Learning Task-Aware Abstract Representations for Meta-Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

A central challenge in meta-reinforcement learning (meta-RL) is enabling agents trained on a set of environments to generalize to new, related tasks without requiring full policy retraining. Existing model-free approaches often rely on context-conditioned policies learned via encoder networks. However, these context encoders are prone to overfitting to the training environments, resulting in poor out-of-sample performance on unseen tasks. To address this issue, we adopt an alternative approach that uses an abstract representation model to learn augmented, task-aware abstract states. We achieve this by introducing a novel architecture that offers greater flexibility than existing recurrent network-based approaches. In addition, we optimize our model with multiple loss terms that encourage predictive, task-aware representations in the abstract state space. Our method simplifies the learning problem and provides a flexible framework that can be readily combined with any off-the-shelf reinforcement learning algorithm. We provide theoretical guarantees alongside empirical results, showing strong generalization performance across classical control and robotic meta-RL benchmarks, on par with state-of-the-art meta-RL methods and significantly better than non-meta RL approaches.

1 Introduction

Achieving robust generalization to new environments is challenging for agents trained within a single setting. While such agents may perform well in the training environment, they often fail when faced with even minor changes to dynamics. Meta-reinforcement learning (meta-RL) addresses this limitation by training across a distribution of tasks, thereby enabling the few-shot adaptation of classical RL agents (Beck et al., 2025; Nagabandi et al., 2018).

One of the most promising directions in meta-RL is context conditioning (Zintgraf et al., 2021; Lee et al., 2020). The core idea is to learn a context representation that captures task-specific information from a distribution of environments. At test time, the agent can explicitly infer the context of a new, unseen environment and adapt its behavior accordingly. Recent work (Lee et al., 2020; Rimon et al., 2024) has shown that context-conditioned dynamics models are particularly effective to meta-RL settings.

A key limitation of context-based meta-RL approaches is that the agent's performance depends heavily on the encoder's ability to accurately distinguish between tasks. Consequently, if the context encoder overfits to the training distribution - or fails to generalize - its out-of-sample performance can degrade substantially (Zintgraf et al., 2021).

This work aims to improve generalization in meta-RL by leveraging abstract representation models (ARMs), which learn high-level representations of environmental states using a pretrained model. Our goal is to produce task-aware, augmented abstract state representations. We obtain task-aware states with a neural architecture that encodes states into an abstract space through a state-space encoder and infers task encodings with a recurrent neural network (RNN). These representations are optimized jointly using a set of complementary loss functions. Our approach offers two main advantages: (1) ARMs are straightforward to train and generalize well, reducing reliance on the context encoder; and (2) they integrate seamlessly

with any off-the-shelf model-free reinforcement-learning algorithm, giving the agent direct access to task information.

Our network architecture integrates abstract state representation learning with task inference in a unified framework. Unlike previous approaches that rely primarily on task-encoders that are trained separately, we introduce a joint optimization framework based on predictive losses in the abstract space—capturing transitions, rewards, and task structure.

We call our methodology **E**nvironment-aware **M**eta **E**ncoding and **R**epresentation **A**bstraction for **L**atent **D**omains—**EMERALD**. It is a novel reinforcement learning framework that learns *task-aware abstract representations* capturing both environmental dynamics and task-specific factors, enabling improved generalization and task disambiguation in meta-RL settings. Our key contributions are:

- EMERALD's performance is on par with *state-of-the-art performance* meta-RL methods across a suite of meta-RL benchmarks and outperforms classical RL approaches in *few-shot adaptation and transfer* tasks, and is compatible with standard reinforcement learning algorithms such as SAC and PPO.
- We provide *theoretical justification* for EMERALD's design, showing how incorporating contextual information into the abstract state space promotes fast generalization to unseen tasks.

2 Related Work

Meta Reinforcement Learning Meta-reinforcement learning (meta-RL) enables RL agents to rapidly adapt to new, unseen tasks or environments (Beck et al., 2025; Nagabandi et al., 2018). The goal is to learn a policy trained on a set of tasks $\mathcal{T}_i \sim p(\mathcal{T}_{\text{train}})$ that can quickly adapt to new tasks. Achieving this requires the agent to generalize effectively across tasks, leveraging experience from the training distribution to perform well in novel settings (Finn et al., 2017). Meta-RL can also be viewed through the lens of partially observable Markov decision processes (POMDPs), where the true task identity is unobserved and must be inferred through interaction, making context inference analogous to belief state estimation (Humplik et al., 2019; Rakelly et al., 2019).

Context-based Meta-RL Context-based meta-RL approaches learn a context encoder to capture variations across tasks. PEARL (Rakelly et al., 2019) and VariBAD (Zintgraf et al., 2021) formulate the context-encoding problem using a Bayesian Adaptive MDP (BAMDP), where a context variable z is modeled as a belief state inferred from past transitions and used to condition the policy, yielding $\pi(s_t \mid z)$. VariBAD passes an augmented state—consisting of the current state and the belief variable—to the policy. This allows the agent to adapt its behavior based on the inferred task identity. We build on this idea of an augmented state space, but rather than concatenating the state and context, we feed the learned context variable directly into the state-space encoder of the representation model.

Model-based Contextual Meta-RL MAMBA (Rimon et al., 2024) extends VariBAD to the model-based setting by integrating context encoding into Dreamer's latent imagination process. A related method, CaDM (Lee et al., 2020), learns contextualized dynamics models by conditioning both the reward and transition models on context, and employs a backward-and-forward transition mechanism to stabilize training. Our approach builds on these methods, but uses only forward predictions in the abstract space. As in Dreamer-based methods such as MAMBA, we allow the policy to operate in an abstract space with lower cardinality than the raw state space, $|\mathcal{X}| \ll |\mathcal{S} \times \mathcal{T}|$. However, our state–task encoder can also be deployed in model-free settings via abstract representation models. Figure 1 illustrates the high-level architectural differences between our approach and other context-based methods.

Abstract Representation Models in Deep RL Abstract representation models (ARMs) are widely used in deep reinforcement learning to improve generalization and sample efficiency (Botteghi et al., 2022; Starre et al., 2022; Ni et al., 2024). These models map high-dimensional states $\mathcal{S}$ to compact latent spaces $\mathcal{X}$ via encoders $\psi : \mathcal{S} \to \mathcal{X}$, enabling more structured learning. Unlike "standard" auto-encoders, abstract representation models in the context of RL are typically used to obtain generalizable abstractions of the reward and transition dynamics. Several works illustrate this: for instance, Zhang et al. (2018) decouple

dynamics and rewards for domain generalization, while Li et al. (2021) apply abstract models to adversarial settings with shared dynamics but varying visual observations. We build on these insights and extend ARMs to the meta-RL setting. Similar to prior combined RL approaches (François-Lavet et al., 2019; Lee et al., 2020), our method is flexible and can be integrated with both model-free and model-based agents.

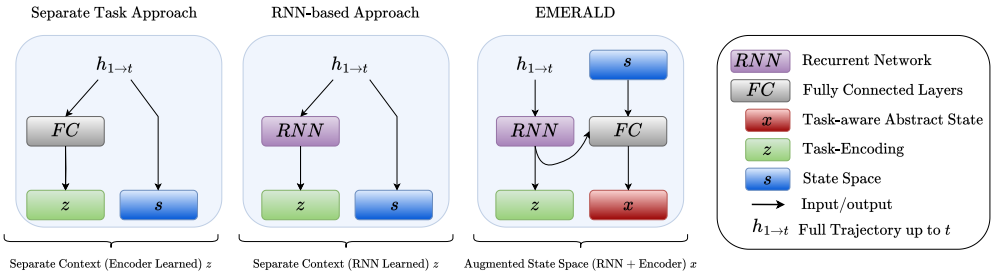


Figure 1: Comparison of state and context representation strategies. **Left:** Methods such as CaDM learn state (s) and context (z) representations independently, without temporal modeling. **Middle:** VariBAD-like approaches use RNNs to learn z from trajectory history $h_{1\to t}$, but still treat s and z as separate components. **Right:** Our proposed EMERALD ARM architecture integrates the trajectory-derived context z with the current abstract state to form a task-aware shared latent state representation x, enabling unified and task-aware modeling of the environment dynamics.

3 Task-aware Abstract Representation Models

Notation We define a task $\mathcal{T}_i$ as a Markov Decision Process (MDP), represented by the 5-tuple $\mathcal{T}_i = (\mathcal{S}, \mathcal{A}, P_i, r_i, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $P_i(s_{t+1} \mid s_t, a_t)$ the transition kernel, and r_i the reward function mapping state-action pairs to real-valued rewards. The discount factor is denoted $\gamma \in [0, 1)$. We index time steps by the subscript t, e.g., s_t and s_{t+1}.

We sample N independent and identically distributed (i.i.d.) training tasks $\mathcal{T}_i \sim p(\mathcal{T})$. From a task, we can collect data sets of transitions $\mathcal{D}_i = \{(s_{k,t}, a_{k,t}, s_{k,t+1}, r_{k,t})\}_{k=1}^{m_i}$. The combined dataset across all N tasks is denoted $\mathcal{D} = \bigcup_{i=1}^{N} \mathcal{D}_i$. We let $m_i = |\mathcal{D}_i|$ denote the number of transitions collected from task $\mathcal{T}_i$. We denote the entropy of a random variable X as $\mathbb{H}(X)$, and the conditional entropy as $\mathbb{H}(X \mid Z)$. $\mathcal{L}_i(\theta)$ denotes the loss of a model with parameters θ in a single environment i and $\mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}\mathcal{L}_{\mathcal{T}}(\theta)$ the loss over all tasks in the distribution.

Model Components The abstract *state* space is denoted as $\mathcal{X}$ and the *task* embedding space as $\mathcal{Z}$. Note that, in our architecture, $\mathcal{X}$ contains both the information about the task as well as the state in that task. The ARM has four components: (1) the contextual state-space encoder $\psi : \mathcal{S} \times \mathcal{Z} \to \mathcal{X}$, (2) the transition function $\tau : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$, (3) the reward predictor $\rho : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$. Lastly, (4) a learned encoder $\phi : (\mathcal{S} \times \mathcal{A} \times \mathbb{R})^{t-1} \times \mathcal{S} \to \mathcal{Z}$, mapping a history of transitions to an environment embedding $\mathcal{Z} \in \mathbb{R}^k$: $z_i = \phi(h_{0\to t-1})$, where $h_{0\to t-1} = (s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{t-1}, a_{t-1}, r_t, s_t)$. Figure 1 (right) shows a schematic overview of the model architecture.

Policy We denote the policy as $\pi : \mathcal{X} \to \mathcal{A}$. The *value function* $V^\pi(x)$ is defined as the expected return when starting from abstract state x and following policy π:

$$V^\pi(x) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, x_0 = x \right].$$

The *Q-function* $Q^\pi(x, a)$ denotes the expected return when taking action a in abstract state x, and thereafter following π:

$$Q^\pi(x, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, x_0 = x, a_0 = a \right].$$

Goal The aim is to learn a policy that maximizes the expected return across tasks sampled from the task distribution $p(\mathcal{T})$. Formally, we optimize:

$$\max_{\pi,\theta} \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}\left[\mathbb{E}_{(x_0,a_0,r_1,\ldots)\sim\pi,\mathcal{T}}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right]\right],$$

where the expectation is taken over both the task distribution and the rollout distribution induced by the policy π across a task distribution, operating in the abstract state space $\mathcal{X}$.

3.1 Task-Aware Augmented State-Representations in Meta-RL

Necessity of task-aware ARMs in meta-RL Despite the capacity of representation models to generalize well in simple meta-RL settings such as a variety of different mazes, we argue that only through the addition of task-specific information can their potential be fully realized, a feat well-established in standard meta-RL literature Beck et al. (2025); Ni et al. (2024). Within our framework specifically, the intuition is that without task-awareness (measured by how much task information is retained by the encoder), the abstract representation model will not be able to learn a representation that can differentiate based solely on the input it receives. As such, task-agnostic state-space encoders essentially introduce a many-to-one mapping, which collapses into a single point during optimization. We provide an illustration of this insight in Figure 2.

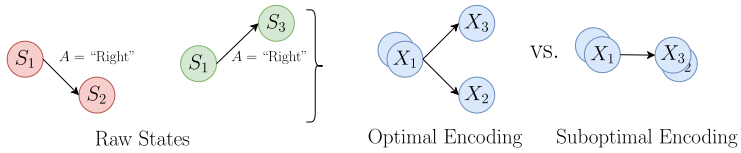


Figure 2: Illustration of the main intuition that one-to-one task-aware mappings produce better encodings. Raw input states, where S_1 maps to state S_2 in task red and to S_3 in task green. **Left:** Optimal encoding where both follow-up abstract states are mapped to different regions in space. **Right:** Suboptimal encoding we would expect if the encoder does not retain contextual information. In that case, we may observe a collapse of the two states, because $\psi(s_3^{\mathcal{T}_1})$ and $\psi(s_2^{\mathcal{T}_2})$ are pushed to the same point in order to minimize the loss.

How can task-awareness be achieved? We introduce a novel approach that integrates task information directly into the abstract state space $\mathcal{X}$ via the encoder ψ, producing representations to be *both* compact *and* task-relevant. Unlike prior meta-RL methods that condition downstream components (such as dynamics, reward models, or policies) on context (e.g., VariBAD, PEARL), our method performs *early* context injection by embedding task identity during the state abstraction phase. This design allows our agent to resolve task ambiguity upstream, providing a strong inductive bias and leading to more robust representations that generalize better to unseen tasks.

By shifting task uncertainty into the encoder, we also simplify the overall architecture and training objective, eliminating the need for amortized inference or latent variable optimization at test time. Furthermore, this abstraction-first formulation is highly modular and integrates seamlessly with both model-based and model-free RL algorithms. We formally justify this design through the lens of entangled versus disentangled modeling: in the latter, state abstraction and task representation are learned separately, whereas in the former — our setting — they are integrated into a single representation. We then empirically show that our approach leads to improved generalization across standard meta-RL benchmarks.

Proposition 1 (Entanglement Yields Lower Entropy) *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two tasks such that for some state-action pair (s_t, a_t), the transitions differ: $P_1(s_{t+1} \mid s_t, a_t) \neq P_2(s_{t+1} \mid s_t, a_t)$. Let $z = \phi(h_{i,:t}) \in \mathcal{Z}$ be a deterministic task embedding computed from transition history. Consider two model variants: (1) a task-state encoder, where $x_t = \psi(s_t, z)$, the transition prediction is given by $x_{t+1} \approx x_t + \tau(x_t, a_t)$ and the estimated reward $\hat{r}_t = \rho(x_t, a_t)$; and (2) a state encoder, where $x_t = \psi^*(s_t)$, $x_{t+1} \approx x_t + \tau^*(x_t, a_t, z)$ and the*

estimated reward $\hat{r}_t = \rho^*(x_t, a_t, z)$*. Then, assuming* ψ*,* ρ *and* τ *as well as* ψ^**,* ρ^* *and* τ^* *are deterministic and trained to perfectly fit the transition and reward losses, the total entropy of the representations satisfies:*

$$\mathbb{H}\big(\psi(s_t, z)\big) \;\leq\; \mathbb{H}\big((\psi^*(s_t), z)\big).$$

That is, the task-state version yields lower joint entropy due to resolving task ambiguity in the representation, resulting in simpler transitions and reward functions.

We provide the full proof in Appendix C. Intuitively, Proposition 1 states that entanglement causes context ambiguity to be resolved *before* it reaches downstream tasks (e.g., before computing transition probabilities or state-to-action mappings via the policy). As such, following a general "conditioning reduces entropy" rule (Cover & Thomas, 1999; Gray, 2011; MacKay, 2003), which has been explored extensively in the machine learning literature (Bounoua et al., 2025; Pandey & Dukkipati, 2017; Shamir et al., 2010), early conditioning effectively ensures that all model components are one-to-one mappings. We call the task-aware abstract state space produced by the the state-task encoder the *augmented abstract* state space.

4 EMERALD: Architecture & Objectives

ARM: Architecture Blocks We introduce a parameterized abstract representation model, consisting of (1) a state-space encoder ψ, (2) a latent transition function τ, (3) a reward predictor ρ, and (4) a transitions-to-context function ϕ to adjust for environment-specific dynamics (see Appendix G for a more detailed overview). Formally, we have four parameterized model components:

$$\psi\big(s_t, z; \theta_\psi\big) \;\to\; x_t, \quad x_{t+1} \;=\; x_t \;+\; \tau\big(x_t,\, a_t;\, \theta_\tau\big), \quad \hat{r}_t \;=\; \rho\big(x_t,\, a_t;\, \theta_\rho\big), \quad z \;=\; \phi\big(h_t;\, \theta_\phi\big).$$

We combine the model parameters into a jointly optimizable model via θ, where $\theta = (\theta_\psi,\, \theta_\tau,\, \theta_\rho,\, \theta_\phi)$.

ARM: Objectives The transition and reward dynamics are effectively captured by the loss functions defined below. We define the transition and reward losses as:

$$\mathcal{L}(\theta)_{\text{transition}} = \mathbb{E}_{(s_t, a_t, s_{t+1}, r_t)\sim D}\Big[\big\|\psi\big(s_{t+1}, z; \theta_\psi\big) - \big(\psi\big(s_t, z; \theta_\psi\big) + \tau\big(\psi(s_t, z; \theta_\psi), a_t; \theta_\tau\big)\big)\big\|^2\Big],$$

$$\mathcal{L}(\theta)_{\text{reward}} = \mathbb{E}_{(s_t, a_t, s_{t+1}, r_t)\sim D}\Big[\big\|r_t - \rho\big(\psi(s_t, z; \theta_\psi), a_t; \theta_\rho\big)\big\|^2\Big].$$

The formulations above follow standard practice (e.g., François-Lavet et al. (2019)), but we introduce context vectors z retrieved from the context encoder ϕ and include it directly into the optimization objective. To prevent a potential state space collapse, we also include a regularizer:

$$\mathcal{L}(\theta)_{\text{reg}} = \exp\Big(-C_d\big\|\psi(s^+, z; \theta_\psi) - \psi(s^-, z; \theta_\psi)\big\|^2\Big),$$

where s^+ and s^- are two randomly sampled states and C_d is a constant. Intuitively, this term enforces that any two states are some distance apart.

Putting everything together, we obtain the following joint objective:

$$\mathcal{L}(\theta) \;=\; \mathcal{L}(\theta)_{\text{transition}} \;+\; \mathcal{L}(\theta)_{\text{reward}} \;+\; \beta\,\mathcal{L}(\theta)_{\text{reg}},$$

where $\beta \geq 0$ determines the regularization strength. We minimize this objective using standard mean squared error losses for the transition and reward terms. Algorithm 1 shows the pseudocode for the ARM training process. Throughout the experimental section, we sample the offline data for training (line 1 of the algorithm) the model using a random policy. However, in practice, the offline data might also be collected via a non-random or expert policy. A Python implementation can be found at `https://anonymous.4open.science/r/EMERALD-F324`.

Training of the policy The policy is trained after ARM training. We freeze the representation model's parameters and pass the online transitions through the model. The augmented abstract state approach allows us to combine the learned representations with any reinforcement learning algorithm (e.g., SAC, PPO). We define the optimal policy as:

$$\pi^* = \arg\max_{\pi} V^{\pi}(\psi(s_t, z; \theta_\psi)), \quad \forall s_t \in \mathcal{S}.$$

Algorithm 2 shows the pseudo-code for policy training (full version shown in Appendix D).

Algorithm 1 EMERALD Abstract Representation Model Training

Require: ARM (transition encoder, reward predictor, abstract state encoder, task-encoder), task distribution $p(\mathcal{T})$, learning rate α_θ, batch size B, epochs per task n_{epochs}, initialize replay buffers $\mathcal{D}_i$ for every task $\mathcal{T}_i$, context horizon H

Offline Data Collection: For each task $\mathcal{T}_i$, collect a dataset $\mathcal{D}_i^{\text{offline}} = \{(s_t, a_t, s_{t+1}, r_t)\}_{t=1}^{m_i}$ of offline transitions following policy π.
while not converged **do** ▷ ARM training loop
  for each task $\mathcal{T}_i$ **do**
    Reset *context window*: $h^i \leftarrow \emptyset$
    for epoch $= 1$ to n_{epochs} **do**
      for each batch $\{(s_t, a_t, s_{t+1}, r_t)\} \sim \mathcal{D}_i^{\text{offline}}$ **do**
        Sample task embedding $z \sim \phi(\cdot \mid h^i; \theta_\phi)$
        Encode states: $x_t = \psi(s_t, z; \theta_\psi) \quad x_{t+1}^{\text{true}} = \psi(s_{t+1}, z; \theta_\psi)$
        Predict transition and reward: $\hat{x}_{t+1} = x_t + \tau(x_t, a_t; \theta_\tau)$
        Predict reward: $\hat{r}_t = \rho(x_t, a_t; \theta_\rho)$
        Compute loss: $\mathcal{L}_\theta = \mathcal{L}_{\text{transition}} + \mathcal{L}_{\text{reward}} + \beta\, \mathcal{L}_{\text{reg}}$
        Update ψ parameters: $\theta_\psi \leftarrow \theta_\psi - \alpha_{\theta_\psi} \frac{1}{B} \sum_{i=1}^{B} \nabla_{\theta_\psi} \mathcal{L}_i$ ▷ Parameter updates
        Update ϕ parameters: $\theta_\phi \leftarrow \theta_\phi - \alpha_{\theta_\phi} \frac{1}{B} \sum_{i=1}^{B} \nabla_{\theta_\phi} \mathcal{L}_i$
        Update ρ parameters: $\theta_\rho \leftarrow \theta_\rho - \alpha_{\theta_\rho} \frac{1}{B} \sum_{i=1}^{B} \nabla_{\theta_\rho} \mathcal{L}_i$
        Add to context window: $c^i \leftarrow c^i \cup \{(s_t, a_t, r_t, s_{t+1})\}$ until $|h^i| \leq H$
      end for
    end for
  end for
end while
return Trained ARM.

5 Experiments

Our evaluation consists of four complementary experimental sets, each designed to test a distinct aspect of EMERALD. First, we compare EMERALD with strong baselines on *out-of-distribution* tasks following the evaluation protocol of Lee et al. (2020). Second, we compare EMERALD to another set of baselines on the more recent Meta-World ML1 and ML10 suites Yu et al. (2020). Third, we evaluate the effect of training on multiple environments by measuring performance on various tasks. Lastly, we perform two ablation studies: (i) we examine how the quality of the ARM affects policy learning, and (ii) we visualize the learned task-aware latent space to assess how effectively the policy exploits it.

5.1 Comparative benchmarks

Setting 1: We largely follow the experimental protocol of Lee et al. (2020), using variants of the continuous-control CartPole task and several MuJoCo benchmarks (Todorov et al., 2012). The learning algorithms are trained on a set of configurations and evaluated on previously unseen ones. For CartPole, for example, we train on pole lengths 1.0, 1.5, and 2.0 and test the out-of-distribution performance on poles of length 0.5 and 2.5. HalfCheetah (Volume) is a HalfCheetah setting where we shrink or expand the Cheetah's volume, depending on the exact treatment regimen. The three evaluation domains are illustrated in Figure 3 (and

Algorithm 2 EMERALD Policy Learning

Require: Trained ARM, tasks $\mathcal{T}_i \sim p(\mathcal{T})$, learning rates α_π, α_V, batch size B, rollout length n_r, updates n_u, horizon H; parameters θ_π, θ_V; buffers $\mathcal{D}_i \leftarrow \emptyset$

while not converged **do**
  for all $\mathcal{T}_i$ **do** ▷ Rollout
    $\mathcal{B} \leftarrow \emptyset$, $h^i \leftarrow \emptyset$
    for $t = 1 \dots n_r$ **do**
      $z \sim \phi(\cdot \mid h^i; \theta_\phi)$; $x_t = \psi(s_t, z; \theta_\psi)$, $a_t \sim \pi_{\theta_\pi}(\cdot \mid x_t)$
      env step $\rightarrow s_{t+1}, r_t$, $x_{t+1} = \psi(s_{t+1}, z; \theta_\psi)$
      add (x_t, a_t, r_t, x_{t+1}) to $\mathcal{B}$ and h^i (truncate to H)
    end for
    $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \mathcal{B}$
  end for
  for $u = 1 \dots n_u$ **do** ▷ Update
    for all $\mathcal{T}_i$ **do**
      minibatch $b^i \sim \mathcal{D}_i$; compute L^i_π, L^i_V
    end for
    $\theta_\pi \leftarrow \theta_\pi - \alpha_\pi \nabla_{\theta_\pi} \sum_i L^i_\pi$; $\theta_V \leftarrow \theta_V - \alpha_V \nabla_{\theta_V} \sum_i L^i_V$
  end for
end while
return π_{θ_π}, V_{θ_V}

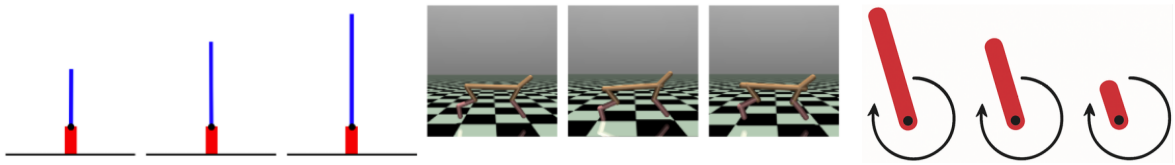


Figure 3: Illustration of the Experimental Setup. **Left:** Cartpole with different sizes. **Middle:** Three HalfCheetah with different volumes. **Right:** Pendulum with different lengths.

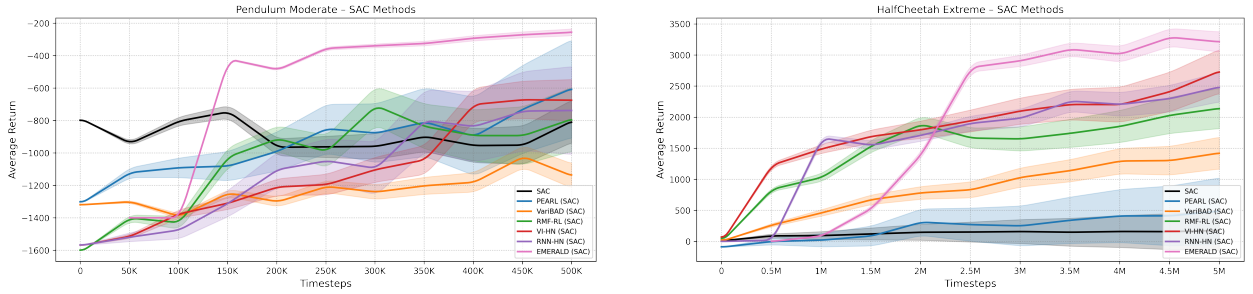
details are provided in Appendix L). EMERALD is paired with Soft Actor-Critic (SAC) (Haarnoja et al., 2018) (EMERALD SAC) and Proximal Policy Optimization (PPO) (Schulman et al., 2017) (EMERALD PPO). The ARM is pre-trained offline with 5×10k CartPole transitions, $11\times \sim 4.5$k Pendulum transitions, and 5×100k HalfCheetah transitions. In order not to give our model an unfair advantage, we collected the offline transitions under a random policy (ablation study in Appendix H shows the effect of using an expert policy).

Following Lee et al. (2020), we evaluate all methods on unseen environments under two regimes (see Appendix J for complete specifications):

1. **Moderate**: Unseen test tasks differ only slightly from the training distribution.

2. **Extreme**: Unseen test tasks deviate substantially from training task.

Setting 2: In the second setting, we compare EMERALD to the same set of baselines on the Meta-World ML10 suite Yu et al. (2020). Meta-World is a benchmark environment for evaluating multi-task and meta-reinforcement learning algorithms, consisting of a diverse collection of robotic manipulation tasks built on the MuJoCo simulator. ML10 includes ten distinct training tasks designed to assess generalization across five test task distributions.



(a) Pendulum

(b) HalfCheetah (Volume)

Figure 4: Learning Curves for two out-of-distribution Experiments from the Moderate Regime (rolling averages over 8 independent runs and adjusted for ARM budget allocation)

Setting 3: In this setting, we compare the performance of EMERALD and VariBAD on the HalfCheetah-Direction task, a standard benchmark in meta-reinforcement learning. Unlike Settings 1 and 2, which test out-of-distribution generalization, this experiment evaluates in-distribution adaptation — specifically, how well the agent adjusts to moving in opposite directions (forward versus backward).

We compare EMERALD PPO and EMERALD SAC with ten baselines:

1. **PPO** (Schulman et al., 2017): Standard Proximal policy optimization (PPO).
2. **SAC** (Haarnoja et al., 2018): Standard Soft Actor Critic (SAC).
3. **Stacked PPO** (Lee et al., 2020): PPO variant that feeds a fixed window of past transitions to the policy.
4. **PEARL** (Rakelly et al., 2019): Infers a context variable by maximizing expected return.
5. **PPO-EP** (Zhou et al., 2019): PPO version that concatenates an embedding from early interactions to the state.
6. **CaDM** (Lee et al., 2020): Augments a forward–backward dynamics model with a context encoder.
7. **VariBAD** (Zintgraf et al., 2021): Uses an RNN to produce a belief state that is concatenated with the environment state and passed to both the policy and value functions.
8. **RMF-RL** (Ni et al., 2021): Recurrent meta-RL model-free approach with optimized architecture and hyperparameters.
9. **RNN-HN** (Beck et al., 2023): Recurrent hypernetwork approach for meta-RL.
10. **VI-HN** (Beck et al., 2023): Variational hypernetwork approach for meta-RL.

To identify the effect of the learning algorithm (PPO or SAC), we pair the most important competitors to a learning algorithm.[1][2] Moreover, to ensure fairness, the ARM is pretrained on offline data and the corresponding transitions are deducted from each method's overall interaction budget, which guarantees that all agents experience the same total number of transitions (see Appendix M for the complete allocation budget). We further use the same hyperparmeter tuning process for all baselines (Appendix N).

[1]N.B. For the hypernetwork-based approaches (VI-HN and RNN-HN) as well as VariBAD, we could not find any official SAC implementations. In prior work these methods are typically paired with on-policy algorithms such as PPO, and empirical results with off-policy algorithms like SAC are rarely reported. For completeness of comparison, we implemented SAC variants of these baselines ourselves using the existing PPO-based source code.

[2]Appendix O contains an analysis whereby we treat the learning algorithm as a hyperparameter.

(a) PPO-based methods

	CartPole (Lengths)		Pendulum (Lengths)		HalfCheetah (Volume)	
	Moderate	Extreme	Moderate	Extreme	Moderate	Extreme
PPO	**198.2** ± 0.9	187.8 ± 4.7	-1113.2 ± 69.1	-1356.8 ± 48.0	807.7 ± 553.6	574.0 ± 645.6
Stacked PPO	197.8 ± 1.3	189.2 ± 6.1	-475.7 ± 228.1	-488.2 ± 178.2	361.1 ± 141.7	5.7 ± 208.1
PPO EP	196.3 ± 4.0	184.5 ± 9.7	-374.3 ± 24.6	**-256.7** ± 26.4	895.3 ± 445.1	674.2 ± 686.8
CaDM (PPO)	197.9 ± 3.0	193.0 ± 3.5	**-279.8** ± 42.1	-426.4 ± 227.0	1224.2 ± 630.0	1021.1 ± 676.6
VariBAD (PPO)	**199.1** ± 2.8	192.0 ± 3.2	**-318.5** ± 56.6	-643.1 ± 311.4	**2964.0** ± 179.5	1618.3 ± 251.8
RMF-RL (PPO)	198.3 ± 0.9	191.9 ± 2.4	-505.6 ± 42.5	-899.3 ± 185.3	2303.5 ± 146.3	1037.0 ± 268.1
VI-HN (PPO)	197.7 ± 2.7	194.5 ± 5.3	-421.1 ± 56.8	-531.3 ± 121.3	1882.1 ± 91.5	1167.9 ± 185.2
RNN-HN (PPO)	198.7 ± 2.1	193.7 ± 4.1	-444.9 ± 51.0	-497.8 ± 117.2	1982.9 ± 88.2	984.2 ± 242.5
EMERALD (PPO)	**199.1** ± 1.1	**199.3** ± 3.5	**-312.4** ± 19.0	-313.7 ± 101.5	**2942.5** ± 213.5	**1401.4** ± 421.3

(b) SAC-based methods

	CartPole (Lengths)		Pendulum (Lengths)		HalfCheetah (Volume)	
	Moderate	Extreme	Moderate	Extreme	Moderate	Extreme
SAC	**199.2** ± 0.3	**199.4** ± 0.1	-402.0 ± 131.3	-1274.2 ± 133.7	1998.3 ± 142.1	177.4 ± 321.3
PEARL (SAC)	198.0 ± 1.4	187.5 ± 10.9	-645.3 ± 320.7	-1136.6 ± 251.0	642.1 ± 488.3	462.1 ± 534.5
VariBAD (SAC)	194.7 ± 4.5	189.6 ± 1.7	-1001.3 ± 75.1	-1193.8 ± 94.1	1755.6 ± 194.2	1424.9 ± 258.3
RMF-RL (SAC)	198.4 ± 2.1	197.2 ± 3.1	-769.3 ± 198.1	-1058.5 ± 238.1	4419.1 ± 219.3	2142.2 ± 323.6
VI-HN (SAC)	199.2 ± 3.6	196.2 ± 6.4	-657.4 ± 128.4	-1098.6 ± 202.3	4365.4 ± 307.9	**2744.3** ± 439.4
RNN-HN (SAC)	196.3 ± 2.3	194.7 ± 3.0	-738.4 ± 271.9	-991.3 ± 298.1	2837.7 ± 218.0	2539.3 ± 260.8
EMERALD (SAC)	194.6 ± 2.7	**196.2** ± 4.9	**-198.3** ± 23.3	**-749.4** ± 340.9	**5101.1** ± 151.8	**3271.8** ± 180.2

Table 1: Performance comparison means over 8 seeds (Mean Return ± Std.) for CartPole, Pendulum, and HalfCheetah (Volume). The best statistically significant (Welch's t-test $p < 0.01$) results in each group are boldfaced and quantify the rolling average over the last 100 timesteps.

5.2 Results Benchmark Experiments

Results for Setting 1: Table 1b shows that EMERALD (SAC) surpasses most SAC-based baselines; the corresponding learning curves are displayed in Figures 4a-4b. In the PPO-based setting, we achieve performance on par with VariBAD (PPO). On the Pendulum task, EMERALD (PPO) does not outperform PPO-EP in the extreme regime, suggesting a potential limitation when the environment exhibits strong periodic dynamics. However, the difference between EMERALD (PPO) and PPO-EP is not large. Notably, both EMERALD (SAC) and EMERALD (PPO) perform well overall, underscoring the benefit of modularity: one can boost performance by combining the ARM with a different learning algorithm. Interestingly, the performance of the non-meta RL learners appears to be predictive of performance with EMERALD: for instance, SAC and EMERALD (SAC) both outperform other methods on the HalfCheetah tasks. Furthermore, while VariBAD does achieve higher performance when paired with PPO, in the SAC-based setting EMERALD (SAC) achieves much higher performance, suggesting that our approach is more naturally paired with SAC than VariBAD.

Results for Setting 2: Across both Meta-World ML10 and ML1, EMERALD variants generally achieve strong and stable performance compared to existing meta-reinforcement learning baselines. On ML10, EMERALD (SAC) attains the highest average performance in both training and testing settings, indicating effective adaptation across multiple tasks. EMERALD (PPO) performs slightly below its SAC counterpart but still matches or exceeds other baselines such as PEARL and RMF-RL on most metrics. Overall, these results suggest that EMERALD's framework contributes to improved consistency and generalization, with

(a) PPO-based methods

Method	HalfCheetah (Direction)
PPO	-219.7 ± 75.2
VariBAD (PPO)	**2112.5** ± 193.6
EMERALD (PPO)	**2259.9** ± 317.1

(b) SAC-based methods

Method	HalfCheetah (Direction)
SAC	-173.9 ± 58.2
VariBAD (SAC)	2806.3 ± 118.7
EMERALD (SAC)	**3343.2** ± 291.1

Table 2: Performance over 8 seeds (Mean Return ± Std.) with random seeds.

SAC providing a performance edge in both efficiency and asymptotic returns. Interestingly, when considering the other baselines, we observe that performance drops significantly when using SAC. For example, VI-HN (PPO) reaches a training score of 41.8%, whereas its SAC counterpart achieves barely half of that. We observe a similar trend on the SAC-based versions on ML1, where for the hypernetworks (RNN-HN (SAC) and VI-HN (SAC)) as well as VariBAD, underperform compared to their PPO-based counterparts. We believe this may be due to the fact that architectures rely on an on-policy, trajectory-consistent data distribution, while SAC's off-policy replay buffer breaks this temporal coherence and destabilizes both latent adaptation and hypernetwork weight generation (see Appendix N for a more extensive discussion). EMERALD, on the other hand, shows a relatively stable performance, regardless of the chosen policy.

(a) PPO-based methods

Method	Avg (Train)	Avg (Test)
RMF-RL (PPO)	16.7 ± 1.0	13.8 ± 0.4
VariBAD (PPO)	**52.4** ± 2.9	**16.1** ± 1.2
VI-HN (PPO)	44.3 ± 1.9	14.2 ± 1.1
RNN-HN (PPO)	14.7 ± 0.8	12.3 ± 0.3
EMERALD (PPO)	46.7 ± 1.9	15.3 ± 0.9

(b) SAC-based methods

Method	Avg (Train)	Avg (Test)
RMF-RL (SAC)	48.5 ± 1.4	10.7 ± 0.9
VariBAD (SAC)	16.6 ± 0.7	13.1 ± 1.1
VI-HN (SAC)	20.8 ± 0.4	14.0 ± 0.6
RNN-HN (SAC)	0.0 ± 0.0	0.0 ± 0.0
EMERALD (SAC)	**53.9** ± 0.6	**17.6** ± 1.0

Table 3: Performance comparison on Meta-World ML10. Results are reported as mean percentages ± standard error over 8 seeds.

Results for Setting 3: The results are reported in Table 2 and show that the PPO version of our approach performs on par with VariBAD (PPO). The improvement over regular PPO further suggests that training *with* augmented abstract states provides an additional boost in performance in this experimental setting. Similar to setting 1, we observe that EMERALD SAC outperforms VariBAD (SAC).

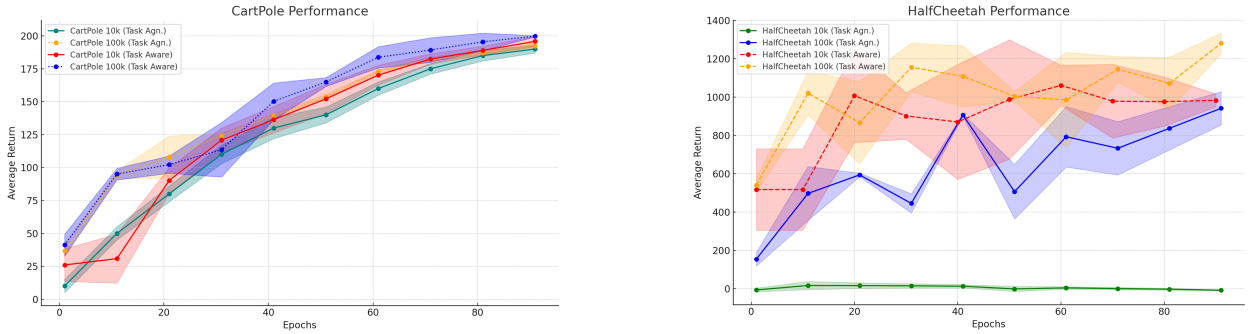


Figure 5: Average Policy Performance (8 seeds) with stdev. on CartPole and HalfCheetah (Volume) for several Training Iterations, starting from a model trained with 1 epoch up to 100 epochs. Dashed lines are task-aware iterations, solid lines display task-agnostic versions (no context encoding).

(a) PPO-based methods

Method	door-open	basketball	window-open	pick-place	button-topdown
RMF-RL (PPO)	0.0 ± 0.0	0.0 ± 0.0	34 ± 1.0	0.0 ± 0.0	0.0 ± 0.0
VariBAD (PPO)	**91** ± 2.4	0.0 ± 0.0	**100** ± 0.0	**28** ± 2.0	98 ± 1.0
VI-HN (PPO)	0.0 ± 0.0	0.0 ± 0.0	**100** ± 0.0	33 ± 0.3	0.0 ± 0.0
RNN-HN (PPO)	0.0 ± 0.0	0.0 ± 0.0	95 ± 3.5	0.0 ± 0.0	0.0 ± 0.0
EMERALD (PPO)	**92** ± 0.6	**4.0** ± 0.9	65 ± 3.8	0.0 ± 0.0	**99** ± 1.0

(b) SAC-based methods

Method	door-open	basketball	window-open	pick-place	button-topdown
RMF-RL (SAC)	99 ± 0.3	**40** ± 0.6	**96** ± 0.6	0.0 ± 0.0	98 ± 0.7
VariBAD (SAC)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
VI-HN (SAC)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
RNN-HN (SAC)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
EMERALD (SAC)	**100** ± 0.0	2.0 ± 0.5	64 ± 0.4	0.0 ± 0.0	**99** ± 0.6

Table 4: Performance comparison on Meta-World ML1 tasks. Results are reported as mean percentages ± standard error over 8 seeds

Environment	Config	# Training Tasks	Samples per Task	Mean Return ± Std
CartPole	1	1	200k	91.4 ± 37.5
CartPole	2	5	40k	187.0 ± 19.8
HalfCheetah	1	1	3M	749.2 ± 209.3
HalfCheetah	2	5	600k	1218.0 ± 338.7

Table 5: EMERALD In-Distribution Performance on Training Tasks (Mean Return ± Std. over 5 seeds). Mean difference statistically significant (Welch's t-test $p < 0.01$)

5.3 Ablation Studies

Effect of Environment Diversity on Performance We ask whether training on a *greater diversity* of tasks is more beneficial than training with *more data* drawn from a single environment. This is an in-distribution setting, as train and test tasks are the same. We allocate a total budget of 3M transitions for HalfCheetah (Volume) and 200k for CartPole. In **Configuration 1** the entire budget is spent on one task, whereas in **Configuration 2** the budget is split evenly across multiple tasks. Table 5 summarizes the outcomes. Configuration 2 yields higher overall performance, indicating that exposure to a wider set of environments outweighs simply scaling up data for a single task, demonstrating that increased task-diversity offers greater sample efficiency.

Model Effect on Policy Performance We study how performance varies with (i) the number of training epochs, (ii) the amount of offline data, and (iii) the presence of task-aware encodings. Figure 5 reports the results (each marker is the mean of 8 seeds). In the CartPole setting with 10k transitions, we observe a slightly higher performance of the task-aware variant. Similarly, for CartPole with 100k transitions, the task-aware variant outperforms the task-agnostic one. In the HalfCheetah (Volume) setting, the same pattern emerges but is more pronounced: both the task-aware 100k and 10k-transition regimes converge earlier. Moreover, the performance of the task-agnostic (10k) variant is effectively zero, which suggests that removing task encodings (i.e., using task-agnostic abstract states) sharply degrades performance, particularly on HalfCheetah, demonstrating the effectiveness of using a state-task encoder.

Latent Trajectory Analysis Figure 6 visualizes policy behavior in both training and test environments. For each environment, we record the first 50 transitions and project them into a three-dimensional principal

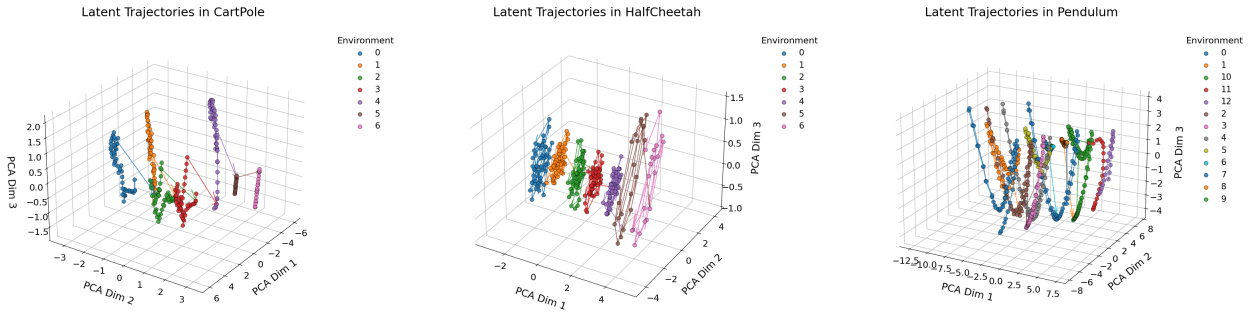


Figure 6: Visualization of the Latent Space (PCA mapping) of the EMERALD ARM with Training and Test Environments of the first 50 transitions. **Left:** Cartpole. **Middle:** HalfCheetah (Volume). **Right:** Pendulum. In CartPole and HalfCheetah (Volume) the out-of-distribution trajectories are 5 and 6, while in Pendulum 11 and 12.

component space. The resulting training and test trajectories exhibit similar shapes but occupy distinct regions, suggesting that EMERALD learns behaviors that transfer across tasks while remaining sensitive to task identity.

6 Discussion & Conclusion

We introduced EMERALD, a meta-reinforcement-learning method that leverages augmented, context-aware abstract states. Across benchmarks, our approach either outperforms other model-free context-based baselines or performs on par with state-of-the art approaches. In particular, we identified the combination of EMERALD with SAC as a natural pairing, performing generally better than competitors. On the other hand, when using PPO the performance of EMERALD compared to other PPO-based baselines is less pronounced. Still, the fact that EMERALD easily pairs with both on-policy and off-policy methods, shows its versatility. Further, relative to its non-meta-RL counterparts, we show that task-aware representation models yield substantial gains.

Beyond improved performance, we provided justification for the representational and architectural advantages of our approach: by disentangling task-relevant features early in the pipeline, we simplify training, improve robustness, and gain compatibility with a broad class of RL algorithms as our procedure yields a simple encoder that produces task-aware abstractions. Despite these strengths, EMERALD has limitations. First, the optimal structure of the latent space (e.g., dimensionality) can be environment-dependent and difficult to determine *a priori*. Second, as the number of training tasks grows, the latent space may need to expand to capture task diversity, potentially increasing computational cost. In such cases, hybrid approaches that decouple task inference from state abstraction may offer better scalability. Identifying where this trade-off becomes critical is an important direction for future work.

A natural extension of EMERALD is to integrate it into a *model-based* framework. Whereas the present study focuses on using a *pretrained* latent model within a modular, model-free pipeline, one could instead exploit the latent dynamics directly to train a policy, thereby opening the door to direct comparisons with established model-based meta-RL methods.

References

Jacob Beck, Risto Vuorio, Zheng Xiong, and Shimon Whiteson. Recurrent hypernetworks are surprisingly strong in meta-rl. *Advances in Neural Information Processing Systems*, 36:62121–62138, 2023.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, Shimon Whiteson, et al. A tutorial on meta-reinforcement learning. *Foundations and Trends® in Machine Learning*, 18(2–3): 224–384, 2025.

Nicolò Botteghi, Mannes Poel, and Christoph Brune. Unsupervised representation learning in deep reinforcement learning: A review. *arXiv preprint arXiv:2208.14226*, 2022.

Mustapha Bounoua, Giulio Franzese, and Pietro Michiardi. Learning to match unpaired data with minimum entropy coupling. *arXiv preprint arXiv:2503.08501*, 2025.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1999.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Vincent François-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. Combined reinforcement learning via abstract representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3582–3589, 2019.

Robert M. Gray. *Entropy and Information Theory*. Springer Science & Business Media, 2011.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G. M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

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.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pp. 5757–5766. PMLR, 2020.

Bonnie Li, Vincent François-Lavet, Thang Doan, and Joelle Pineau. Domain adversarial reinforcement learning. *arXiv preprint arXiv:2102.07097*, 2021.

David J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. *arXiv preprint arXiv:2110.05038*, 2021.

Tianwei Ni, Benjamin Eysenbach, Erfan Seyedsalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive rl. *arXiv preprint arXiv:2401.08898*, 2024.

Gaurav Pandey and Ambedkar Dukkipati. Variational methods for conditional multimodal deep learning. In *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 308–315. IEEE, 2017.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 5331–5340. PMLR, 2019.

Zohar Rimon, Tom Jurgenson, Orr Krupnik, Gilad Adler, and Aviv Tamar. Mamba: an effective world model approach for meta-reinforcement learning. *arXiv preprint arXiv:2403.09859*, 2024.

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

Ohad Shamir, Sivan Sabato, and Naftali Tishby. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29–30):2696–2711, 2010.

Rolf AN Starre, Marco Loog, and Frans A Oliehoek. Model-based reinforcement learning with state abstraction: A survey. In *Benelux Conference on Artificial Intelligence*, pp. 133–148. Springer, 2022.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033. IEEE, 2012.

Geoffrey Van Driessel and Vincent François-Lavet. Component transfer learning for deep rl based on abstract representations. *arXiv preprint arXiv:2111.11525*, 2021.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pp. 1094–1100. PMLR, 2020.

Amy Zhang, Harsh Satija, and Joelle Pineau. Decoupling dynamics and reward for transfer learning (2018). *arXiv preprint arXiv:1804.10689*, 2018.

Wenxuan Zhou, Lerrel Pinto, and Abhinav Gupta. Environment probing interaction policies. In *ICLR*, 2019.

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.