Offline Reinforcement Learning from Datasets with Structured Non-Stationarity

Johannes Ackermann
ackermann@ms.k.u-tokyo.ac.jp
The University of Tokyo, RIKEN AIP

Takayuki Osa
The University of Tokyo, RIKEN AIP

Masashi Sugiyama
RIKEN AIP, The University of Tokyo

Abstract

Current Reinforcement Learning (RL) is often limited by the large amount of data needed to learn a successful policy. Offline RL aims to solve this issue by using transitions collected by a different behavior policy. We address a novel Offline RL problem setting in which, while collecting the dataset, the transition and reward functions gradually change between episodes but stay constant within each episode. We propose a method based on Contrastive Predictive Coding that identifies this non-stationarity in the offline dataset, accounts for it when training a policy, and predicts it during evaluation. We analyze our proposed method and show that it performs well in simple continuous control tasks and challenging, high-dimensional locomotion tasks. We show that our method often achieves the oracle performance and performs better than baselines.

1 Introduction

A main challenge of Reinforcement Learning (RL) is the large amount of interactions required to learn a proficient policy. One recently popular way to tackle this challenge is to use Offline Reinforcement Learning (Levine et al., 2020). In Offline RL we aim to learn a policy from a given dataset of previous transitions generated by a different behavior policy, without needing to interact with the environment further. This avoids the cost and potential risks of online data collection, allowing us to collect large datasets. Consider, for example, a policy being trained to improve the controller of a deployed pick and place robot. Over shorter time frames, we would not expect wear and tear to have a large effect on the robot: We can expect our environment to be stationary, i.e., the reward and transition functions should be the same over the course of data collection. However, if we collect the dataset over a longer time frame wear and tear does occur, leading to nonstationarity which causes an important challenge to real world RL (Dulac-Arnold et al., 2021). With recent works such as Kalashnikov et al. (2021) training on multiple robots over time-frames of up to 16 months, this challenge is becoming increasingly relevant.

As episodes tend to be short compared to the lifespan of a robot, we can assume that the change in transition and reward functions during each episode is small. We thus tackle this setting by making the structural assumption of a slowly evolving non-stationarity, that remains fixed during each episode and changes between them, allowing us to formulate the setting as multiple rollouts of a Dynamic-Parameter MDP (DP-MDP) (Xie et al., 2021). A DP-MDP is a Hidden-Parameter MDP (HiP-MDP) (Doshi-Velez & Konidaris, 2013) in which the hidden-parameter (HiP) depends on the previous HiPs. One way to address our problem setting is to use Bayes-Adaptive RL methods, such as BOReL (Dorfman et al., 2021) or ContraBAR (Choshen & Tamar, 2023). These methods learn a policy that optimally identifies and exploits the HiP during the same episode. Another way to approach our problem setting is to derive an offline variant of a lifelong-learning methods like

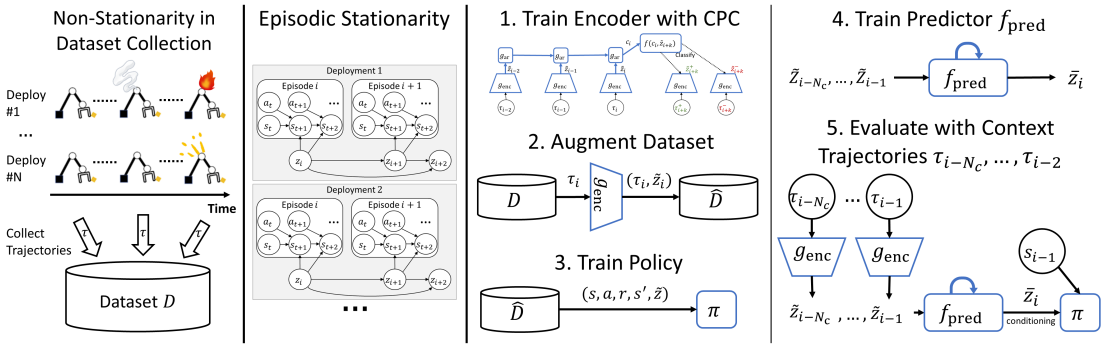


Figure 1: We address an Offline RL setting in which the dataset is generated from multiple deployments with evolving non-stationarity. We make the structural assumption of the reward and transition functions depending on a hidden-parameter z that is constant during each episode but evolves between episodes. Following this assumption, we develop a method based on Contrastive Predictive Coding that infers the hidden parameter from the deployments in our dataset. We then train a predictor and policy to use during evaluation with access to context trajectories.

Lifelong Actor Critic (LILAC) (Xie et al., 2021) which trains a Dynamic Variational Autoencoder (VAE) (Chung et al.) to learn a model of the reward and transition functions. However, as we will discuss in detail later, these methods contain techniques (reward relabeling, policy replaying, and hard negative mining) that are not applicable in our setting and struggle in high-dimensional tasks with changing transition functions.

We therefore propose a method that avoids the need for these additional techniques and performs well in high-dimensional tasks by using Contrastive Predicitive Coding (CPC) (Oord et al., 2019). We show that our method is able to learn a meaningful representation of the HiP, identify it in the dataset, predict it during evaluation and use it to learn an effective policy. To summarize our contributions, we 1) propose a new offline RL problem setting of learning from a dataset including a structured nonstationarity, 2) address this setting by deriving a method based on CPC that infers the non-stationarity in the dataset and predicts it during evaluation, 3) show that our method outperforms baselines in both simple and high-dimensional continuous control tasks and publish our code and datasets for the community to use.

2 Background

In this section, we will briefly describe the necessary background on RL, CPC and HiP-MDPs.

2.1 Reinforcement Learning

In RL, we are given an MDP $M = (S, A, R, p, p_0)$ with, in this work, continuous state space S, continuous action space A, deterministic reward function $R(s, a)$, transition probability density $p(s'|s, a)$ and initial state density $p_0(s)$ (Sutton & Barto, 2018). We then aim to learn a policy with conditional probability density $\pi(a|s)$ of choosing action a in state s, that maximizes the expected return $J = \mathbb{E}\left[\sum_{t=0}^{H} r_t\right]$, where H is the duration of an episode. We refer to a rollout of this MDP as trajectory $\tau = (s_0, a_0, r_0, \dots, s_H, a_H, r_H)$.One important quantity of interest is the value function $Q(s, a) = \mathbb{E}[\sum_t \gamma^{t-1} r_t | s_0 = s, a_0 = a]$. A common way to estimate it is to use the recurrent formula $Q(s, a) = r(s, a) + \gamma \mathbb{E}[Q(s', a')]$, where s', a' are the subsequent state and action. In the continuous control setting, Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018) is one popular method to learn the policy. As an actor-critic method it consists of a critic-network Q_θ that estimates the action-value function and a deterministic policy $\mu_\phi : S \to A$. Both are represented as Multi Layer Preceptrons (MLPs) with parameters θ and ϕ respectively and the policy is updated according to the deterministic policy gradient $\nabla_\phi J(\phi) = \mathbb{E}[\nabla_\phi \mu_\phi(s) \nabla_a Q_\theta(s, a)|_{a=\mu_\phi(s)}]$.

In Offline RL, we train a policy from a dataset of transitions $D = \{(s^{\mathrm{i}}, a^{\mathrm{i}}, r^{\mathrm{i}}, s'i)\}_{\mathrm{i}=1}^{N_{\mathrm{D}}}$ obtained by executing a behavior policy $\beta(a|s)$. The main challenge in Offline RL is the distribution shift between the states visited and actions chosen by the behavior policy β and the learned policy π (Levine et al., 2020). If the shift is large, estimates of the Q-value become inaccurate, leading to the policy choosing actions that result in a poor performance. Therefore, methods such as TD3 do not perform well if applied directly in Offline RL (Fujimoto & Gu, 2021), and instead most methods introduce some way to constrain the learned policy π to remain close to the behavior policy β. TD3+BC (Fujimoto & Gu, 2021) is a successful Offline RL method that achieves this by adding a behavior cloning (BC) penalty $\mathbb{E}_{(s,a)\sim D}[(\pi(s) - a)^2]$ to the policy loss, weighted by a hyperparameter $\lambda > 0$.

2.2 Contrastive Predictive Coding

CPC (Oord et al., 2019) uses contrastive learning to learn a representation c_t for a sequence of observations $o_{1:t}$[1]. Each observation o is first encoded by the same encoder g_{enc}, yielding encodings $x_t = g_{\mathrm{enc}}(o_t)$. These encodings are then processed by an auto-regressive model g_{ar}, such as a Gated-Recurrent Unit (GRU) (Cho et al., 2014), which gives us a representation of the sequence $c_t = g_{\mathrm{ar}}(x_{1:t})$. We then take the future observation o^+_{t+k} from the same sequence as a positive sample and sample a set of N^- observations $\{o^{-,j}_{t+k}\}_{j=1}^{N^-}$ from different sequences as negative samples. A classifier $f_k(c_t, x_{t+k})$ is then trained to classify which embeddings are from the positive sample $x^+_{t+k} = g_{\mathrm{enc}}(o^+_{t+k})$ or negative samples $x^{-,j}_{t+k} = g_{\mathrm{enc}}(o^{-,j}_{t+k})$. The encoder g_{enc}, autoregressive model g_{ar} and classifier f are trained jointly to optimize the InfoNCE loss:

$$\mathcal{L}_{\texttt{InfoNCE}} = -\mathbb{E}\left[\log \frac{\exp f_k(c_t, x^+_{t+k})}{\exp f_k(c_t, x^+_{t+k}) + \sum_{j=1}^{N^-} \exp f_k(c_t, x^{-,j}_{t+k})}\right], \tag{1}$$

which is minimized when the classifier is proportional to the density ratio of a sample being from the conditional density $p(o_{k+t}|c_t)$ instead of the proposal density $p(o_{k+t})$, i.e., $f(o_{t+k}, c_t) \propto \frac{p(o_{k+t}|c_t)}{p(o_{k+t})}$, thereby maximizing the mutual information $I(c_t; o_{t+k})$ (Oord et al., 2019).

2.3 Partially-Observable MDPs

In the classical MDP the reward function R and transition function P are stationary during training, i.e., they do not change. While this assumption is easily satisfied in constructed examples, in realistic settings external influences and unobservable factors can make it difficult or impossible to choose a state formulation that permits a stationary transition and reward functions. Partially-Observable MDPs (POMDPs) (Åström, 1965) extend the MDP formulation by assuming that while the transition and reward functions seem non-stationary from the given observation s, they are stationary given the unobserved state $\hat{s}$. The observation is given by an observation function $h : \hat{S} \to S$. We can represent any kind of non-stationary transition- or reward-function as a POMDP, however, the generality of the formulation makes efficient training difficult. HiP-MDPs introduced by Doshi-Velez & Konidaris (2013) address this issue by constraining the true state $\hat{s}$ to be the combination of the observation received by the agent with a hidden parameter z, i.e. $\hat{s} = (s, z)$. While in general continuous sets of HiPs can be considered, following related works (Xie et al., 2021; Dorfman et al., 2021) we focus on a discrete set of HiPs Z in our experiments and thus also in the rest of this work. This HiP is sampled at the beginning of each episode from a distribution $z \sim P(z)$ and remains constant throughout the episode. The transition function $P(s'|s, a, z)$ then depends on the hidden parameter z. DP-MDPs (Xie et al., 2021), visualized in Fig. 2 (left), generalize the HiP-MDP by considering a structured evolution of HiPs:

Definition 2.1. Dynamic-Parameter MDP (Xie et al., 2021) A DP-MDP is a is an MDP with the addition of a HiP space Z, transition probability $P_{\mathrm{z}}(z_i|z_{0:i-1})$, and initial probability $P_{\mathrm{z}_0}(z_0)$. The HiP is constant during each episode and follows $P_{\mathrm{z}}(z_i|z_{0:i-1})$ between episodes. The transition density $p(s'|s, a, z)$ and reward function $R(s, a, z)$ depend on the HiP.

[1] We sometimes use the notation $x_{a:b} := (x_a, x_{a+1}, \ldots, x_b)$ for brevity.

2.4 Bayes-Adaptive RL

The HiP-MDP setting can be addressed by Bayes-Adaptive RL methods, which train a policy that infers and exploits the HiP during an episode. Variational Bayes Adaptive Deep RL (VariBAD) (Zintgraf et al., 2020) achieved this by training a VAE with a Gaussian encoder $[\mu_t, \Sigma_t] = q_\phi(s_{1:t}, a_{1:t}, r_{1:t})$, reward decoder $p_{r,\phi}(r_{t'}|s_{t'}, a_{t'}, \tilde{b}_t)$ and transition decoder $p_{t,\phi}(s'_{t'}|s_{t'}, a_{t'}, \tilde{b}_t)$, for $1 \leq t' << t$, where $\tilde{b}_t \sim \mathcal{N}(\mu_t, \Sigma_t)$. $\mathcal{N}(\mu, \Sigma)$ is a multivariate Gaussian distribution with mean μ and diagonal covariance matrix Σ. $b_t = (\mu, \Sigma)$ is the belief over the current HiP, which the policy is then conditioned on. Dorfman et al. (2021) investigated the application of VariBAD to offline datasets generated by behavior policies $\beta(a|s, z)$, conditioned on the task HiP z, and introduced Bayes Adaptive Offline Reinforcement Learning (BOReL) with two new techniques that enable successful training: *Reward Relabeling* which for each transition (s, a, r, s') in the dataset creates additional transitions $(s, a, R(s, a, z_i), s')$ for all HiPs $z_i \in Z$. *Policy Replaying*, which generates trajectories with HiP z using the behavior policy conditioned on each other HiP $z_i \neq z$. Choshen & Tamar (2023) introduced Contrastive Bayes Adaptive Deep RL (ContraBAR) which instead of a VAE uses CPC to learn a belief over the HiP. Specifically, it encodes transitions using an encoder $z_t = g_{\text{enc}}(s_t, a_{t-1}, r_t)$ and combines them using a Recurrent Neural Network (RNN) to a belief $b_t = g_{\text{ar}}(z_{1:t})$, which is trained by discriminating the next transitions (s^+_{t+k}, r^+_{t+k}) from the same episode against those from a different episode (s^-_{t+k}, r^-_{t+k}). However, as they pointed out, it is possible to discriminate the future transition not by learning a belief over the HiP but by learning a transition model $p(s_{t+k}|s_t)$, leading the training to fail. To prevent this they used *hard negative mining*, either by reward relabeling when only the reward changes or by simulating transitions when the transition function changes, requiring access to a simulator of the environment. As we will see in the next section, *reward relabeling* is not applicable in our setting as we do not consider access to the reward function, *policy replaying* is not applicable as our behavior policy is not conditioned on z and *hard negative mining* is not applicable as we do not have access to a simulator of the environment.

2.5 Problem Formulation

Recall that our motivation is a setting in which data is collected from multiple deployments over an extended time-frame. As episodes tend to be short compared to the lifespan of a deployment, it is reasonable to assume stationarity during the duration of each episode, making the DP-MDP (Xie et al., 2021) a natural fit. We further assume that the data is generated by a behavior policy $\beta(a|s)$, that does not have access to the HiP z, for example a robust controller that performs well but not optimally on all HiPs. We aim to improve on this behavior policy by inferring and using the HiP z.

Problem Setting: We are given a dataset $D = \{d^j\}_{j=1}^N$ consisting of N deployments $d = (\tau_1, \ldots, \tau_i, \ldots, \tau_M)$, each containing M trajectories τ_i. Each deployment d is a rollout of the same DP-MDP $\mathcal{M}$. The deployments are generated by, for each deployment, first sampling a HiP sequence $z_0 \sim P_{\text{z}_0}(z)$, $z_i \sim P_{\text{z}}(z_i|z_{1:i-1})$ and then each trajectory τ_i is generated by sampling $s_0 \sim p_0(s)$ and following behavior policy $\beta(a|s)$, transition density $p(s'|s, a, z_i)$, and reward function $R(s, a, z_i)$. During evaluation we are given a context of N_c previous trajectories $\tau_{i-N_\text{c}:i-1}$ generated by the behavior policy β and our objective is to learn a policy conditioned on the context $\pi(a|s, \tau_{i-N_\text{c}:i-1})$ that maximizes the return over the next episode, i.e., $J = \mathbb{E}_{\pi, P_\text{z}, P, P_0, z_{i-N_\text{c}:i-1}, \tau_{i-N_\text{c}:i-1}} \left[\sum_{t=0}^H R(s_t, a_t, z_i)\right]$.

3 Algorithm

We introduce our proposed method named Contrastive Predictive Non-Stationarity Adaptation (COSPA). As the reward and transition functions depend on the HiP z_i, we first must infer it in the dataset, use it to train a policy and then predict it during evaluation. The offline setting allows us to separate these steps and train an inference model and predictor model first.

Inferring the Hidden Parameter From the generative model of the DP-MDP, shown in Fig. 2, we know that the next episode τ_i with HiP z_i only depends on the HiPs $(z_{i-1}, z_{i-2}, \ldots, z_1)$ of

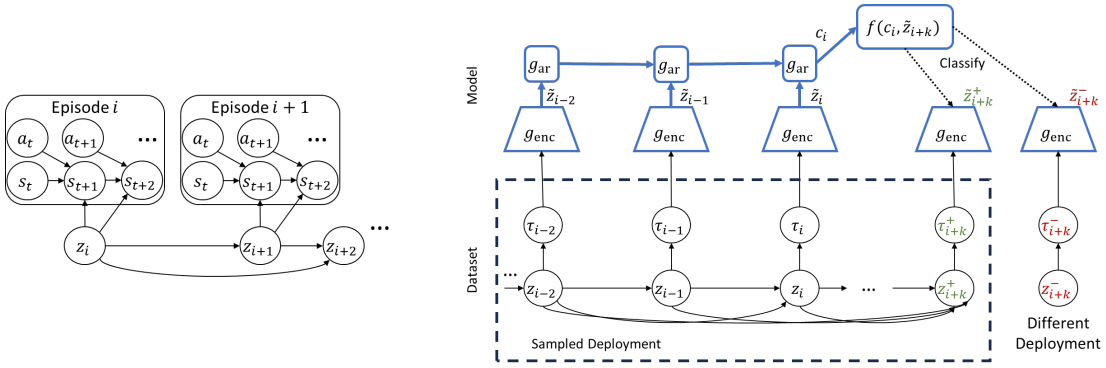


Figure 2: Left: Graphical model of the DP-MDP. Right: Illustration of a deployment sampled from the dataset and our approach to infer the hidden variable. We use Contrastive Predictive Coding to learn a model that can discriminate future trajectories τ_{i+k} based on past trajectories $(\tau_i, \tau_{i-1}, \ldots, \tau_1)$ by learning a representation of the past trajectories $(\tilde{z}_i, \tilde{z}_{i-1}, \ldots, \tilde{z}_1)$.

previous episodes $(\tau_{i-1}, \tau_{i-2}, \ldots, \tau_1)$. One way of learning to infer an approximate HiP $\tilde{z}$ in this setting is to derive an offline variant of LILAC (Xie et al., 2021). LILAC trains a Dynamic VAE with an encoder $\tilde{z}_i = q_\phi(\tau_i)$, dynamic prior $p_\psi(\tilde{z}_i|\tilde{z}_{i-1}, \ldots, \tilde{z}_1)$ and decoder $p_\phi(\tau_i|\tilde{z}_i)$. While training an accurate decoder $p_\phi(\tau_i|\tilde{z}_i)$ is feasible in settings with varying reward functions or in low-dimensional problems, it becomes challenging in high-dimensional settings with small variations in the transition function, as we will see in the experiments in Section 4.1.

Instead of learning a generative model $p_\phi(\tau_i|\tilde{z}_i)$, it is often easier to learn a discriminative model. This makes the application of contrastive learning and in particular CPC a natural choice for our problem setting. As shown in Fig. 2 (right), we treat each trajectory τ as an observation of a time-sequence, encode them separately using an encoder $\tilde{z}_i = g_{\text{enc}}(\tau_i)$ and then summarize the past encodings to a context c_i using an autoregressive encoder $c_i = g_{\text{ar}}(\tilde{z}_i, \tilde{z}_{i-1}, \ldots, \tilde{z}_1)$. Finally, in the InfoNCE loss, a classifier $f(c_t, \tau_{t+k})$ is used to distinguish a future trajectory τ^+_{i+k} of the same deployment from negative trajectory samples $\{\tau^{-,j}_{i+k}\}_{j=1}^{N^-}$ from different deployments. The InfoNCE loss in (1) therefore becomes

$$\mathcal{L}_{\text{repr}} = -\mathbb{E}\left[\log \frac{\exp f_k(\tau^+_{i+k}, c_i)}{\exp f_k(\tau^+_{i+k}, c_i) + \sum_{j=1}^{N^-} \exp f_k(\tau^{-,j}_{i+k}, c_i)}\right]. \tag{2}$$

This structure is shown in Fig. 2 (right). As we learn to discriminate future trajectories τ^+_{i+k}, instead of future transitions (s^+_{t+k}, r^+_{t+k}) as in ContraBAR, the model can not simply learn the transition function $p(s_{t+k}|s_t)$ but has to learn a representation of the HiP z to discriminate τ_{i+k}.

Following the same argument as Oord et al. (2019), we can show that minimizing this loss maximizes the mutual information $I(c_i; \tau_{i+k})$, which we show in Appendix B for completeness. We could thus directly use c_i as an approximation of the hidden parameter z_{i+k}. However, this would have the disadvantage of requiring at least k episodes as context during evaluation, and prevent usage of the first k episodes per deployment during training. We avoid this issue by using the output of the encoder $\tilde{z}_i = g_{\text{enc}}(\tau_i)$ as an approximation of the HiP, obtaining the augmented dataset $\hat{D}$ with $\hat{s} = (s, \tilde{z})$ to train the policy π, and train a separate prediction network to use during evaluation.

Predicting the Next Hidden State During evaluation we only have access to a context of N_c previous trajectories and need to infer the next HiP τ_{i+1} to condition our policy on. We therefore train a predictor RNN f_{pred} to predict the next HiP z_{i+1}. To train it, we sample sequences of inferred latents $(\tilde{z}_{i-N_c}, \ldots, \tilde{z}_i)$ from the dataset $\hat{D}$ and minimize the mean squared error $\mathcal{L}_{\text{pred}} = \mathbb{E}_{(\tilde{z}_{i-N_c}, \ldots, \tilde{z}_i) \sim \hat{D}}[(f_{\text{pred}}(\tilde{z}_{i-N_c}, \ldots, \tilde{z}_{i-1}) - \tilde{z}_i)^2]$. As we will show in Section 4.1, we found that this works well even with relatively simple network structure, consisting of two hidden layers and a GRU.

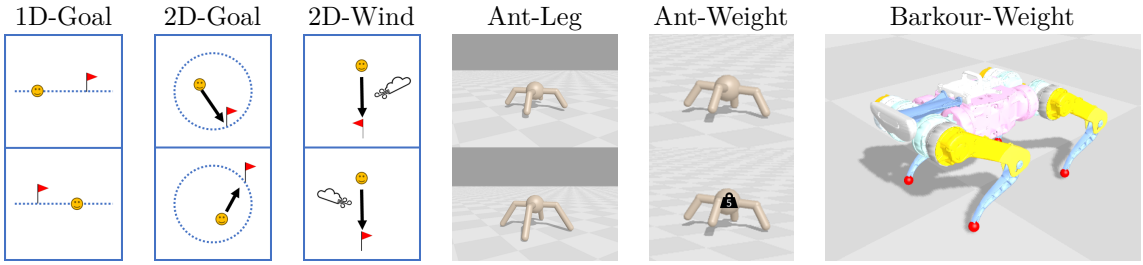


Figure 3: Illustrations of our evaluation environments. From left to right: *1D-Goal*, *2D-Goal*, *2D-Wind*, *Ant-Leg*, *Ant-Weight*, *Barkour-Weight*. In *1D-Goal* and *2D-Goal* the goal location and thus the reward function depends on the HiP z. In the remaining tasks the transition function changes.

Reinforcement Learning Having inferred the HiP and relabeled our offline dataset as $\hat{s} = (s, \tilde{z})$, we now need to train a policy $\pi(a|\hat{s})$. While in principle any Offline RL method may be used, we need to consider how our problem setting differs from the popular D4RL benchmark (Fu et al., 2021), which many popular methods are designed for. More so than in D4RL, our setting requires large deviations from the behavior policy due to the difference in transition and reward functions. This makes methods such as Advantage-Weighted Regression (Peng et al., 2019) or Implicit Q Learning (Kostrikov et al., 2021) that contain strong constraints to in-dataset actions disadvantageous. TD3+BC (Fujimoto & Gu, 2021) uses a deterministic policy $\mu_\phi(s)$ and conservativity is achieved by a BC term $\mathbb{E}_{(s,a)\sim D}\left[(\mu_\phi(s) - a)^2\right]$. In our setting we extend this to $\mathbb{E}_{(s,a,\tilde{z})\sim \hat{D}}\left[(\mu_\phi(s, \tilde{z}) - a)^2\right]$, which can be estimated using samples from the augmented dataset $\hat{D}$. We can adjust the strength of this constraint by varying the hyperparameter λ, allowing more flexibility to learn the policy μ_ϕ, and thus use TD3+BC in our experiments.

4 Implementation and Experiments

We make several design choices in the implementation of our method to enable efficient training. We implement g_{enc} as a two-layer MLP with ReLU activations, g_{ar} as a GRU (Cho et al., 2014) and the classifier f as an MLP with two hidden-layers. One important consideration is how to encode the trajectory τ with encoder g_{enc}. To enable efficient training, we use sampled transitions (s_t, a_t, r_t, s_{t+1}) as input. This works well in tasks with changing transition dynamics, such as changes to the configuration of a robot, or dense reward functions. For sparse reward tasks recurrent encoders can be considered, but we focus on the case with changing transition functions. When augmenting the dataset, we average the output over each trajectory, i.e., each transition of trajectory τ_i is augmented with the average embedding $\frac{1}{H}\sum_{t=1}^{H-1} g_{\text{enc}}(s_t, a_t, r_t, s_{t+1})$ of the trajectory. We further use a low dimensionality $(2, 4, 8)$ for $\tilde{z}$ and normalize it before using it in the policy training. The prediction RNN f_{pred} is implemented as a two-layer MLP followed by a GRU. For TD3+BC, we use similar parameters to the ones proposed by the authors Fujimoto & Gu (2021), but decrease the strength of the BC penalty to account for the larger difference between behavior policy and optimal policy. Finally, we also add layer-normalization (Ba et al., 2016) after each hidden layer of the critic, as suggested by Kumar et al. (2022). We implement our method using JAX and publish our implementation.[2] We use the same RL hyper-parameters per environment for all methods, but optimize the hyper-parameters of our baseline representation methods by extensive grid-search per task on the low-dimensional and Ant tasks. Additional details can be found in Appendix C.

4.1 Evaluation

To validate our approach, we compare its performance with multiple baselines: 1) *Blind*, where we do not add any information to the states in the offline dataset, $\hat{s} = (s)$. 2) *Oracle*, where we augment

[2]see https://github.com/JohannesAck/OfflineRLStructuredNonstationarity

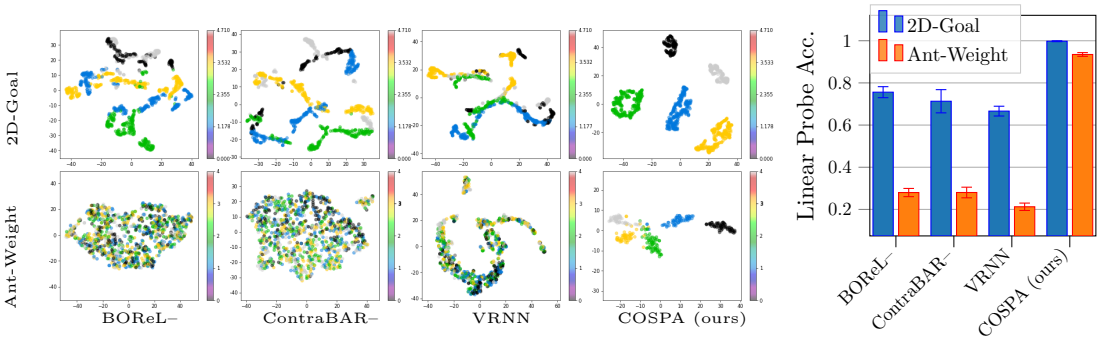


Figure 4: Comparison of the learned representations. The left side shows T-SNE visualizations, the right side shows the mean test accuracy with 95% CIs across 20 trials of linear probes trained to predict the ground-truth HiPs. Each dot is the embedding of a trajectory, the HiP of which is represented by the color. For BOReL and VRNN the mean μ of the posterior is visualized.

the dataset with the ground-truth HiP $\hat{s} = (s, z)$. Note that this value is not accessible in practice. 3) *BOReL*, a method that infers the HiP during the episode using a VAE (Dorfman et al., 2021). As we are not able to use reward relabeling or policy replaying for the complete BOReL, we denote it with *BOReL–*. 4) *ContraBAR–*, ContraBAR (Choshen & Tamar, 2023) addresses the same setting as BOReL, but like us uses a CPC-based architecture. We are not able to use reward relabeling or hard negative mining and therefore denote it *ContraBAR–*. 5) *VRNN*, we also evaluate a Dynamic VAE based method, similar to the LILAC (Xie et al., 2021) which was proposed to address the online DP-MDP setting. Our implementation deviates from LILAC by using a Variational RNN (Chung et al.) as dynamic VAE, as we found it to perform well in preliminary experiments. We also use TD3-BC to train the policy, while LILAC uses SAC.

To generate our dataset, we train a policy using TD3, or PPO for *Barkour*, in the same DP-MDP without access to the HiP. As the HiP changes after each episode, this is similar to utilizing domain randomization (Tobin et al., 2017), resulting in a robust policy that is not specialized to any HiP but close to optimal on the "marginal" MDP, i.e., an MDP where the transition density and reward functions are marginalized over z. We then collect the dataset by generating rollouts with the same exploration noise as during training. Our evaluation tasks are based on related work (Xie et al., 2021; Dorfman et al., 2021) and we begin with three low-dimensional tasks: In *1D-Goal* the agent starts in a random position and navigates to one of two goals, while in *2D-Goal* the agent moves to a goal location on a circle, the exact position of which depends on the HiP z. In *2D-Wind* we instead change the transition function, by adding a disturbance to the location depending on the HiP z. As higher-dimensional tasks we consider two variations of the well known Ant (Schulman et al., 2018) task, simulated using BRAX (Freeman et al., 2021). In *Ant-Weight* we vary the mass of the body of the ant to simulate a varying load, in *Ant-Leg* we change the length of the legs of the ant to simulate dyanmics change due to wear and tear. Finally, in *Barkour-Weight* we validate our approach on a realistic simulation of a full robot. We utilize the simulation of the Barkour robot provided by Caluwaerts et al. (2023), which they have shown to be successful in sim-to-real transfer. We therefore consider it to be a good proxy for real world applications. Note that the observation here includes the last three observations and actions, $s_t = (o_{t-2}, a_{t-2}, o_{t-1}, a_{t-1}, o_t)$, such that the weight could be inferred without any additional input. We illustrate these tasks in Fig. 3.

We split our evaluation into three parts: 1) Does the model learn a useful representation of the HiP? 2) Can we accurately infer the next latent during evaluation? 3) Does this allow us to learn a better policy from the offline dataset?

Learned Representation To evaluate our learned representation of the HiP, we follow common practices from the representation learning community (Nozawa & Sato, 2022). We quantitatively evaluate the learned representation using linear probes and qualitatively evaluate it using T-SNE

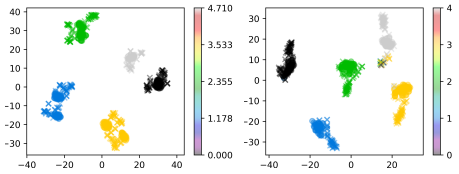


Figure 5: T-SNE visualization of the inferred latents $\tilde{z}$ as crosses ($\times$), and predicted latents $\bar{z}_i = f_{\text{pred}}(\tilde{z}_{i-N_c}, \ldots, \tilde{z}_{i-1})$ as circles ($\circ$).

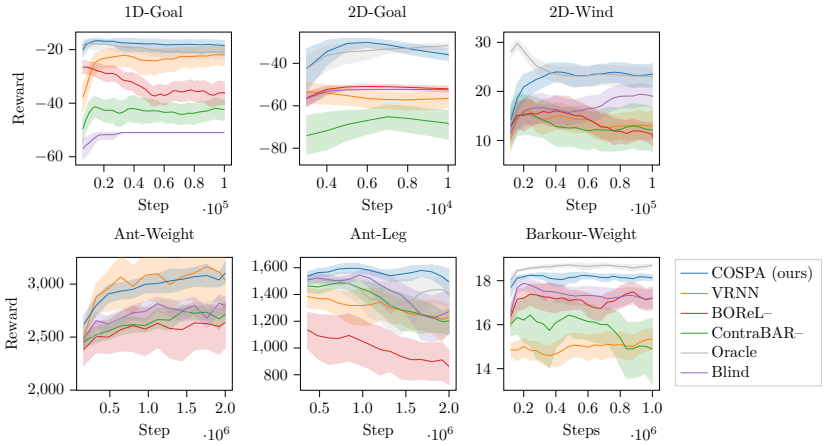


Figure 6: Results of our proposed method on the evaluation environments. Shown are the mean evaluation rewards for 20 seeds per experiment, the shaded areas show 95% confidence intervals.

(Maaten & Hinton, 2008) to visualize the learned representation. As we show in Fig. 4, our method learns a representation which captures the underlying structure well, clustering tasks by HiP. The baselines perform reasonably well in the 2D-Goal task. On the more difficult Ant-Weight task, our method performs significantly better. While we can see some structure in the representations learned by the ContraBAR and BOReL baselines, they are not able to learn a useful representation.

Next Latent Prediction As we need to predict the latent $\tilde{z}_i$ from the given context during deployment, we also evaluate the prediction. We visualize the similarity between predicted latent $\bar{z}_i = f_{pred}(\tilde{z}_{i-N_c:i-1})$ and inferred latents $\tilde{z}_i = g_{\text{enc}}(\tau_i)$ by embedding them in a shared T-SNE visualization. The results are shown in Fig. 5, and we can see a good correspondence between the inferred and predicted latents. The predicted latents are more concentrated than the inferred latents, which can be explained by the denoising properties of the regression loss.

Offline RL Having shown that our method is able to identify a useful latent and predict it during inference, we now evaluate the performance of policies trained with the augmented state $\hat{s} = (s, \tilde{z})$. The results are shown in Fig. 6. Overall, our proposed method performs well, matching or exceeding

	Blind	Oracle	VRNN-	BOReL	ContraBAR-	COSPA (ours)
1D-Goal	-50.95 ± 0.02	-20.79 ± 2.40	$\mathbf{-21.86 \pm 3.96}$	-36.41 ± 4.32	-42.43 ± 3.85	$\mathbf{-18.59 \pm 1.83}$
2D-Goal	-52.34 ± 0.10	-31.48 ± 1.46	-56.20 ± 3.47	-52.09 ± 1.18	-68.77 ± 6.63	$\mathbf{-36.39 \pm 2.72}$
2D-Wind	18.88 ± 2.03	22.81 ± 2.01	13.03 ± 2.75	10.63 ± 2.17	12.04 ± 4.45	$\mathbf{23.63 \pm 2.38}$
Ant-Weight	2797 ± 87	2750 ± 98	$\mathbf{3035 \pm 211}$	2637 ± 188	2715 ± 79	$\mathbf{3104 \pm 100}$
Ant-Leg	$\mathbf{1273 \pm 142}$	1407 ± 92	1231 ± 98	862 ± 121	1201 ± 92	$\mathbf{1493 \pm 97}$
Barkour-Weight	17.19 ± 0.40	18.71 ± 0.05	15.34 ± 0.46	17.22 ± 0.50	14.90 ± 1.38	$\mathbf{18.13 \pm 0.14}$

Table 1: Evaluation reward at the end of the training. Mean and 95% CI across 20 trials. The best performing method and overlapping CIs without access to privileged information are printed **bold**.

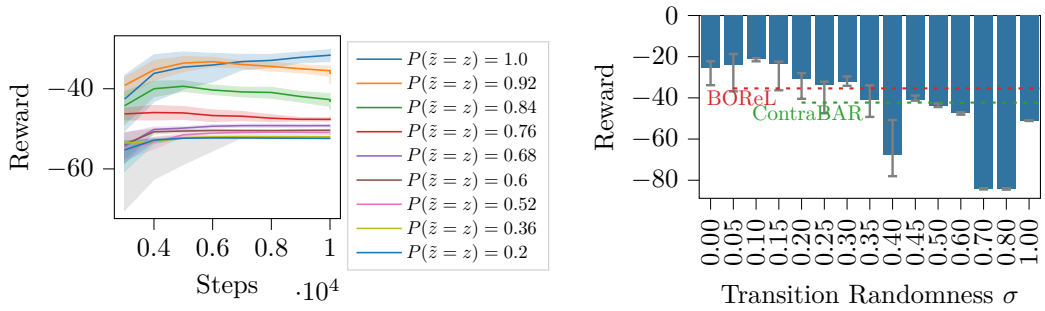


Figure 7: Left: Performance on 2D-Goal when using a noisy ground-truth HiP, with a uniformly random HiP being chosen with probability σ to achieve a desired $P(\tilde{z} = z)$. Right: Performance of our method on 1D-Goal with varying randomness of the HiP transition. Both show means and 95% CI across 10 trials.

the Oracle performance in most tasks. Interestingly, our proposed method and VRNN perform better than the Oracle in the *Ant-Weight* task. One explanation for this is that the learned representation is better able to represent similarity between different HiPs than the simple ground-truth weight value. While the baselines perform well in the simple *1D-Goal* task, they generally do not perform well in the more difficult settings, sometimes performing worse than the *blind* baseline. As noted above, in the *Barkour-Weight* task the state includes the previous two observations and actions, in principle allowing even the *blind* baseline to perform optimally. However, we find that in practice our method still performs significantly better with the inclusion of the inferred latent $\tilde{z}$.

ContraBAR and BOReL The relatively poor performance of the ContraBAR– and BOReL– baselines in some of our experiments might be surprising, we therefore highlight some differences between their settings and ours to explain the difference in performance. Compared to BOReL and (offline) ContraBAR, one signficant difference is the nature of the datasets and environments. While BOReL and offline ContraBAR were designed for datasets generated by task-specific policies for each different HiP, our datasets are generated by a HiP-agnostic policy, making the task inference more challenging. Further, the majority of the experiments by Dorfman et al. (2021) and Choshen & Tamar (2023) focus on settings with varying reward functions, while our setting focuses on settings with varying transition functions. Finally, as outlined above, the techniques of *reward relabeling*, *policy replaying* and *hard negative mining* are not applicable in our setting and are shown to be important in the ablations in Dorfman et al. (2021); Choshen & Tamar (2023).

4.2 Further Experiments

It is interesting that BOReL performs relatively poor in the 2D-Goal task, while achieving a relatively high linear probe accuracy. To investigate this finding, we perfomed an experiment in which we simulate a noisy Oracle that returns the correct HiP with probability $P(\tilde{z} = z)$ and else returns a uniformly random different HiP. The results in Fig. 7 (left) show that even with a relatively high accuracy $P(\tilde{z} = z) = 0.84$ the noise can lead to significantly worse results. This is consistent with results in Yang et al. (2022; 2024), showing the sensitivity of Offline RL to state noise.

We are also interested in the question of when our method should be prefered over Bayes Adaptive approaches such as BOReL and ContraBAR. Intuitively, our method should perform better when the next HiP can be accurately predicted, while Bayes Adaptive methods should be preferable if it is less predictable, i.e. closer to a HiP-MDP than a DP-MDP. In Fig. 7 (right) we show the attained reward when the HiP transition function $P(z_i|z_{1:i-1})$ is changed to a uniformly random HiP with probability σ. The results align with our intuition of our method performing better than BOReL when the HiP is predictable, but worse when it becomes less predictable.

5 Conclusion

We investigated a novel problem setting in Offline Reinforcement Learning, in which the training data is generated from multiple deployments with non-stationary transition and reward functions. The problem is formulated as multiple rollouts of a Dynamic-Parameter MDP, a Hidden-Parameter MDP (HiP-MDP) in which the HiP evolves across trajectories. We proposed a method using contrastive learning that learns a representation of the HiP, predicts the HiP during evaluation and trains a policy conditioned on it. We showed that our method is able to learn a useful representation of the HiP, allowing us to train a policy that often performs better than baseline methods in experiments.

Acknowledgements

J.A. was supported by the Microsoft Research Asia D-CORE program. T.O. was partially supported by JSPS KAKENHI Grant Number JP23K18476.

A Further Related Work

We will here discuss additional related work, first on Nonstationary RL, then Multi-Task Offline RL and finally Contrastive Learning in RL.

Nonstationary RL The perhaps closest related work has been presented by Xie et al. (2021), in which they propose the DP-MDP formulation which we use in our problem setting. Unlike us, their work focuses on an online lifelong-RL setting with a single deployment for both training and evaluation. They also proposed the method LILAC, an offline variation of which we use as a baseline. Chen et al. (2022) address non-stationary environments with piece-wise stable context. They address a setting where parts of each episode are stationary, but do not consider structure between tasks. Wang et al. (2023) derive a robust method to test whether nonstationarity occurs in a given offline RL dataset. Their work focuses on identifying whether such non-stationarity occurs, while we focus on how we can learn a policy that adapts to it. Yin & Wang (2021) provide a theoretical investigation of nonstationary Offline RL in a setting where the nonstationary occurs during each episode but not between episodes, while in our setting it occurs between episodes and not within each episode. Dulac-Arnold et al. (2021) also discusses non-stationarity introduced by wear and tear and proposes environments to investigate these challenges in online RL. Chandak et al. (2022) consider off-policy evaluation in a generalization of the DP-MDP in which the next HiP depends on the previous HiPs and the actions taken within the previous episode. Off-policy evaluation is an important step towards Offline RL, but it is not immediately clear how to extend their method to the control setting or to complex environments which we address in our work.

Multi-Task Offline RL A related setting to ours is Multi-Task Offline RL. The main difference is that in Multi-Task RL we are given the task identity during training and therefore do not have to learn an inference model in an unsupervised setting. Li et al. (2020) and Li et al. (2022) propose methods to address such a setting where the task ID is given during training and can therefore not be applied to our setting. Liu et al. (2022) and Xue et al. (2023) address the Offline Transfer Learning setting, in which a large amount of source domain data is available but limited target domain data with different dynamics.

Contrastive Learning CPC (Oord et al., 2019) has been used in RL before, the original authors themselves used it as an auxiliary loss in Atari tasks (Oord et al., 2019). ContraBAR (Choshen & Tamar, 2023) proposed to use CPC to learn Bayes-Optimal policies in the Bayes adaptive RL setting (Zintgraf et al., 2020). Aside from CPC, other contrastive learning approaches have been applied in RL, for representation learning (Kipf et al., 2019; Srinivas et al., 2020; van der Pol et al., 2020), or for domain inference networks in multi-task RL (Li et al., 2022; Lan et al., 2023).

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *NeurIPS 2021*, January 2022. doi: 10.48550/arXiv.2108.13264. URL http://arxiv.org/abs/2108.13264.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016. URL http://arxiv.org/abs/1607.06450.

Ken Caluwaerts, Atil Iscen, J. Chase Kew, Wenhao Yu, Tingnan Zhang, Daniel Freeman, Kuang-Huei Lee, Lisa Lee, Stefano Saliceti, Vincent Zhuang, Nathan Batchelor, Steven Bohez, Federico Casarini, Jose Enrique Chen, Omar Cortes, Erwin Coumans, Adil Dostmohamed, Gabriel Dulac-Arnold, Alejandro Escontrela, Erik Frey, Roland Hafner, Deepali Jain, Bauyrjan Jyenis, Yuheng Kuang, Edward Lee, Linda Luu, Ofir Nachum, Ken Oslund, Jason Powell, Diego Reyes, Francesco Romano, Feresteh Sadeghi, Ron Sloat, Baruch Tabanpour, Daniel Zheng, Michael Neunert, Raia Hadsell, Nicolas Heess, Francesco Nori, Jeff Seto, Carolina Parada, Vikas Sindhwani, Vincent Vanhoucke, and Jie Tan. Barkour: Benchmarking Animal-level Agility with Quadruped Robots, May 2023. URL http://arxiv.org/abs/2305.14654.

Yash Chandak, Shiv Shankar, Nathaniel D. Bastian, Bruno Castro da Silva, Emma Brunskill, and Philip S. Thomas. Off-Policy Evaluation for Action-Dependent Non-stationary Environments. In *NeurIPS 2022*, 2022. URL https://openreview.net/forum?id=PuagBLcAf8n.

Xiaoyu Chen, Xiangming Zhu, Yufeng Zheng, Pushi Zhang, Li Zhao, Wenxue Cheng, Peng Cheng, Yongqiang Xiong, Tao Qin, Jianyu Chen, and Tie-Yan Liu. An Adaptive Deep RL Method for Non-Stationary Environments with Piecewise Stable Context. In *NeurIPS 2022*, December 2022. URL http://arxiv.org/abs/2212.12735.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *EMMNLP*, 2014. doi: 10.3115/v1/d14-1179.

Era Choshen and Aviv Tamar. ContraBAR: Contrastive Bayes-Adaptive Deep RL. In *ICML 2023*, June 2023. URL http://arxiv.org/abs/2306.02418.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A Recurrent Latent Variable Model for Sequential Data. In *NeurIPS 2015*. doi: 10.48550/arXiv.1506.02216. URL http://arxiv.org/abs/1506.02216.

Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline Meta Reinforcement Learning – Identifiability Challenges and Effective Data Collection Strategies. In *NeurIPS 2021*, October 2021. URL https://openreview.net/forum?id=IBdEfhLveS.

Finale Doshi-Velez and George Konidaris. Hidden Parameter Markov Decision Processes: A Semi-parametric Regression Approach for Discovering Latent Task Parametrizations. In *IJCAI 2016*, August 2013. URL https://arxiv.org/abs/1308.3513v1.

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

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – A Differentiable Physics Engine for Large Scale Rigid Body Simulation. In *NeurIPS 2021 Datasets and Benchmarks Tracl*, 2021. URL http://arxiv.org/abs/2106.13281.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for Deep Data-Driven Reinforcement Learning, February 2021. URL http://arxiv.org/abs/2004.07219.

Scott Fujimoto and Shixiang Shane Gu. A Minimalist Approach to Offline Reinforcement Learning. In *NeurIPS 2021*, December 2021. doi: 10.48550/arXiv.2106.06860. URL http://arxiv.org/abs/2106.06860.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *ICML 2018*, February 2018. URL http://arxiv.org/abs/1802.09477.

Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale, April 2021. URL http://arxiv.org/abs/2104.08212.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR 2015*, 2015. URL http://arxiv.org/abs/1412.6980.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive Learning of Structured World Models. In *ICLR 2020*, volume 2, pp. 235–239, November 2019. URL http://arxiv.org/abs/1911.12247.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline Reinforcement Learning with Implicit Q-Learning, October 2021. URL http://arxiv.org/abs/2110.06169.

Aviral Kumar, Rishabh Agarwal, Tengyu Ma, Aaron Courville, George Tucker, and Sergey Levine. DR3: Value-Based Deep Reinforcement Learning Requires Explicit Regularization. In *ICLR 2022*, May 2022. URL https://openreview.net/forum?id=POvMvLi91f.

Siming Lan, Rui Zhang, Qi Yi, Jiaming Guo, Shaohui Peng, Yunkai Gao, Fan Wu, Ruizhi Chen, Zidong Du, Xing Hu, Xishan Zhang, Ling Li, and Yunji Chen. Contrastive Modules with Temporal Attention for Multi-Task Reinforcement Learning, November 2023. URL http://arxiv.org/abs/2311.01075.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems, November 2020. URL http://arxiv.org/abs/2005.01643.

Jiachen Li, Quan Vuong, Shuang Liu, Minghua Liu, Kamil Ciosek, Keith Ross, Henrik Iskov Christensen, and Hao Su. Multi-task Batch Reinforcement Learning with Metric Learning. In *NeurIPS 2020*, 2020. URL http://arxiv.org/abs/1909.11373.

Lanqing Li, Rui Yang, and Dijun Luo. FOCAL: Efficient Fully-Offline Meta-Reinforcement Learning via Distance Metric Learning and Behavior Regularization. In *ICLR 2021*, February 2022. URL https://openreview.net/forum?id=8cpHIfgY4Dj.

Jinxin Liu, Zhang Hongyin, and Donglin Wang. DARA: Dynamics-Aware Reward Augmentation in Offline Reinforcement Learning. March 2022. URL https://openreview.net/forum?id=9SDQB3b68K.

Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *MLR*, 9(86):2579–2605, 2008. ISSN 1533-7928. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Kento Nozawa and Issei Sato. Empirical Evaluation and Theoretical Analysis for Representation Learning: A Survey, April 2022. URL http://arxiv.org/abs/2204.08226.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding, January 2019. URL http://arxiv.org/abs/1807.03748.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning, October 2019. URL http://arxiv.org/abs/1910.00177.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation, October 2018. URL http://arxiv.org/abs/1506.02438.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. In *ICML 2020*, 2020. URL http://arxiv.org/abs/2004.04136.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In *IROS 2017*, 2017. URL http://arxiv.org/abs/1703.06907.

Elise van der Pol, Thomas Kipf, Frans A. Oliehoek, and Max Welling. Plannable Approximations to MDP Homomorphisms: Equivariance under Actions. In *AAMAS 2020*, 2020. URL http://arxiv.org/abs/2002.11963.

Jitao Wang, Chengchun Shi, and Zhenke Wu. A Robust Test for the Stationarity Assumption in Sequential Decision Making. In *ICML 2023*, 2023. URL https://proceedings.mlr.press/v202/wang23ai.html.

Annie Xie, James Harrison, and Chelsea Finn. Deep Reinforcement Learning amidst Continual Structured Non-Stationarity. In *ICML 2021*, pp. 11393–11403. PMLR, July 2021. URL https://proceedings.mlr.press/v139/xie21c.html.

Zhenghai Xue, Qingpeng Cai, Shuchang Liu, Dong Zheng, Peng Jiang, Kun Gai, and Bo An. State Regularized Policy Optimization on Data with Dynamics Shift. In *NeurIPS 2023*. arXiv, October 2023. URL http://arxiv.org/abs/2306.03552. arXiv:2306.03552 [cs].

Rui Yang, Chenjia Bai, Xiaoteng Ma, Zhaoran Wang, Chongjie Zhang, and Lei Han. RORL: Robust Offline Reinforcement Learning via Conservative Smoothing. In *NeurIPS 2022*, May 2022. URL https://openreview.net/forum?id=_QzJJGH_KE.

Rui Yang, Han Zhong, Jiawei Xu, Amy Zhang, Chongjie Zhang, Lei Han, and Tong Zhang. Towards Robust Offline Reinforcement Learning under Diverse Data Corruption. In *ICLR 2024*. arXiv, January 2024. URL http://arxiv.org/abs/2310.12955.

Ming Yin and Yu-Xiang Wang. Towards Instance-Optimal Offline Reinforcement Learning with Pessimism. In *NeurIPS2021*, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/212ab20dbdf4191cbcdcf015511783f4-Abstract.html.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning. *ICLR 2020*, pp. 1–20, October 2020. URL http://arxiv.org/abs/1910.08348.

Karl Johan Åström. Optimal Control of Markov Processes with Incomplete State Information I. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965. URL http://lup.lub.lu.se/record/8867084.

B Theoretical Analysis of CPC Inference

For completeness, we show our derivation which follows the original CPC paper closely (Oord et al., 2019) and is related to those by Choshen & Tamar (2023), which studies CPC in a Bayes-Adaptive MDP setting.

One difference to note between our setting and that considered in Choshen & Tamar (2023), is the change of the hidden-parameter between episodes and therefore between inputs to the CPC model. While in their setting the hidden-parameter is the same for all inputs (transitions), in our setting it evolves between inputs (trajectories). We provide a full illustration of the data generation graphical model and CPC model in Fig. 8:

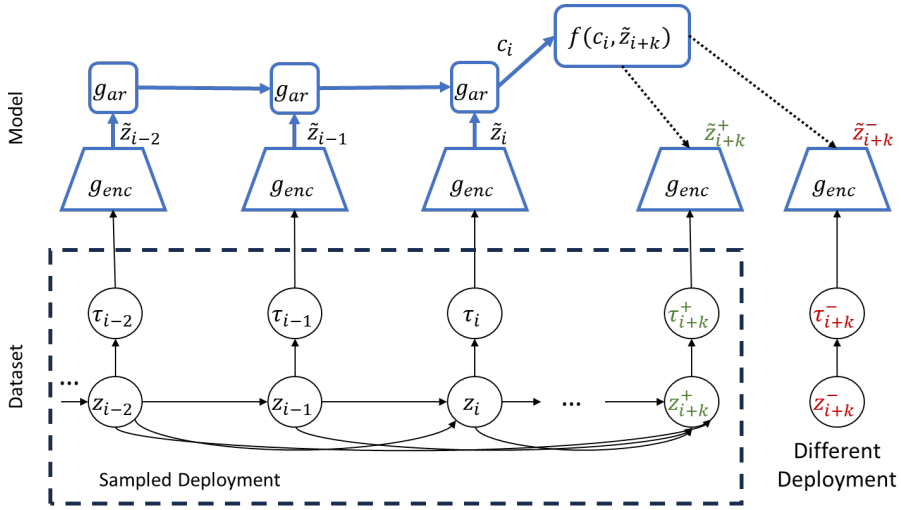


Figure 8: Illustration of the data generation and our model.

To show that CPC maximizes the mutual information between c_i and τ_{i+k}, we use the same derivation as 2.3 and A.1 of Oord et al. (2019):

Lemma B.1. *Let the InfoNCE loss in equation 2 be jointly minimized by $f, g_{\text{enc}}, g_{\text{ar}}$, then for any trajectory τ, with $c_i = g_{\text{ar}}(g_{\text{enc}}(\tau_{i-1}, \tau_{i-2}, \ldots, \tau_1))$, we have*

$$f(\tau_{i+k}, c_i) \propto \frac{P(\tau_{i+k}|c_i)}{P(\tau_{i+k})}$$

Proof. This follows from the InfoNCE loss (2) being the categorical cross-entropy of correctly classifying the positive sample in the given batch of samples $B = (\tau^1_{i+k}, \ldots, \tau^N_{i+k})$. Following Oord et al. (2019), we can write the classification problem as learning a classifier $P(d = o|B, c_i)$ with $[d = o]$ being the indicator of the positive sample (omitting the subscript $i + k$ for all τ):

$$\begin{aligned} p(d = o|B, c_i) &= \frac{p(\tau^o|c_i) \prod_{l \neq o} p(\tau^l)}{\sum_{j=1}^N p(\tau^j|c_i) \prod_{l \neq j} p(\tau^l)} \\ &= \frac{\frac{p(\tau^o|c_i)}{p(\tau^o)}}{\sum_{j=1}^N \frac{p(\tau^j|c_i)}{p(\tau^j)}}, \end{aligned}$$

which we can see is proportional to $\frac{P(\tau_{i+k}|c_i)}{P(\tau_{i+k})}$. As such, the optimal value of the classification problem learned by $f(\tau_{i+k}, c_i)$ is proportional to it as well. □

Lemma B.2. *Let the InfoNCE loss in equation 2 be jointly minimized by $f, g_{\text{enc}}, g_{\text{ar}}$, then for any trajectory τ, with $c = g_{\text{ar}}(g_{\text{enc}}(\tau_{i-1}, \tau_{i-2}, \ldots, \tau_1)$, we have*

$$\mathcal{L}_{repr} \geq \log(N-1) - I(\tau_{i+k}; c_i)$$

Proof. Let B_{neg} be the negative samples in a batch B. By inserting the optimal value of $f(\tau_{i+k}, c_i)$ into the loss function (2), we get (omitting the subscript $i+k$ for all τ)

$$\begin{aligned}
\mathcal{L}_{repr} &= -\mathbb{E}\log\left[\frac{\frac{p(\tau^+|c_i)}{p(\tau^+)}}{\frac{p(\tau^+|c_i)}{p(\tau^+)} + \sum_{\tau^- \in B_{\text{neg}}} \frac{p(\tau^-|c_i)}{p(\tau^-)}}\right] \\
&= \mathbb{E}\log\left[1 + \frac{p(\tau^+)}{p(\tau^+|c_i)} \sum_{\tau^- \in B_{\text{neg}}} \frac{p(\tau^-|c_i)}{p(\tau^-)}\right] \\
&\approx \mathbb{E}\log\left[1 + \frac{p(\tau^+)}{p(\tau^+|c_i)}(N-1)\mathbb{E}_{\tau^-}\frac{p(\tau^-|c_i)}{p(\tau^-)}\right] \\
&= \mathbb{E}\log\left[1 + \frac{p(\tau^+)}{p(\tau^+|c_t)}(N-1)\right] \\
&\geq \mathbb{E}\log\left[\frac{p(\tau^+)}{p(\tau^+|c_t)}(N-1)\right] \\
&= H(\tau^+|c_i) - H(\tau^+) + \log(N-1) \\
&= -I(\tau^+; c_i) + \log(N-1)\,.
\end{aligned}$$

□

C Implementation Details

Our implementation is available on github, additional implementation details can be found there. Note that we smooth the reward plots with a moving window of size 3 for clarity. Confidence intervals and means are calculated across trials consisting of representation learning and offline RL with a fixed dataset for all trials.

C.1 Environments

We use the same DP-MDPs and thus the same sets of HiPs in training and evaluation.

C.1.1 Low-Dimensional Tasks

In 1D-Goal, the state space is $S = [-2, 2]$ and action space is $A = [-0.1, 0.1]$. The initial state is uniformly sampled from S and the goal g is at $g = z \in \{-1, 1\}$, depending on the hidden parameter z. The deterministic transition function is $s_{t+1} = s_t + a_t$ and the reward is the negative absolute distance to the goal location $r_t = -|s_t - g|$. The hidden parameter deterministically switches each episode, i.e. $P(z_{i+1} = -1|z_i = 1) = P(z_{i+1} = 1|z_i = -1) = 1$, with each deployment consisting of 10 episodes. For the experiment in Fig. 7 (right), we change the transition function to $P(z_{i+1} = 1|z_i = -1) = P(z_{i+1} = -1|z_i = 1) = 1 - \sigma/2$.

Our 2D-tasks are a continuous environment with action space $S = [-2, 2]^2$ and action space $A = [-0.1, 0.1]^2$. The transition function is deterministic and follows $s_{t+1} = s_t + a_t$. The goal is to navigate to a goal location. In *2D Goal*, the agent starts at a uniformly random location and the reward function is the negative Euclidean distance to the goal location $r_t = -|s_t - g|_2$. The goal location g lies on the unit circle and is determined by the hidden-parameter z: $g(z) = (\sin(z), \cos(z))^\intercal$. The hidden-parameter follows a triangle-wave with period 8 with $z \in [0, \frac{3}{2}\pi]$, i.e.$z \in \{0, \frac{3}{8}\pi, \frac{6}{8}\pi, \frac{9}{8}\pi, \frac{3}{2}\pi\}$, and each deployment is 20 episodes long. This task is similar to "Semi-Circle" in Dorfman et al. (2021).

In *2D Wind*, the agent always starts at the origin $(0,0)^\intercal$ and the goal location is always $g = (1,0)^\intercal$. Here we use a sparse unit reward $r_t = 1$ if the distance $|s_t - g|_2 < 0.2$, else no reward is given $r_t = 0$. The dynamics are changed to include a disturbance, $s_{t+1} = s_t + a_t + 0.09(\sin(z), \cos(z))^\intercal$, where z follows a sawtooth-wave with period five on $[0, 2\pi]$, i.e. $z \in \{(0, \frac{2}{5}\pi, \frac{4}{5}\pi, \frac{6}{5}\pi, \frac{8}{5}\pi\}$, and each deployment is 20 episodes long. This task is based on "Wind" in Dorfman et al. (2021).

C.1.2 Ant

We modify the Ant environment provided in Brax (Freeman et al., 2021) to allow for multiple different robot configurations. We keep the original reward function, which rewards the robot for forward movement.

In *Ant-Weight*, we modify the mass of the base (the sphere) of the robot by multiplying it by the current hidden-parameter z. For the HiP evolution we here use a sawtooth-wave with period 5 and $z \in [0.5, 2.5]$, i.e. $z \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$, with each deployment lasting 20 episodes. We use the default observation.

In *Ant-Leg* we multiply the length of each leg, both the "femur" and "tibia", by the hidden-parameter z. We again use a sawtooth-wave with period 5 and $z \in [0.75, 1.25]$, i.e. $z \in \{0.75, 0.875, 1.0, 1.125, 1.25\}$, with 20 episodes per deployment. We alter the observation by removing the z-component of the position of the base, as it otherwise directly represents the hidden-parameter in the first timestep.

C.1.3 Barkour

Unlike for the Ant experiment, for Barkour we now use Mujoco-MJX to simulate the robot and modify the environment provided in the Mujoco-MJX Tutorial.[3] The Barkour environment usually trains the robot to track an input command linear velocity and angular velocity, which is sampled uniformly during training. As this sampling adds a high amount of variance to reward function, we change the reward-function by setting a constant desired forward velocity and removing the reward for angular velocity tracking. We use this altered reward to finetune a pretrained Barkour policy and then also to collect the dataset and evaluate the Offline RL policy. To simulate the robot having to carry a varying load, we alter the weight of the chassis by multiplying it with the hidden-parameter, following a sawtooth-wave with period 5 in $z \in [1.0, 4.0]$, i.e. $z \in \{1.0, 1.75, 2.5, 3.25, 4.0\}$, for 15 episodes per deployment. This task is inspired by "minitaur-payload" in Xie et al. (2021).

D Method Details

We outline the main implementation details here, while additional details can be found in the implementation on github. We use Adam (Kingma & Ba, 2015) in all experiments.

D.1 CPC Implementation

We implement g_{enc} as a two layer MLP with 128 units per layer and ReLU activations and g_{ar} as a GRU with 16 units in the low-dimensional and 32 units in the high-dimensional tasks. The classifier f is implemented as an MLP with two hidden layers with 128 units per layer and ReLU activations. As the initial parameters performed well across tasks, we did not perform extensive grid-search as for the baselines, and only searched over latent dimensionality $\{2, 4, 6, 8\}$ per task.

The positive and negative samples are sampled as follows: To obtain positive and negative trajectories we sample one positive deployment $d^+ = (\tau_1^+, \ldots, \tau_M^+)$, N^- negative deployments $\{d^{-,j}\}_{j=1}^{N^-}$, with $d^{-,j} = (\tau_1^{-,j}, \ldots, \tau_M^{-,j})$ and deployment step $i \sim U([N_{\text{CPC}}, M-k])$. Then we obtain the context

[3] https://github.com/google-deepmind/mujoco/blob/721e2d5589d3fdafd440009374a31521214088b7/mjx/tutorial.ipynb

Hyperparameter	Gridsearch Values BORel, VRNN
Hidden Units Decoder/Encoder	$\{128, 256\}$
Decoder Hidden Layers	$\{2, 3\}$
VAE β	$\{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 0\}$
Latent Dimensionality	$\{4, 8\}$

Table 2: Hyperparameters values used in Grid-Search for BOReL and VRNN

encoding $c_i = g_{\text{ar}}(g_{\text{enc}}(\tau_i), \ldots, g_{\text{enc}}(\tau_{i-N_{\text{CPC}}}))$ and use it as input to f with positive sample τ^{+}_{i+k} and negative samples $\{\tau^{-,j}_{i+k}\}_{j=1}^{N}$.

D.2 Bayesian RL Baseline Implementation

We implement our Bayesian RL baselines based on VariBAD (Zintgraf et al., 2020)/BOReL (Dorfman et al., 2021) and ContraBAR (Choshen & Tamar, 2023). We note that while they use two episodes to measure the performance of the agent, we measure the reward over a single episode due to hidden-parameter changing after each episode in our setting. We truncate the trajectory length used for HiP inference to 10 in Ant-Leg and Barkour-Weight and 50 in Ant-Weight, to allow for efficient training. We freeze the last inferred latent and use it for the remainder of the episode.

D.2.1 BOReL Implementation

We use a recurrent encoder consisting of two fully connected layers with RELU activations, followed by a GRU, followed by a fully connected layer each to output μ and Σ. The reward and transitions decoders each consist of two or three hidden layers with ReLU activations, where the latent is input into the second layer. We only use either the reward or transition decoder depending on whether the reward or transition function changes in an environment. We found RL training to perform better when only conditioning on the mean μ without the covariance matrix Σ. We perform a grid-search over representation hyperparameters as shown in the Table 2 on 1D-Goal, 2D-Goal, 2D-Wind, Ant-Weight and Ant-Leg. Due to resource constraints we do not perform a full grid-search on barkour, but did our best effort to choose well-performing hyperaprameters and evaluated different choices for the latent-dimensionality.

D.2.2 ContraBAR Implementation

For ContraBAR we found the network structure of encoder and classifier to strongly impact the achieved performance. For fairness of comparison we evaluated two different structures: Firstly, a network structure as proposed by Choshen & Tamar (2023) with separate encoders for reward, state, and action, a GRU with larger dimensionality $d = 64$ and a small, single-layer classifier f. Secondly, we evaluated the same structure as used in our approach, with more powerful fully-connected encoders and classifiers, but smaller GRU dimensionality $d = 4, 8$. We found the latter structure to perform better on the 2D tasks, while the former usually resulted in better representations in the other tasks. As recommended by Choshen & Tamar (2023), we use the Action-GRU only in tasks where the transition function changes, omitting it otherwise. We performed a gridsearch to optimize the hyperparameters over the values shown in Table 3 on 1D-Goal, 2D-Goal, 2D-Wind, Ant-Weight and Ant-Leg. Due to resource constraints we do not perform a full grid-search on Barkour, but did our best effort to choose well-performing hyperaprameters and evaluated different choices for the latent-dimensionality.

D.3 VRNN Implementation

We evaluated different approaches to creating an offline variant of LILAC with different network architectures and training mechanisms, and report the most successful one we found. As it is different in network structure and training method we refer to it simply as *VRNN* in our experiments. We base

Hyperparameter	Gridsearch Values ContraBAR
Hidden Units Decoder/Encoder	$\{64, 128, 256\}$
Encoder Architecture	{SplitEnc, MLPEnc}
Encoder Hidden Layers	$\{1, 2\}$
Latent Dimensionality	$\{2, 4, 8\}$

Table 3: Hyperparameters values used in Grid-Search for ContraBAR

our implementation on a VRNN (Chung et al.), a type of dynamic VAE with an LSTM prior. The encoder is a two layer MLP with ReLU activations, the observation and latent feature extractors are implemented as fully connected layers with ReLU activations and the learned prior is implemented an LSTM followed by a fully connected layer with ReLU activations and two linear output layers for μ, Σ. The decoders are MLPs with two or three layers and ReLU activations. We train the VRNN by sampling two different transitions from each episode, where one transition is used in the encoder and a different transition is used in the decoder, to prevent the encoder from simply learning the state or reward functions. We had issues with innacurate predictions and thus, during evaluation, we condition the policy on a sampled latent $\tilde{z}$ of an episode from our dataset with the same HiP z. This method can therefore be considered a semi-oracle that uses oracle information during evaluation but not during training. As in BOReL, we found RL training to perform better when only conditioning on the mean μ without the covariance matrix Σ. Hyperparameters are optimized by gridsearch over values shown in Table 2 on 1D-Goal, 2D-Goal, 2D-Wind, Ant-Weight and Ant-Leg. Due to resource constraints we do not perform a full grid-search on Barkour, but did our best effort to choose well-performing hyperaprameters and evaluated different choices for the latent-dimensionality.

D.4 Offline Reinforcement Learning Implementation

On each task, we use the same network structure and RL hyper-parameters for all methods. We do deviate from the original parameters of TD3+BC (Fujimoto & Gu, 2021) as shown in Table 4.

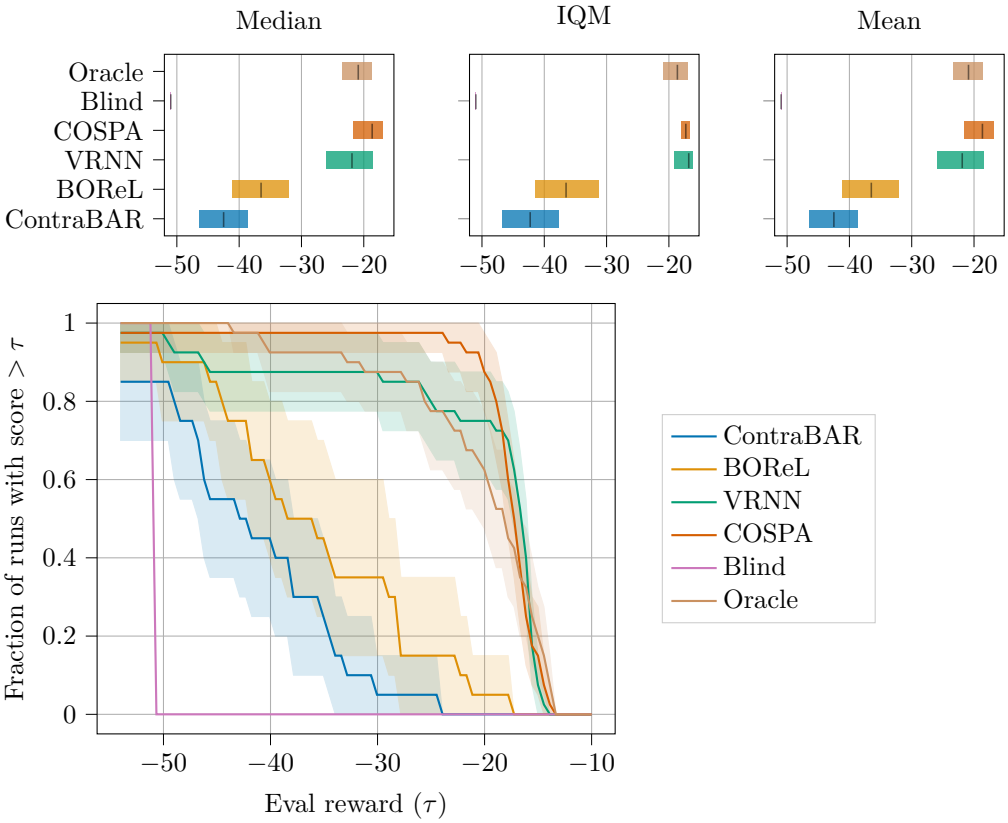
Hyperparameter	1D-Goal	2D-Win	2D-Goal	Ant-Weight, Ant-Leg	Barkour
Critic Network	[256, 256]	[256, 256]	[256, 256]	[128, 128]	[128, 128, 128]
Policy Network	[256, 256]	[256, 256]	[256, 256]	[128, 128]	[128, 128, 128]
Layer Norm	Yes	Yes	No	Yes	Yes
BC λ	2.5	6.5	6.5	6.5	6.5
Learning Rate	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
Batch Size	512	512	512	512	512

Table 4: TD3 BC Hyperparameters

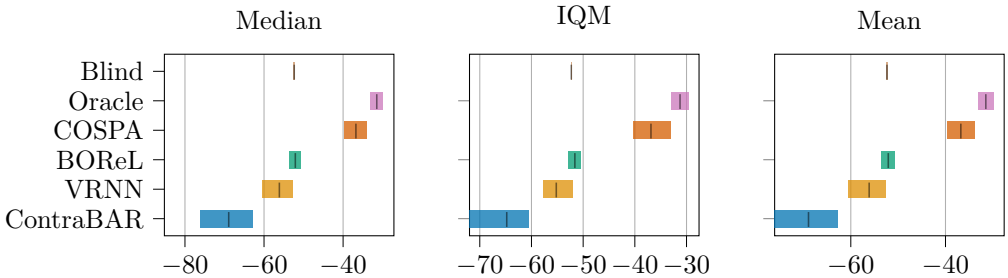
E RLiable Visualization

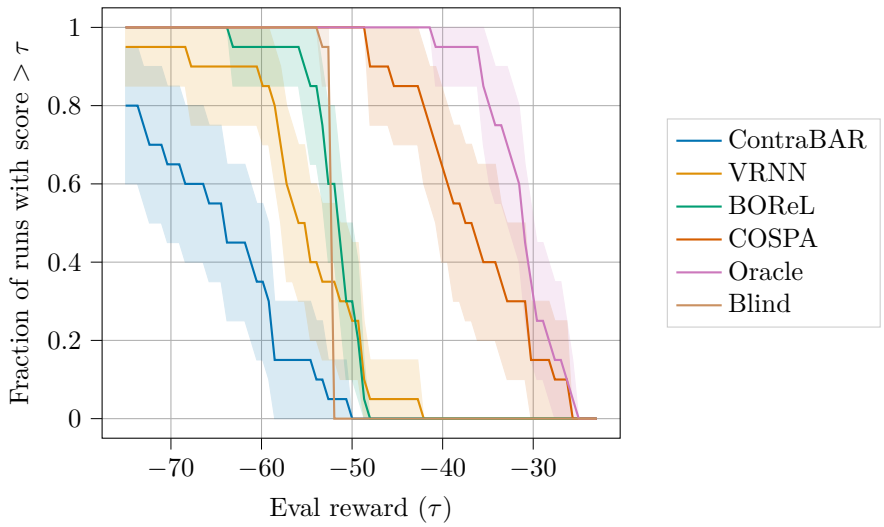
To provide more informative evaluation metrics than the mean reward values reported in the main text, we provide additional visualizations as suggested by Agarwal et al. (2022), using the RLiable package.

E.1 1D-Goal

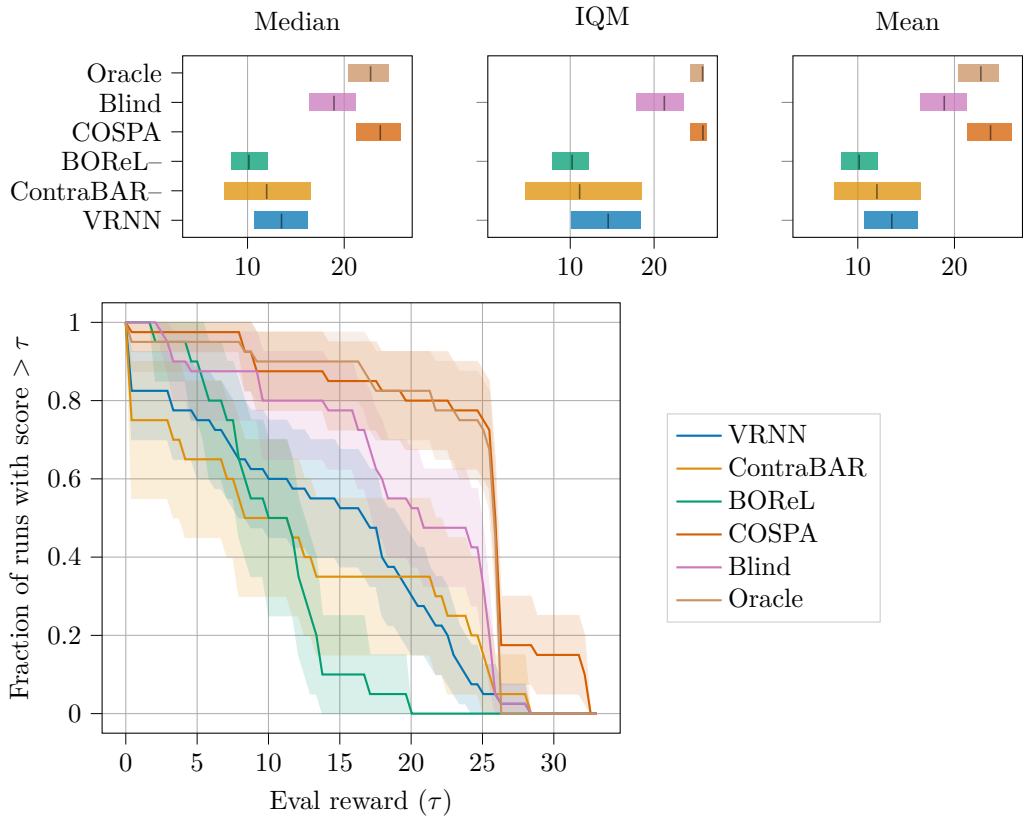


E.2 2D-Goal

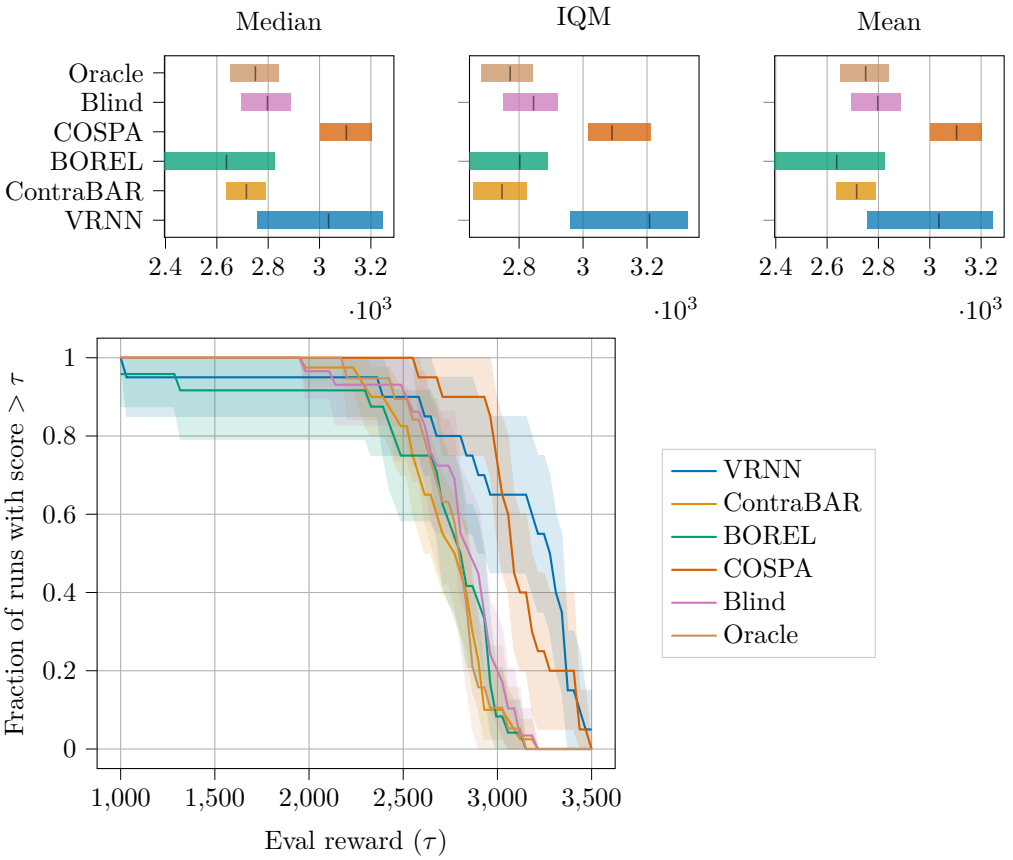



Fraction of runs with score > τ
Eval reward (τ)
ContraBAR
VRNN
BOReL
COSPA
Oracle
Blind

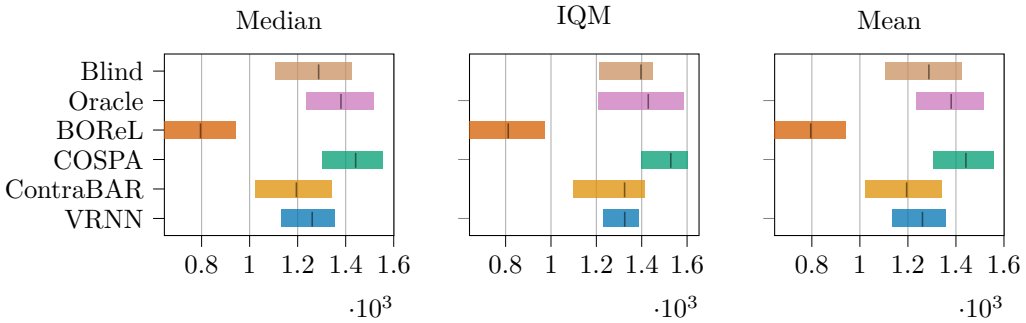
E.3 2D-Wind

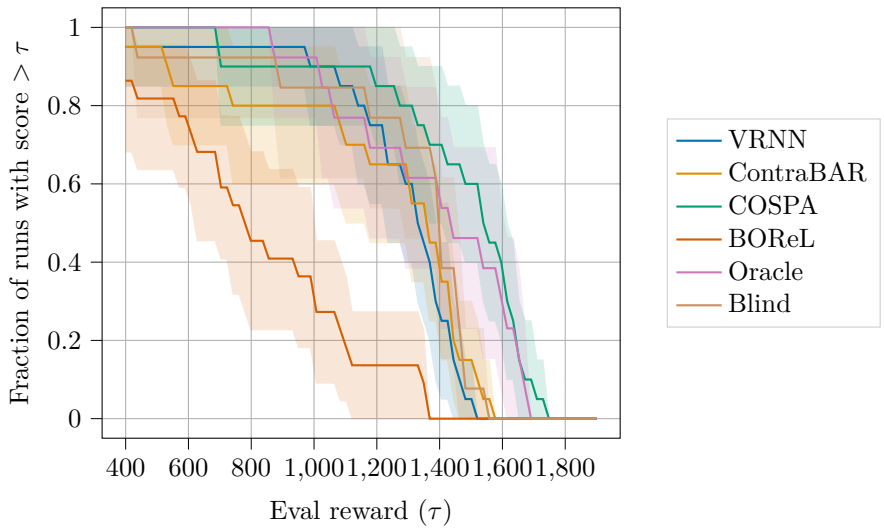
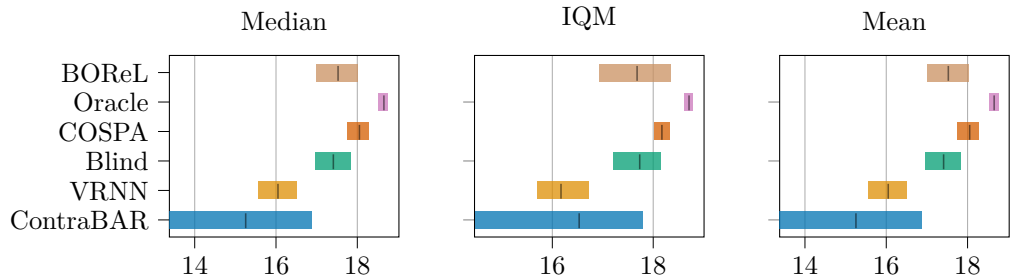
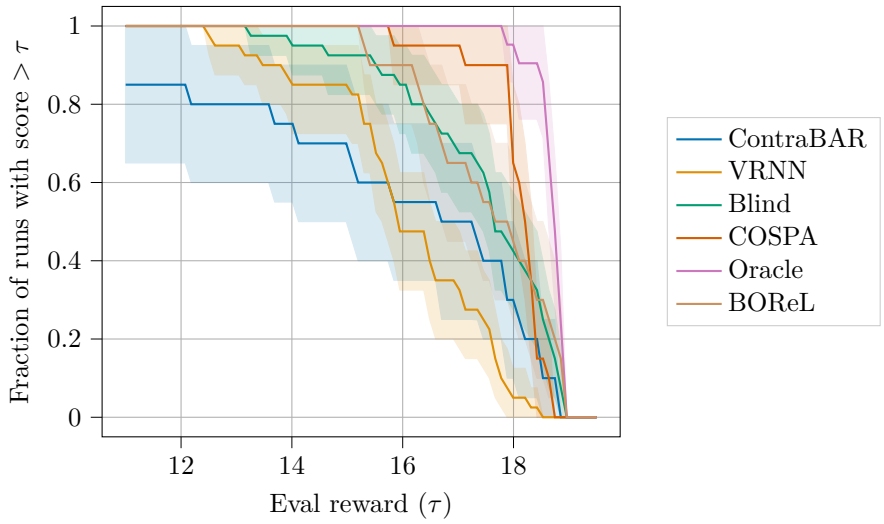

Median
IQM
Mean
Oracle
Blind
COSPA
BOReL–
ContraBAR–
VRNN
Fraction of runs with score > τ
Eval reward (τ)
VRNN
ContraBAR
BOReL
COSPA
Blind
Oracle

E.4 Ant-Weight



E.5 Ant-Leg





E.6 Barkour-Weight