0.0)$ on ML-10.

Number of Parameters

- We report the number of parameters used by our methods and baselines.
- We demonstrate that our gain is not mainly from the increased capacity.
  - VariBAD (hidden $\times 2$) with hidden dimensions of 512 possesses more parameters than SDVT.
  - We add MLP layers with hidden sizes of [1600, 256] into the encoder and [128] into the policy. The decoder's hidden size is increased from [64, 64, 32] to [128, 256, 160] to match the capacities of VariBAD (matched) and SDVT-LW.

Table 5: Number of parameters and success rates on ML-10.

	Number of Parameters				ML-10 Success Rate (%)	
Methods	Encoder	Decoder	Policy	Sum	Train	Test
SDVT-LW	1,047,455	174,821	235,401	1,457,677	62.1	**33.4**
SD-LW	1,047,455	25,637	235,401	1,308,493	**75.5**	26.2
LDM	502,580	25,577	202,249	730,406	56.7	19.8
LDM (matched)	2,144,692	175,593	267,401	2,587,686	64.2	22.0
VariBAD	251,290	25,577	202,249	479,116	58.2	14.1
VariBAD (hidden $\times 2$)	894,362	25,577	663,305	1,583,244	62.4	12.0
VariBAD (matched)	1,072,346	175,593	267,401	1,515,340	67.6	17.2

Computational Complexity

- We report the total wall-clock time required to generate the results for the Half-cheetah-velocity.
  - LDM requires more time than RL^2, due to more complex architecture that involves the simultaneous training of both the policy network and an independent latent dynamics network.

Table 6: Computation complexity on Half-cheetah-velocity.

	LDM	Mixreg	RL^2	VariBAD	ProMP	E-MAML	PEARL
Half-cheetah-velocity Runtime (hours)	31	28	25	10	2	2	25

- We report the total wall-clock time required to generate the results for the ML-10,
  - Despite incorporating the GMVAE and virtual training, our method's computational demand does not substantially surpass that of VariBAD.

Table 7: Computational complexity on ML-10.

	SDVT	SDVT-LW	SD	SD-LW	RL^2	MAML	PEARL	VariBAD	LDM
Wall-clock time (hours)	142	140	138	135	192	17	258	126	131