Entropy Regularized Task Representation Learning for Offline Meta-Reinforcement Learning

Mohammadreza Nakhaeinezhadfard[1], **Aidan Scannell**[1,2], **Joni Pajarinen**[1]

[1]Department of Electrical Engineering and Automation (EEA), Aalto University, Finland
[2]Finnish Center for Artifcial Intelligence, Finland
{mohammadreza.nakhaei, aidan.scannell, joni.pajarinen}@aalto.fi

Abstract

Offline meta-reinforcement learning aims to equip agents with the ability to rapidly adapt to new tasks by training on data from a set of different tasks. Context-based approaches utilize a history of state-action-reward transitions – referred to as the context – to infer representations of the current task, and then condition the agent, i.e., the policy and value function, on the task representations. Intuitively, the better the task representations capture the underlying tasks, the better the agent can generalize to new tasks. Unfortunately, context-based approaches suffer from distribution mismatch, as the context in the offline data does not match the context at test time, limiting their ability to generalize to the test tasks. This leads to the task representations overfitting to the offline training data. Intuitively, the task representations should be independent of the behavior policy used to collect the offline data. To address this issue, we approximately minimize the mutual information between the distribution over the task representations and behavior policy by maximizing the entropy of behavior policy conditioned on the task representations. We validate our approach in MuJoCo environments, showing that compared to baselines, our task representations more faithfully represent the underlying tasks, leading to outperforming prior methods in both in-distribution and out-of-distribution tasks.

Code —
https://github.com/MohammadrezaNakhaei/ER-TRL

1 Introduction

The goal of offline reinforcement learning (RL; Prudencio, Maximo, and Colombini 2023; Levine et al. 2020) is to leverage offline datasets to learn policies that can accomplish tasks without interacting with the environment. It is a promising approach for real-world applications where online RL may be expensive or dangerous, such as robotics (Zhou et al. 2023; Luo et al. 2023) and healthcare (Emerson, Guy, and McConville 2023; Zhu, Li, and Georgiou 2023). Offline meta-RL (OMRL) extends offline RL to the meta-RL setting, where the goal is to solve a previously unseen task, by leveraging behaviors learned from a set of training environments. In contrast to meta-RL, OMRL considers the setting where the agent does not directly interact with the training environments but instead has access to offline datasets from each training environment.

A promising approach to OMRL is context encoding (Li et al. 2020; Li, Yang, and Luo 2020; Yuan and Lu 2022; Zhao, Zhou, and Liu 2023; Wang et al. 2023; Gao et al. 2024), where a context encoder learns a representation of the task from a history of state-action-reward transitions, referred to as context. The agent is then conditioned on this learned task representation. For example, in off-policy RL, the policy and value functions are conditioned on the task representation, which enables adaptation to different tasks, including previously unseen ones.

As highlighted by Gao et al. (2024), context-based OMRL suffers from a distribution shift in the context encoder. This is because the context encoder is trained with contexts collected by the behavior policy, but at test time the context is collected by a different policy, including the learned policy conditioned on a prior task representation. As such, there is a discrepancy between the contexts at train and test time. This limits the context encoder's ability to infer the correct task at test time, which consequently limits the agent's ability to adapt to previously unseen tasks.

One approach to alleviating this distribution mismatch is to minimize the mutual information between the task representations and the behavior policy. The intuition is that for environments that require dissimilar policies for each task, the context encoder should not rely on the characteristics of the behavior policy. As such, we would like our learned task representations to be independent of the behavior policy in order to reduce the impact of the behavior policy on the task representation. We sidestep the context shift issue by showing that minimizing the mutual information between the task representations and the behavior policy is equivalent to maximizing the entropy of a meta-behavior policy conditioned on the learned task representation.

Our contributions are as follows:

- We present *Entropy Regularized Task Representation Learning* (ER-TRL), an OMRL method that improves the ability of context-based OMRL methods to generalize to previously unseen environments. ER-TRL reduces the amount of context distribution shift by leveraging a generative adversarial network (GAN; Goodfellow et al. 2014) to minimize the mutual information between the task representations and the behavior policy, by estimat-

ing the entropy of the meta-behavior policy.

- We show that our task representation learning improves performance in in-distribution tasks and also enhances generalization to out-of-distribution tasks.
- We then show that ER-TRL's task representation learning outperforms prior methods as it can better predict the true task representation, *e.g.* the target velocity or direction in locomotion tasks.

2 Background

This section formally defines the context-based OMRL framework and subsequently explores the critical challenge of context shift, where the distribution of contexts during training and testing diverges, significantly impacting the performance of context-based OMRL algorithms.

Context-based Offline Meta-RL In OMRL, there is a distribution over tasks where each task is represented as a Markov Decision Process (MDP), $\mathcal{M}_i = \langle \mathcal{S}, \mathcal{A}, R_i, P_i, \gamma, \rho_0 \rangle$, consisting of a shared state space $\mathcal{S}$, action space $\mathcal{A}$, discount factor $\gamma \in [0, 1]$, and initial state distribution $\rho_0(s_0)$, and task-specific reward function $R_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and transition dynamics $P_i(s_{t+1}|s_t, a_t)$, which represents the distribution of possible next states conditioned on the current state and action. The objective is to train a meta-policy π that can generalize to new tasks, *i.e.*, maximizing the expected cumulative reward over the distribution of tasks

$$J(\pi) = \mathbb{E}_{\substack{\mathcal{M}_i \sim p(\mathcal{M}) \\ s_0 \sim \rho_0(s_0) \\ s_{t+1} \sim P_i(\cdot|s_t, a_t) \\ a_t \sim \pi(a_t|\cdot)}} \left[\sum_{t=0}^{T} \gamma^t R_i(s_t, a_t) \right]. \quad (1)$$

A common assumption for OMRL is that the reward function and the transition dynamics for all tasks are deterministic.

The offline dataset $\mathcal{D}_i = \{(s_j, a_j, r_j, s'_j)^i\}_{j=1}^{n_i}$ for task $\mathcal{M}_i$ consists of state, action, reward, next state tuples collected using its corresponding behavior policy $\pi_\beta^i(a|s)$. During training, context-based OMRL methods use a mini-batch of transitions from the offline dataset $c_i \subset \mathcal{D}_i$ – referred to as the context – to infer the task. A context encoder e_θ learns a mapping from the context c_i to latent task representations z_i. The policy and value functions are then conditioned on the task representations z_i and trained to maximize Eq. (1) using the offline datasets. Note that there is no online interaction.

Distribution Shift in Context Encoder During training, the context encoder is trained on offline datasets, collected by the behavior policy. The distribution over offline contexts $p(c_i)$ depends on the transition dynamics, reward function, and the behavior policy:

$$p(c_i) = p(s_1) \prod_{j=1}^{n} \pi_\beta^i(a_j|s_j) R_i(s_j, a_j) P_i(s_{j+1}|s_j, a_j).$$

However, at test time, the context is collected with an exploration policy π_{explore}, which may be *(i)* the learned policy conditioned on some prior task representations z_0, *(ii)* a random policy, or *(iii)* a combination of them. As such, the difference in the behavior and exploration policies leads to the distribution over the context being different at train and test time

$$p(c_i^{\text{test}}) = p(s_1) \prod_{j=1}^{n} \pi_{\text{explore}}(a_j|.) R_i(s_j, a_j) P_i(s_{j+1}|s_j, a_j).$$

Ideally, the context encoder should compress task-related information in the context and since the transition dynamics and reward function are different across tasks, the learned representation should only depend on them. However, as illustrated, the behavior policy affects the context distribution so the context encoder embeds some characteristics of the behavior policy specific to the task. In testing, the exploration policy's characteristics might differ, leading to inferring an irrelevant task representations and causing a reduction in performance.

3 Related Work

In this section, we provide an overview of context-based OMRL methods. We aim to highlight how different approaches attempt to tackle the distribution shift problem, which in turn motivates our approach.

Pre-collected Contexts Some methods (Li et al. 2020; Li, Yang, and Luo 2020; Yuan and Lu 2022) sidestep the issue of distribution shift in the context by collecting datasets from the test tasks using the behavior policy and using these to infer the task representations. Whilst these approaches sidestep the distribution shift problem, we highlight that this setting is unrealistic since it requires the agent to have access to all of the test tasks a priori, such that contexts for the test tasks can be collected with the behavior policy. In contrast, we consider the more realistic setting, where the agent does not have access to the test tasks a priori. It collects the context by online interaction with the environment.

Context Filtering IDAC (Wang et al. 2023) addresses the context shift problem by filtering transitions while collecting the context. They utilize uncertainty quantification, *e.g.* prediction error of an ensemble of dynamics models, to filter out-of-distribution trajectories in the context. Trajectories with higher uncertainty than a pre-defined threshold are disregarded in the context. Therefore, the collected context contains in-distribution transitions for inferring the task. However, this approach requires more interaction with the environment and is brittle to the choice of the threshold for uncertainty quantification. Moreover, in out-of-distribution tasks, the uncertainty for all the trajectories is high, leading to rejecting all the transitions.

Reconstruction UNICORN (Li et al. 2024) utilizes a decoder to predict the reward function and next state conditioned on the learned task representations, state, and action. Since the reward and transition dynamics are task-related components, the reconstruction objective encourages the context encoder to embed task-related characteristics.

Mutual Information The CSRO (Gao et al. 2024) method is most similar to ours as it implicitly minimizes an upper bound estimate of mutual information between the task representations and the behavior policy. CSRO computes the upper bound based on Cheng et al. (2020), by approximating the conditional distribution of the task representations given a behavior policy $p(z_i|s,a)$ as Gaussian. In contrast, we show that we can minimize the mutual information by maximizing the entropy of the behavior policy conditioned on the task representations. We use generative modeling to approximate this without making strong assumptions about the underlying distribution.

4 Method

In this section, we present our context-based OMRL algorithm, named *Entropy Regularized Task Representation Learning* (ER-TRL). We start by providing a general overview and then we detail our method for learning the task representation. In particular, we detail how we reduce the context distribution shift via our novel use of GANs.

Overview ER-TRL has five main components which we wish to learn:

$$\text{Context enc.:} \quad z_i = \mathbb{E}_{(s,a,r,s')\sim c_i}[e_\theta(s,a,r,s')] \tag{2}$$

$$\text{Critic:} \quad q = Q_\omega(s, z_i, a) \tag{3}$$

$$\text{Actor:} \quad a \sim \pi_\phi(s, z_i) \tag{4}$$

$$\text{Generator:} \quad a^{\text{fake}} = G_\psi(s, z_i, \epsilon) \tag{5}$$

$$\text{Discriminator:} \quad p(a) = D_\zeta(a, s, z_i) \tag{6}$$

For a particular task i, our encoder e_θ infers the task representations z_i from the context $c_i = \{(s,a,r,s')^i\}_{j=1}^H$, where H is the number of transitions in the context c_i. Note that the H transitions are not necessarily sequential. In contrast to previous methods, our task representation learning utilizes a GAN – consisting of a generator $G_\psi(s, z_i, \epsilon)$ and discriminator $D_\zeta(a, s, z_i)$ – to reduce context distribution shift, via approximately minimizing the mutual information. See Sec. 4.1, for more details. Once we have inferred the task representations z_i, ER-TRL follows standard OMRL methodology and conditions the actor $\pi_\phi(s, z_i)$ and critic $Q_\omega(s, z_i, a)$ on it.

We find that training GAN more often is beneficial since the context encoder is updated at each iteration, leading to changes in task representations z_i. We also bound the task representations z_i by using Tanh as the activation function of the context encoder.

4.1 Task Representation Learning

The encoder in Eq. (2) learns to map the context c_i to task representations z_i. We train our encoder to minimize

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{MI}}(\theta) + \lambda \mathcal{L}_{\text{DML}}(\theta), \tag{7}$$

where $\mathcal{L}_{\text{MI}}(\theta)$ (Eq. (14)) is a term to reduce the amount of context distribution shift, $\mathcal{L}_{\text{DML}}(\theta)$ (Eq. (15)) is a term to preserve distance in the embedding space through distance metric learning and λ is a hyper-parameter for balancing these objectives. We will now motivate each of these terms and detail how we calculate them in practice.

Algorithm 1: Meta-training

Input: Offline datasets $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^N$, context encoder e_θ, generator G_ψ, discriminator D_ζ, actor π_ϕ, critic Q_ω

while not done **do**
  Sample context $\{c_i\}_{i=1}^H$ and training data $\{(s_j, a_j, r_j, s'_j)^i\}_{j=1}^n$ from dataset $\mathcal{D}_i$.
  Obtain task representations z_i according to Eq. (2)
  for $k = 1, 2, \ldots,$ Num$_{\text{Update-GAN}}$ **do**
    $\epsilon \sim \mathcal{N}(0, \mathbb{I})$
    Generate actions a^{fake} according to Eq. (10).
    Update discriminator D_ζ and generator G_ψ according to Eq. (11) and Eq. (12).
  end for
  $\epsilon \sim \mathcal{N}(0, \mathbb{I})$
  Generate actions a^{fake} according to Eq. (10).
  Compute covariance of the a^{fake}.
  Compute $\mathcal{L}_{\text{MI}}(\theta)$ according to Eq. (14).
  Compute $\mathcal{L}_{\text{DML}}(\theta)$ according to Eq. (15).
  Update context encoder e_θ according to Eq. (7).
  Update the critic Q_ω according to Eq. (16).
  Update the actor π_ϕ according to Eq. (18).
end while
return context encoder e_θ and actor π_ϕ

Reducing Context Distribution Shift via Mutual Information Minimization We seek our context encoder to learn task representations z_i that is independent of the task-specific behavior policy π_β^i which collected the offline data. The mutual information between two random variables indicates the amount of information obtained from one random variable after observing the other. As such, minimizing the mutual information between our task representations z_i and the task-specific behavior policy π_β^i would achieve our goal of learning task representations that are independent of the task-specific behavior policy. However, calculating mutual information is not trivial as the task-specific behavior policy π_β^i is unknown.

We now detail our approach to minimizing the mutual information. Our approach builds upon the fact that we can rewrite the mutual information in terms of entropies and then utilize a GAN to approximate the objective. We first note that minimizing the mutual information between each task's representations and its associated behavior policy is equivalent to maximizing the conditional entropy

$$\min_\theta \frac{1}{N}\sum_{i=1}^N I(z_i, \pi_\beta^i) = \max_\theta \frac{1}{N}\sum_{i=1}^N H(\pi_\beta^i(a|s)|p(z_i)). \tag{8}$$

This is because the mutual information can be written in terms of entropies $I(z_i, \pi_\beta^i) = H(\pi_\beta^i) - H(\pi_\beta^i|p(z_i))$ and as the entropy of the behavior policy $H(\pi_\beta^i)$ is independent of the encoder's parameters θ, the behavior policy's entropy $H(\pi_\beta^i)$ is a constant.

Whilst we cannot compute $H(\pi_\beta^i(a|s)|p(z_i))$, we can utilize a GAN to estimate the meta-behavior policy $\hat{\pi}_\beta(a|s, z_i)$

by generating actions similar to those in the offline data. Importantly, this allows us to approximate the conditional entropy in Eq. (8) with the entropy of the meta-behavior policy

$$H(\pi^i_\beta(a|s)|p(z_i)) \approx H(\hat{\pi}_\beta(a|s, z_i)). \tag{9}$$

The GAN consists of a generator and discriminator, where the generator samples actions resembling the meta-behavior policy given an observation s, task representations z_i, and a noise vector ϵ

$$a^{\text{fake}} = G_\psi(s, z_i, \epsilon). \tag{10}$$

The discriminator $p\big(a \sim G_\psi(s, z_i, \epsilon)\big) = D_\zeta(a, s, z_i)$ then distinguishes between actions in the datasets and the generated actions, where $p\big(a\big)$ is the probability of action a being generated. To differentiate between the fake (generated) and real (from datasets) actions, the discriminator is trained according to

$$\mathcal{L}_D(\zeta) = -\log D_\zeta(a^{\text{real}}, s, z_i) - \log\left(1 - D_\zeta(a^{\text{fake}}, s, z_i)\right). \tag{11}$$

On the other hand, the generator is trained to fool the discriminator by generating fake actions similar to the actions in the dataset:

$$\mathcal{L}_G(\psi) = \log D_\zeta(G_\psi(s, z_i, \epsilon), s, z_i). \tag{12}$$

As training progresses, the distribution of the generated samples should better reflect the underlying meta-behavior policy. We generate samples from the meta-behavior policy to decrease context shift by maximizing the entropy based on the task representation. We assume a multivariate Gaussian distribution for samples and compute the covariance matrix of the generated samples $\Sigma_{j,k} = \mathbb{E}[(a^{\text{fake}}_j - \mathbb{E}[a^{\text{fake}}_j])(a^{\text{fake}}_k - \mathbb{E}[a^{\text{fake}}_k])]$ and determine the entropy accordingly:

$$H(\hat{\pi}_\beta(a|s, z_i)) \approx \frac{1}{2}\log\det\Sigma + \frac{k}{2}\log(2\pi e), \tag{13}$$

where k is the dimensionality of the action space. As stated, maximizing the entropy of samples is equivalent to minimizing the mutual information between task representations and behavior policy, leading to a decrease in context shift. This gives our context encoder's loss term

$$\mathcal{L}_{\text{MI}}(\theta) = -\frac{1}{2}\log\det\Sigma. \tag{14}$$

To avoid numerical instability, we add a small constant 10^{-5} to diagonal elements of the covariance matrix.

Distance Metric Learning Ideally, the context encoder should preserve distances when embedding, that is, it should embed contexts from the same task close together and contexts from different tasks further apart. This enables the agent to distinguish between different tasks and adapt accordingly. We use distance metric similar to FOCAL (Li, Yang, and Luo 2020) to achieve distinct task representations:

$$\mathcal{L}_{\text{DML}}(\theta) = \mathbf{1}\{i = j\}||z_i - z_j||_2^2 + \mathbf{1}\{i \neq j\}\frac{\beta}{||z_i - z_j||_2^2 + \epsilon_0}, \tag{15}$$

where ϵ_0 is a hyper-parameter added to avoid zero division. It has been shown that this objective approximately maximizes the mutual information between the task representations and the corresponding task (Gao et al. 2024; Li et al. 2024).

4.2 Meta-RL with Task Representations

Given the task representation learning strategy in Sec. 4.1, ER-TRL follows standard OMRL methodologies and conditions the agent on the task representations z_i so that it can adapt accordingly. We utilize the actor-critic framework to train the agent using temporal difference (TD) learning from offline datasets. Distribution shift (Levine et al. 2020) is a known problem in offline RL: Q values are overestimated because of a difference between the behavior policy and the policy being learned. We adopt BRAC (Wu, Tucker, and Nachum 2019) as the base offline RL algorithm which regularizes policy to alleviate issues arising from distribution shift when learning the actor and critic. The critic is trained to minimize

$$\mathcal{L}_Q(\omega; s, a, r, s', z_i) = \left(Q_\omega(s, a, z_i) - y\right)^2 \tag{16}$$

$$y = r - \gamma Q_{\bar{\omega}}(s', a', z_i), \tag{17}$$

where the task representations z_i is inferred by the encoder in Eq. (2), the next action is given by $a' \sim \pi_\phi(.|s', z_i)$ and the target y uses the exponential moving average (EMA) of the critic's weights, i.e. $\bar{\omega} = \tau\omega + (1 - \tau)\bar{\omega}$. The actor is then trained to minimize

$$\mathcal{L}_\pi(\phi; s, z_i) = Q_\omega(s, \tilde{a}, z_i) - \alpha D_{\text{KL}}\left(\pi_\phi(.|s, z_i)||\pi^i_\beta(.|s)\right), \tag{18}$$

where the action is a sample from the policy $\tilde{a} \sim \pi_\phi(a \mid s, z_i)$, D_{KL} is the estimate of the KL divergence between the learned policy π_ϕ and the behavior policy π_β, and α is a hyper-parameter for the amount of regularization. We used the dual form of KL divergence according to BRAC (Wu, Tucker, and Nachum 2019).

5 Experiments

In this section, we evaluate ER-TRL in a set of multi-task MuJoCo environments (Todorov, Erez, and Tassa 2012). Our experiments seek to answer the following questions:

1. Does ER-TRL's method for learning task representations improve adaptation performance and generalization?
2. Does our method for decoupling task representation learning from datasets capture the underlying task?
3. When does reducing the context shift improve the performance of OMRL agents?

Environments We now detail the environments and the dynamics parameters which we considered to be in-distribution and out-of-distribution. Note that we train on in-distribution parameters and evaluated on both in- and out-of-distribution parameters.

- **Cheetah-Vel**: a cheetah robot moves forward at a target velocity. The velocity for in-distribution tasks is between $[1, 2]$ and for out-of-distribution tasks is between $[0.5, 1]$ and $[2, 2.5]$.
- **Ant-Goal**: an ant robot moves to reach a goal state. The goals are in a semi-circle where the radius for in-distribution tasks is either 0.8 or 1.2 and for out-of-distribution tasks is 1.6, which is further.

ENVIRONMENT	CONTEXT	OFFLINEPEARL	FOCAL	CSRO	UNICORN	ER-TRL (Ours)
Cheetah-Vel	Offline	66.19 ± 14.41	69.33 ± 0.99	66.48 ± 2.66	66.44 ± 1.55	**72.61** ± 1.39
Ant-Goal		**82.16** ± 4.71	61.37 ± 5.56	52.05 ± 3.68	61.60 ± 3.37	79.14 ± 4.15
Ant-Dir		-18.87 ± 3.15	47.75 ± 3.31	51.51 ± 3.99	50.23 ± 2.34	**54.86** ± 2.82
Humanoid-Dir		**61.30** ± 6.94	49.13 ± 2.65	48.94 ± 2.82	44.82 ± 2.28	59.93 ± 2.79
Hopper-Mass		84.13 ± 15.47	80.56 ± 14.54	80.23 ± 7.53	83.88 ± 8.10	**99.72** ± 0.86
Hopper-Friction		51.61 ± 10.02	57.77 ± 7.18	63.12 ± 8.93	56.77 ± 7.55	**67.44** ± 6.71
Walker-Mass		36.97 ± 2.60	35.36 ± 5.27	43.21 ± 5.31	39.61 ± 8.82	**61.73** ± 2.15
Walker-Friction		60.15 ± 5.35	39.54 ± 6.48	49.37 ± 5.14	46.18 ± 6.84	**62.78** ± 10.89
Cheetah-Vel	Online	56.24 ± 10.60	54.94 ± 3.50	50.57 ± 11.06	51.61 ± 3.17	**61.85** ± 2.18
Ant-Goal		20.07 ± 4.98	29.23 ± 5.59	26.45 ± 3.57	26.31 ± 4.91	**49.15** ± 4.30
Ant-Dir		-19.91 ± 3.97	4.01 ± 3.60	8.52 ± 6.63	4.50 ± 7.95	**27.63** ± 5.46
Humanoid-Dir		**59.43** ± 7.00	44.37 ± 3.72	46.67 ± 5.82	36.65 ± 1.83	53.39 ± 3.33
Hopper-Mass		77.36 ± 12.14	81.61 ± 9.33	84.80 ± 10.06	83.36 ± 5.77	**99.59** ± 1.39
Hopper-Friction		48.04 ± 10.95	56.93 ± 12.62	60.95 ± 9.26	52.77 ± 16.28	**66.35** ± 8.73
Walker-Mass		34.48 ± 6.46	39.10 ± 4.82	39.70 ± 5.90	38.29 ± 7.51	**65.15** ± 4.48
Walker-Friction		41.73 ± 9.48	45.51 ± 5.78	49.47 ± 9.21	47.18 ± 7.21	**58.80** ± 9.94

Table 1: **Improved out-of-distribution generalization** The average normalized return for out-of-distribution test tasks after 100k training steps, averaged over 5 random seeds, $\pm$ represents standard deviation. **Bold** indicate highest mean value and **underline** indicate statistical significance according to t-test with p-value < 0.05.

- **Ant-Dir** and **Humanoid-Dir**: an ant/humanoid robot moves in a certain direction. The directions for in-distribution tasks is between $[-\frac{\pi}{2}, \frac{\pi}{2}]$ and for out-of-distribution tasks is between $[-\frac{3\pi}{4}, -\frac{\pi}{2}]$ and $[\frac{\pi}{2}, \frac{3\pi}{4}]$.
- **Hopper-Mass**, **Walker-mass**, **Hopper-Friction**, and **Walker-Friction**: a hopper (one-legged robot) or walker (bi-legged) robot must move as fast as it can while either the mass of the links or the friction varies in each task. The original mass and friction are scaled by a factor. For in-distribution tasks the factor is between $[1.5^{-1}, 1.5^{1}]$ and for out-of-distribution tasks the factor is between $[1.5^{-1.5}, 1.5^{-1}]$ and $[1.5^{1}, 1.5^{1.5}]$.

In the first 4 environments, the reward function is different for each task and the transition dynamics is different in the other 4 environments. To test the generalization of OMRL methods, we use different task distributions for training and testing and we divide the testing tasks into *in-distribution* and *out-of-distribution* tasks. In each environment, we consider 20 tasks for training, 10 tasks for in-distribution testing, and 10 tasks for out-of-distribution testing.

Baselines To evaluate the performance of our method, we compare it to the following baselines:

- **FOCAL** (Li, Yang, and Luo 2020) which utilizes distance metric objective to train the context encoder.
- **CSRO** which minimizes an upper bound of mutual information using CLUB (Cheng et al. 2020).
- **UNICORN** (Li et al. 2024) which utilizes a decoder to predict the next state, reward, and distance metric.
- **OfflinePEARL** which is an offline extension of PEARL (Rakelly et al. 2019) where the context encoder is trained based on the objective of the value function end-to-end.

We use the same network architecture, offline RL algorithm, and common hyper-parameters to compare fairly and focus on task representation learning.

Offline Datasets We used soft actor-critic (SAC, Haarnoja et al. 2018) to generate the datasets and trained each agent to an expert level. The dataset consists of trajectories collected from rolling out the corresponding SAC agent at different training stages; each dataset contains 180k transitions.

5.1 ER-TRL's Adaptation Performance

We evaluate the performance of our method alongside baselines on two scenarios. In the first (unrealistic) case, the context is given from the offline dataset collected from the behavior policy. In the second (realistic) case, agents gather the context by interacting with the environment conditioned on a prior task representations z_0. Then, the task representations is computed using the context encoder according Eq. (2) and we evaluate the agent conditioned on the inferred task representations.

Does ER-TRL's tasks representation learning improve performance in out-of-distribution tasks? Table 1 summarizes the results for out-of-distribution test tasks, where the distribution of tasks is different to training, allowing us to compare generalization. We normalized the return for each task such that a random agent yields a return of 0 and an expert SAC agent yields a return of 100. When the context is given, OfflinePEARL, which is trained end-to-end, has a competitive performance in most environments, even surpassing other methods in several tasks. However, when the context is collected by the agent, there is a sharp drop in performance for many environments due to the distribution shift. Our method (ER-TRL), CSRO, and UNICRON address this issue by reducing the correlation between the task representations and the policy that collected the context. However, our method outperformed the others in 7 out of 8 environments and there is a significant difference in 4 of them. This indicates that our approach to reducing the distribution shift is more effective.

ENVIRONMENT	MODEL	OFFLINEPEARL	FOCAL	CSRO	UNICORN	ER-TRL (Ours)
Cheetah-Vel	Linear Regression	0.3451 ± 0.0040	0.1720 ± 0.0014	$\underline{\mathbf{0.1586}} \pm 0.0009$	0.1804 ± 0.0013	0.1791 ± 0.0017
Ant-Goal		0.6477 ± 0.0029	0.5229 ± 0.0015	0.6235 ± 0.0030	0.4996 ± 0.0035	$\underline{\mathbf{0.4617}} \pm 0.0048$
Ant-Dir		1.6278 ± 0.5446	0.6144 ± 0.0066	0.4895 ± 0.0123	0.5313 ± 0.0129	$\underline{\mathbf{0.4546}} \pm 0.0171$
Humanoid-Dir		1.3596 ± 0.0000	0.8661 ± 0.0140	0.9876 ± 0.0081	0.8760 ± 0.0047	$\mathbf{0.8537} \pm 0.0010$
Hopper-Mass		0.3488 ± 0.0001	0.3218 ± 0.0012	0.2644 ± 0.0023	0.2383 ± 0.0030	$\underline{\mathbf{0.2265}} \pm 0.0026$
Hopper-Friction		0.3661 ± 0.0001	$\underline{\mathbf{0.2443}} \pm 0.0028$	0.2586 ± 0.0035	0.2518 ± 0.0015	0.2551 ± 0.0015
Walker-Mass		0.3533 ± 0.0012	0.3035 ± 0.0028	0.2895 ± 0.0020	$\underline{\mathbf{0.2712}} \pm 0.0006$	0.2794 ± 0.0026
Walker-Friction		0.3696 ± 0.0006	0.3715 ± 0.0011	0.3640 ± 0.0011	0.3679 ± 0.0014	$\underline{\mathbf{0.3526}} \pm 0.0015$
Cheetah-Vel	SVR	0.3643 ± 0.0018	0.1696 ± 0.0011	$\underline{\mathbf{0.1656}} \pm 0.0013$	0.1657 ± 0.0013	0.1689 ± 0.0009
Ant-Goal		0.6528 ± 0.0017	0.4998 ± 0.0021	0.5171 ± 0.0043	0.4850 ± 0.0025	$\underline{\mathbf{0.4182}} \pm 0.0022$
Ant-Dir		1.3561 ± 0.0006	0.4832 ± 0.0067	0.4753 ± 0.0082	0.4684 ± 0.0057	$\underline{\mathbf{0.4421}} \pm 0.0105$
Humanoid-Dir		1.3596 ± 0.0000	0.8242 ± 0.0058	0.8878 ± 0.0039	$\mathbf{0.8163} \pm 0.0060$	0.8186 ± 0.0051
Hopper-Mass		0.3453 ± 0.0010	0.3220 ± 0.0011	0.2486 ± 0.0016	0.2362 ± 0.0007	$\underline{\mathbf{0.2146}} \pm 0.0011$
Hopper-Friction		0.3649 ± 0.0011	0.2316 ± 0.0021	0.2443 ± 0.0015	$\underline{\mathbf{0.2285}} \pm 0.0012$	0.2328 ± 0.0017
Walker-Mass		0.3499 ± 0.0009	0.2812 ± 0.0006	0.2802 ± 0.0010	0.2656 ± 0.0011	$\underline{\mathbf{0.2515}} \pm 0.0007$
Walker-Friction		0.3713 ± 0.0004	0.3749 ± 0.0007	0.3618 ± 0.0004	0.3545 ± 0.0009	$\underline{\mathbf{0.3301}} \pm 0.0006$

Table 2: **Better task representation** The RMSE for predicting the true labels based on the task representations for test samples, **Bold** indicate lowest mean value and **underline** indicate statistical significance according to t-test with p-value < 0.05.

Does ER-TRL's tasks representation learning improve performance in in-distribution tasks? Table 3 summarizes the results for in-distribution tasks. Our method, ER-TRL, outperforms baselines in almost all experiments. For the case of online context collection, OfflinePEARL in Humanoid-Dir and CSRO in Ant-Dir perform slightly better than our method. In 3 environments, our method's performance significantly surpasses the others according to the t-test for both offline and online contexts.

5.2 Evaluating ER-TRL's Task Representation Learning

To evaluate the quality of the learned task representations in different methods, we trained simple regression models to predict the true goal label. For example, in the Ant-Dir environment, the goal label is the direction that the ant should move in the task. For each in-distribution and out-of-distribution test task, we sample 1000 transitions and embed them in task representations with the trained context encoder. We then use 80% of the samples for training and 20% for testing. Finally, we utilize linear regression and support vector regression (SVR, Awad et al. 2015) with a radial basis function kernel to compare the quality of learned task representations via root mean squared error (RMSE). Table 2 compares the quality of different task representations based on RMSE of goal predictions for test samples.

5.3 When Does Mutual Information Objective Improve the Performance?

The behavior policy affect the context during training and the context encoder may embed the characteristics of the behavior policy in the task representations. We addressed this issue by minimizing the mutual information between the task representations and the behavior policy. An interesting question is: when does this objective improve performance? Our intuition is that this objective improves the performance for environments where the behavior policies are different for each task. Therefore, the task representations should not embed the characteristics of the behavior policy. To evaluate our intuition, we compute the Wasserstein distance between the expert policies (SAC agents) used for data collection. These agents assumed a Gaussian distribution over the action given an observation. The Wasserstein distance between Gaussian distributions is defined as:

$$W_2(\mathcal{N}(\mu_1, \Sigma_1); \mathcal{N}(\mu_2, \Sigma_2)) = \sqrt{||\mu_1 - \mu_2||_2^2 + \mathrm{Tr}\big(\Sigma_1 + \Sigma_2 - 2(\Sigma_1^{1/2}\Sigma_2\Sigma_1^{1/2})^{\frac{1}{2}}\big)}. \quad (19)$$

We consider the average Wasserstein distance between the expert policies as an estimate of their difference

$$d = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} W_2(\pi_\beta^i; \pi_\beta^j)}{kn^2}, \quad (20)$$

where n is the number of tasks and k is dimensionality of action space. We divide the distance by the dimensionality of the action space since different environments have different action spaces and the Wasserstein distance depends on that. Fig. 1 illustrates the relationship between this distance and the improvement in performance when the mutual information objective (Eq. (14)) is considered for different environments. There is a correlation between the average Wasserstein distance and the improvement in performance for both in-distribution tasks and out-of-distribution tasks.

5.4 Latent Space Visualization

The main focus of context-based OMRL is on task representation learning. We visualize the task representations for 5 in-distribution tasks and 3 out-of-distribution test tasks. We use 256 samples for each task, compute the task representations for the same samples with different methods, and project the task representations into a two-dimensional space

ENVIRONMENT	CONTEXT	OFFLINEPEARL	FOCAL	CSRO	UNICORN	ER-TRL (Ours)
Cheetah-Vel	Offline	83.30 ± 20.31	92.84 ± 1.40	94.01 ± 0.74	93.53 ± 0.80	**96.36** ± 0.19
Ant-Goal		98.97 ± 3.69	93.90 ± 2.83	94.62 ± 3.67	95.53 ± 2.71	**107.66** ± 2.50
Ant-Dir		28.66 ± 2.99	60.75 ± 1.74	65.20 ± 4.66	62.29 ± 1.95	**69.08** ± 4.64
Humanoid-Dir		66.26 ± 4.19	62.14 ± 1.95	56.33 ± 1.36	58.02 ± 2.24	**70.53** ± 5.54
Hopper-Mass		89.45 ± 9.44	79.42 ± 10.97	83.74 ± 12.11	85.84 ± 9.63	**98.73** ± 0.32
Hopper-Friction		79.44 ± 11.90	80.85 ± 14.87	82.60 ± 9.98	80.55 ± 13.39	**86.65** ± 15.66
Walker-Mass		52.80 ± 2.30	54.09 ± 4.76	55.49 ± 1.28	56.68 ± 2.47	**77.52** ± 1.39
Walker-Friction		56.48 ± 4.78	47.76 ± 4.52	48.89 ± 6.18	51.05 ± 2.39	**64.22** ± 6.96
Cheetah-Vel	Online	80.04 ± 19.24	85.98 ± 1.79	89.35 ± 1.79	88.16 ± 1.67	**92.47** ± 0.36
Ant-Goal		49.07 ± 3.22	77.15 ± 3.17	80.86 ± 4.39	79.04 ± 2.11	**93.07** ± 1.95
Ant-Dir		29.09 ± 3.82	42.43 ± 5.18	**54.64** ± 3.89	41.25 ± 5.11	53.56 ± 11.56
Humanoid-Dir		**66.95** ± 5.71	54.40 ± 1.91	52.13 ± 3.08	44.90 ± 2.13	61.36 ± 2.42
Hopper-Mass		88.18 ± 8.82	86.47 ± 8.19	88.85 ± 3.65	84.38 ± 12.25	**99.66** ± 1.11
Hopper-Friction		65.92 ± 13.60	74.31 ± 30.60	81.91 ± 14.96	80.49 ± 14.00	**92.93** ± 6.38
Walker-Mass		36.84 ± 8.80	39.23 ± 4.42	46.31 ± 8.94	37.15 ± 10.26	**70.73** ± 2.32
Walker-Friction		36.52 ± 9.27	39.24 ± 5.51	46.43 ± 8.97	38.42 ± 6.72	**55.64** ± 11.06

Table 3: **Better in-distribution performance** The average normalized return for in-distribution tasks after 100k training steps, averaged over 5 random seeds, $\pm$ represents standard deviation. **Bold** indicate highest mean value and **underline** indicate statistical significance according to t-test with p-value < 0.05.

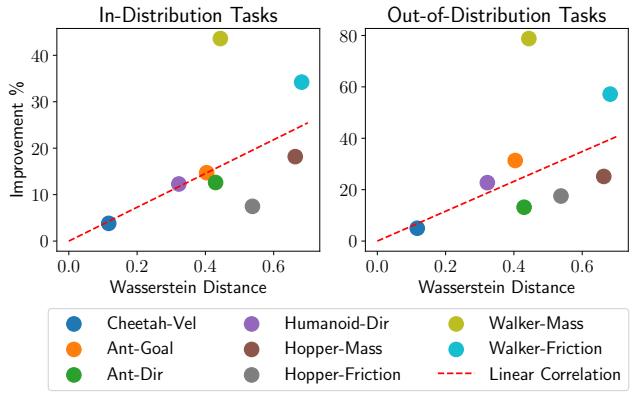

Figure 1: Correlation between the distance between expert policies and performance improvement by utilizing mutual information objective. For ID tasks, the Pearson coefficient is $\rho = 0.75$ with $p = 0.058$, and for OOD tasks $\rho = 0.69$ with $p = 0.086$.

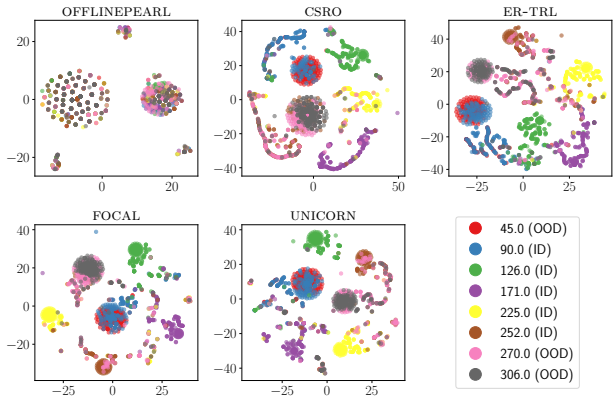

Figure 2: **Distinguishable task representation learning** Visualizing task representations with t-SNE projection for Ant-Dir environment. The labels illustrate the direction in which the ant robot should move in degrees.

using t-SNE (Van der Maaten and Hinton 2008). Fig. 2 illustrates the projection of the task representations for the Ant-Dir environment. OfflinePEARL fails to learn good representations and task representations for different tasks cannot be visually distinguished. Methods using contrastive objective learn separate task representations. ER-TRL learns distinguishable task representations while preserving the distance in the embedding space between different tasks.

6 Conclusion

This paper presents a novel approach to deal with context distribution shift in offline meta-RL, where the context encoder overfits to the dataset by embedding behavior-related characteristics. We propose to indirectly minimize the mutual information between the task representations and the behavior policy, we show that this is equivalent to maximizing the entropy of a meta-behavior policy. We utilized generative modeling, specifically a GAN, to generate actions resembling the meta-behavior policy and subsequently trained the context encoder to maximize the entropy of the generated actions. Experiments show that including this objective when training the context encoder captures the underlying tasks more accurately by learning task representations focused on embedding task-related characteristics. Subsequently, our entropy-regularized task representation learning outperformed previous context-based OMRL methods in both in-distribution and out-of-distribution tasks, improving online adaptation and generalization.

Acknowledgements

We acknowledge CSC – IT Center for Science, Finland, for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC (Finland) and the LUMI consortium through CSC. We acknowledge the computational resources provided by the Aalto Science-IT project. J. Pajarinen was partly supported by Research Council of Finland (345521). M. Nakhaei was supported by Business Finland (BIOND4.0 - Data Driven Control for Bioprocesses). A. Scannell was supported by the Research Council of Finland from the Flagship program: Finnish Center for Artificial Intelligence (FCAI).

References

Awad, M.; Khanna, R.; Awad, M.; and Khanna, R. 2015. Support vector regression. *Efficient learning machines: Theories, concepts, and applications for engineers and system designers*, 67–80.

Cheng, P.; Hao, W.; Dai, S.; Liu, J.; Gan, Z.; and Carin, L. 2020. Club: A contrastive log-ratio upper bound of mutual information. In *International conference on machine learning*, 1779–1788. PMLR.

Emerson, H.; Guy, M.; and McConville, R. 2023. Offline reinforcement learning for safer blood glucose control in people with type 1 diabetes. *Journal of Biomedical Informatics*, 142: 104376.

Gao, Y.; Zhang, R.; Guo, J.; Wu, F.; Yi, Q.; Peng, S.; Lan, S.; Chen, R.; Du, Z.; Hu, X.; et al. 2024. Context shift reduction for offline meta-reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. PMLR.

Levine, S.; Kumar, A.; Tucker, G.; and Fu, J. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. arXiv:2005.01643.

Li, J.; Vuong, Q.; Liu, S.; Liu, M.; Ciosek, K.; Christensen, H.; and Su, H. 2020. Multi-task batch reinforcement learning with metric learning. *Advances in neural information processing systems*, 33: 6197–6210.

Li, L.; Yang, R.; and Luo, D. 2020. Focal: Efficient fully-offline meta-reinforcement learning via distance metric learning and behavior regularization. *arXiv preprint arXiv:2010.01112*.

Li, L.; Zhang, H.; Zhang, X.; Zhu, S.; Zhao, J.; and Heng, P.-A. 2024. Towards an Information Theoretic Framework of Context-Based Offline Meta-Reinforcement Learning. arXiv:2402.02429.

Luo, J.; Dong, P.; Wu, J.; Kumar, A.; Geng, X.; and Levine, S. 2023. Action-quantized offline reinforcement learning for robotic skill learning. In *Conference on Robot Learning*, 1348–1361. PMLR.

Prudencio, R. F.; Maximo, M. R. O. A.; and Colombini, E. L. 2023. A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems. *IEEE Transactions on Neural Networks and Learning Systems*, 1–0.

Rakelly, K.; Zhou, A.; Finn, C.; Levine, S.; and Quillen, D. 2019. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, 5331–5340. PMLR.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Wang, J.; Zhang, J.; Jiang, H.; Zhang, J.; Wang, L.; and Zhang, C. 2023. Offline meta reinforcement learning with in-distribution online adaptation. In *International Conference on Machine Learning*, 36626–36669. PMLR.

Wu, Y.; Tucker, G.; and Nachum, O. 2019. Behavior Regularized Offline Reinforcement Learning. arXiv:1911.11361.

Yuan, H.; and Lu, Z. 2022. Robust task representations for offline meta-reinforcement learning via contrastive learning. In *International Conference on Machine Learning*, 25747–25759. PMLR.

Zhao, C.; Zhou, Z.; and Liu, B. 2023. On context distribution shift in task representation learning for offline meta RL. In *International Conference on Intelligent Computing*, 614–628. Springer.

Zhou, G.; Ke, L.; Srinivasa, S.; Gupta, A.; Rajeswaran, A.; and Kumar, V. 2023. Real world offline reinforcement learning with realistic data source. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 7176–7183. IEEE.

Zhu, T.; Li, K.; and Georgiou, P. 2023. Offline deep reinforcement learning and off-policy evaluation for personalized basal insulin control in type 1 diabetes. *IEEE Journal of Biomedical and Health Informatics*.