Anti-Exploration by Random Network Distillation

Alexander Nikulin [1] **Vladislav Kurenkov** [1] **Denis Tarasov** [1] **Sergey Kolesnikov** [1]

Abstract

Despite the success of Random Network Distillation (RND) in various domains, it was shown as not discriminative enough to be used as an uncertainty estimator for penalizing out-of-distribution actions in offline reinforcement learning. In this paper, we revisit these results and show that, with a naive choice of conditioning for the RND prior, it becomes infeasible for the actor to effectively minimize the anti-exploration bonus and discriminativity is not an issue. We show that this limitation can be avoided with conditioning based on Feature-wise Linear Modulation (FiLM), resulting in a simple and efficient ensemble-free algorithm based on Soft Actor-Critic. We evaluate it on the D4RL benchmark, showing that it is capable of achieving performance comparable to ensemble-based methods and outperforming ensemble-free approaches by a wide margin.

1. Introduction

In recent years, significant success has been achieved in applying Reinforcement Learning (RL) to challenging and large-scale tasks such as Atari (Badia et al., 2020), Go (Schrittwieser et al., 2020), Dota 2 (Berner et al., 2019), and Minecraft (Baker et al., 2022). However, the online nature of such RL algorithms makes it difficult to apply them in the real world, where online collection of large amounts of exploratory data may not be feasible for safety or financial reasons. Offline Reinforcement Learning (Levine et al., 2020) promises a more controllable and data-driven approach, focusing on algorithms that can learn from a fixed, pre-recorded dataset without requiring additional environment interactions.

The use of ensembles for uncertainty-based penalization has proven to be one of the most effective approaches for offline RL. Ensemble-based algorithms, such as SAC-N, EDAC

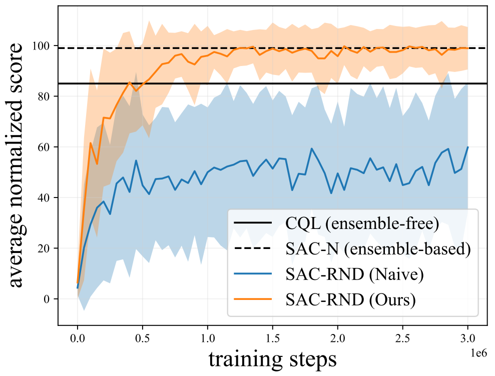


Figure 1. Mean performance of SAC-RND variants on walker and hopper medium-* datasets, each averaged over 3 seeds. We plot performance for the naive version, which uses concatenation conditioning, and our final version, which is described in Section 5. We also plot the final scores for the ensemble-free CQL (Kumar et al., 2020) and the ensemble-based SAC-N (An et al., 2021). It can be seen that our version is a significant improvement over the naive version, achieving performance comparable to ensembles.

(An et al., 2021), and MSG (Ghasemipour et al., 2022) currently achieve state-of-the-art results on most D4RL (Fu et al., 2020) datasets, outperforming ensemble-free methods by a wide margin. Unfortunately, in order to achieve the best performance, these algorithms may require tens or hundreds of ensemble members, leading to significant computational and memory overhead, as well as extended training duration (Nikulin et al., 2022).

Recent research (Yang et al., 2022) has successfully reduced the ensemble size to tens of Q-networks in the worst-case scenarios. However, given the general trend for model scaling in offline RL (Kumar et al., 2022; Reed et al., 2022; Lee et al., 2022), efficiently training even ten Q-networks with 80 million parameters each is not feasible. Furthermore, Ghasemipour et al. (2022) showed that methods for efficient ensemble training found in supervised learning literature do not deliver performance comparable to naive ensembles and can even worsen the results. Thus, further research on efficient uncertainty estimation for offline RL is needed, with the goal of reducing the size of the ensemble as much as possible or even fully removing it.

In this work, we move away from ensembles and take an

[1]Tinkoff, Moscow, Russia. Correspondence to: Alexander Nikulin <a.p.nikulin@tinkoff.ai>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

alternative approach to uncertainty estimation, proposing an efficient offline RL method with ensemble-free uncertainty estimation via Random Network Distillation (RND) (Burda et al., 2018). RND, a simple and fast ensemble competitor for epistemic uncertainty estimation (Ciosek et al., 2019), is an attractive choice for offline RL. However, previous research (Rezaeifar et al., 2022) found RND to be insufficiently discriminative for good results.

In our preliminary experiment (Section 3), we show that RND is discriminative enough to detect OOD actions, which contradicts the previous study (Rezaeifar et al., 2022). Nevertheless, our results show that the naive application of RND does indeed not lead to good results (see Figure 1). Building upon these findings, we further simplify the problem and analyze the reasons for this issue (Section 4). We discover that a naive choice of conditioning for the RND prior can hinder the minimization of the anti-exploration bonus by the actor, and that conditioning based on Feature-wise Linear Modulation (FiLM) (Perez et al., 2018) is particularly effective in solving this problem.

Based on our findings, we propose a new ensemble-free offline RL algorithm called **SAC-RND** (Section 5). We evaluate our method on the D4RL (Fu et al., 2020) benchmark (Section 6), and show that SAC-RND achieves performance comparable to ensemble-based methods while outperforming ensemble-free approaches.

2. Background

Offline Reinforcement Learning. Reinforcement learning problem can be described as a Markov Decision Process (MDP) defined by the $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$ tuple with state space $\mathcal{S} \subset \mathbb{R}^N$, action space $\mathcal{A} \subset \mathbb{R}^M$, transition dynamics $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and a discount factor γ. The goal of reinforcement learning in an infinite horizon setting is to produce a policy $\pi(a|s)$ that maximizes the expected cumulative discounted return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$.

In offline reinforcement learning, a policy must be learned from a fixed dataset $\mathcal{D}$ collected under a different policy or mixture of policies, without any environment interaction. This setting poses unique fundamental challenges (Levine et al., 2020), since the learning policy is unable to explore and has to deal with distributional shift and extrapolation errors (Fujimoto et al., 2019) for actions not represented in the training dataset.

Offline RL as Anti-Exploration. There are numerous approaches for offline RL, a substantial part of which constrain the learned policy to stay within the support of the training dataset, thus reducing (Kumar et al., 2020) or avoiding (Kostrikov et al., 2021) extrapolation errors. For our work, it is essential to understand how such a constraint can be framed as *anti-exploration* (Rezaeifar et al., 2022).

Similarly to online RL, where novelty bonuses are used as additive intrinsic rewards for effective exploration, in offline RL, novelty bonuses can induce conservatism, reducing the reward in unseen state-action pairs. Hence the name *anti-exploration*, since the same approaches from exploration can be used, but a bonus is subtracted from the extrinsic reward instead of being added to it.

However, unlike online RL, subtracting a bonus from the raw reward would not be as useful, since the novelty bonus is, by design, close to zero for in-dataset state-action pairs. Therefore, it is more effective to apply it where the overestimation for OOD actions emerges — the temporal difference learning target:

$$r + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q(s', a') - b(s', a')] \tag{1}$$

where the actor is trained to maximize the expected Q-value, as is usually done in off-policy actor-critic algorithms (Lillicrap et al., 2015; Haarnoja et al., 2018). It can be shown that, theoretically, these approaches are equivalent, but the latter is more suited for use in offline RL (Rezaeifar et al., 2022).

An illustrative example of how such framing can be effective are ensemble-based approaches such as SAC-N & EDAC (An et al., 2021) and MSG (Ghasemipour et al., 2022), which currently outperform their ensemble-free counterparts by a large margin on most D4RL (Fu et al., 2020) benchmark datasets. For the anti-exploration bonus, these methods use ensemble disagreement as a proxy for epistemic uncertainty. However, a large number of ensemble members is usually required for a competitive result.

Random Network Distillation. Random network distillation (RND) was first proposed in online RL (Burda et al., 2018) as a simple and effective exploration bonus. To this day, RND is still considered a strong baseline for exploration that can work well even in stochastic environments, contrary to some more modern approaches (Jarrett et al., 2022).

RND consists of two neural networks: a fixed and randomly initialized *prior* network $\bar{f}_{\bar{\psi}}$, and a *predictor* network f_ψ which learns to predict the prior outputs on the training data:

$$\|f_\psi(s) - \bar{f}_{\bar{\psi}}(s)\|_2^2 \tag{2}$$

Both networks map states to embeddings in $\mathbb{R}^K$, and the gradient through *prior* is disabled. The interpretation of the novelty is straightforward: with the sufficiently diverse *prior*, the *predictor* must learn to match embeddings on data points similar to the training dataset, while failing to predict on new examples. A bonus in such a case may simply be a prediction error, as in Equation (2).

In a subsequent work, Ciosek et al. (2019) analyses the success of RND in a supervised setting, and shows that fitting random priors can be a competitive alternative to ensembles for estimating epistemic uncertainty.

Note that in practice, the choice of predictor and prior having the same architecture and the estimation of novelty from states only are *very common, but arbitrary*. Moreover, for offline RL, we are interested in estimating the novelty of an action conditioned on the state, which is why in our work RND depends on both: $f_\psi(s, a)$.

Multiplicative Interactions. The most common way to fuse two different streams of information is feature concatenation, which is straightforward but can be suboptimal (Dumoulin et al., 2018). Jayakumar et al. (2020) shows that multiplicative interactions provide a powerful inductive bias for fusing or conditioning from multiple streams and are superior in practice. We provide a brief review of those used in our work (excluding concatenation): gating, bilinear, and feature-wise linear modulation (FiLM).

Gating. Simple conditioning with two linear layers and pointwise multiplication of the resulting features (Srivastava et al., 2019).

$$f(a, s) = tanh(W_1 a + b_1) \odot \sigma(W_2 s + b_2)$$

Bilinear. Bilinear layer in its most general form, as proposed by Jayakumar et al. (2020).

$$f(a, s) = s^T \mathbb{W} a + s^T \mathbb{U} + \mathbb{V} a + b$$

where $\mathbb{W}$ is a 3D tensor, $\mathbb{U}$, $\mathbb{V}$ are regular matrices and b is a vector. However, in our work, we also use the implementation as in PyTorch (Paszke et al., 2019), which does not learn $\mathbb{U}$, $\mathbb{V}$ by default.

FiLM. Special case of a bilinear layer with low-rank weight matrices (Perez et al., 2018).

$$f(h, s) = \gamma(s) \odot h + \beta(s)$$

Usually, FiLM operates on hidden activations h before non-linearity between layers. Thus, the main network takes a as an input.

3. Random Network Distillation is Discriminative Enough

To better understand the possible difficulties of applying RND to offline RL, we first reproduce the main experiment from Rezaeifar et al. (2022), which showed that RND is not discriminative enough to be used as a novelty bonus. For convenience, we provide the original figure from Rezaeifar et al. (2022) in the Appendix B. We also compare RND with a trained Q-ensemble (N = 25) from the SAC-N algorithm (An et al., 2021). Similarly to Rezaeifar et al. (2022), we use simple state-action concatenation. Predictor and prior share the identical architecture of 4-layer MLPs.

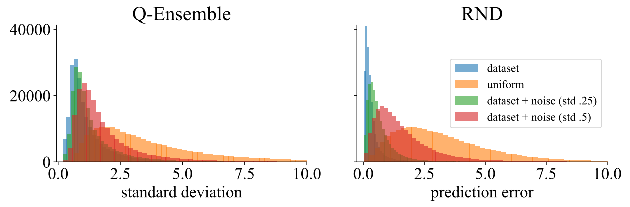


Figure 2. Anti-exploration bonus (Rezaeifar et al., 2022) on the walker2d-medium dataset for trained SAC-N (An et al., 2021), Q-ensemble (N = 25) and RND. Bonus is computed for state-action pairs from the original dataset and different perturbations of actions: random actions, dataset actions to which Gaussian noise is added with different scales. Both RND networks use simple state-action concatenation. The result is strikingly different from a similar figure in the Rezaeifar et al. (2022) (we provide the original figure in the Appendix B for convenience). Contrary to previous research, it can be seen that RND is capable of distinguishing ID from OOD actions and is comparable to a trained Q-ensemble.

The goal of the experiment (see Figure 2) is to visually plot the anti-exploration bonus for ID state-action pairs and different perturbations of actions to model OOD data: random actions sampled from a uniform distribution and dataset actions to which Gaussian noise with different scales is added.

To our surprise, the result on Figure 2 is strikingly different from previous work. It shows that RND is able to discriminate between ID and OOD actions with varying degrees of distributional shift and is comparable to a trained Q-ensemble. In contrast, Rezaeifar et al. (2022) hypothesizes that RND can only work well out of the box for discrete action spaces and visual features, and concludes that extending it to continuous action spaces is not straightforward.

After further investigation of the open-sourced codebase[1] in search of discrepancies with our implementation, we found that the only difference is that, contrary to the advice of Ciosek et al. (2019), Rezaeifar et al. (2022) sets the predictor smaller than prior by two layers during RND pretraining. It is important to make the predictor larger or comparable in capacity to the prior so that it can minimize the loss to zero on the training dataset (Ciosek et al., 2019). However, the actual RND hyperparameters used in the final publication were not listed, so we cannot draw a definitive conclusion about the reason for such different results.

[1]https://github.com/shidilrzf/Anti-exploration-RL

4. Concatenation Prior Hinders Bonus Minimization

A well-behaved anti-exploration bonus for continuous action spaces, be it RND or any other, should satisfy at least two criteria. First, it should be discriminative enough to detect novel actions and downweight their value estimates (see Equation (1)). Ideally, the bonus should be close to zero for ID data so that we do not bias the Q-function, as this can be detrimental to training. Second, it should allow the actor to easily minimize the bonus with gradient descent during training.

In Section 3, we showed that RND can detect OOD actions. Nevertheless, naive use of RND as an anti-exploration bonus on top of the Soft Actor Critic algorithm (Haarnoja et al., 2018) still does not provide satisfactory performance (see Figure 1) with scores lower than CQL (Kumar et al., 2020) and SAC-N (An et al., 2021). This gives us an hint that the problem may not be the discriminative power of RND, but that the actor cannot effectively minimize the anti-exploration bonus during training.

To test our hypothesis that the actor cannot effectively minimize the anti-exploration bonus, we further simplify the problem by removing the critic from the SAC algorithm but keeping the entropy bonus (see Algorithm 2 in the Appendix). We expect that, in such a setting, the actor will be able to successfully minimize the anti-exploration bonus to the possible minimum, i.e. comparable to the bonus for the ground truth data at the end of the RND pretraining. As a consequence, since dataset actions provide the minimum bonus by design, we also expect that the distance from the agent to dataset actions should be small.

We set predictor architecture to state-action concatenation. Additionally, we explore different conditioning schemes for the prior. We use the halfcheetah, walker2d and hopper medium datasets, with 3 seeds each. Figure 3 compares the anti-exploration bonus for dataset actions during RND pretraining (see Figure 3a) and for agent actions during training (see Figure 3b).

As one can see for all prior architectures except one, the anti-exploration bonus during actor training is much higher than it should be according to the values on the dataset actions. These results confirm our hypothesis. Furthermore, we can note from Figure 3c that the actor cannot clone the behavioral policy, since the distance to the dataset actions can even increase during training.

However, RND with the FiLM prior architecture allows the actor to effectively minimize the anti-exploration bonus and successfully clone the behavioral policy. This suggests that, with the right inductive bias for the prior, we can solve the problems of naive RND and possibly achieve better results.

Table 1. Comparison of different RND predictors. Prior uses FiLM conditioning. Predictor uses conditioning in the first layer. All scores are averaged over 3 random seeds. Halfcheetah tasks are omitted, as we found them non-representative of the final performance on harder tasks.

Task Name	Concat	Gating	Bilinear	FiLM
hopper-medium-v2	94.8	39.7	98.4	86.3
hopper-medium-expert-v2	71.5	59.3	110.3	102.7
hopper-medium-replay-v2	100.3	51.3	100.8	100.3
walker2d-medium-v2	94.8	82.3	92.8	95.1
walker2d-medium-expert-v2	86.1	84.2	108.9	110.0
walker2d-medium-replay-v2	90.3	87.5	88.3	75.7
Average	89.6	67.3	**99.9**	95.0

5. Anti-Exploration by Random Network Distillation

We are now ready to present **SAC-RND**: a new offline RL method for continuous action spaces, based on our findings in Section 3 and Section 4. It is simple, ensemble-free and achieves state-of-the-art results comparable to ensemble-based methods. We have chosen the Soft Actor-Critic (Haarnoja et al., 2018) algorithm as the backbone of the method. In this section, we will explain how the RND is trained and how we define the anti-exploration bonus.

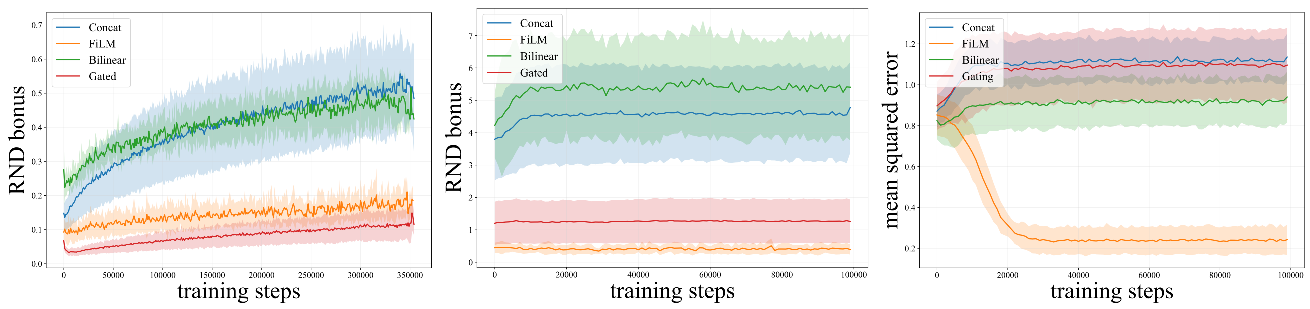
Random Network Distillation. We pretrain RND with MSE loss between prior and predictor embeddings, stopping gradient through prior and freezing both networks afterwards during SAC training. We keep both networks similar in size to the agent and critic, which are 4 layer MLPs. Contrary to Burda et al. (2018); Ciosek et al. (2019), we do not add additional layers to the predictor to prevent undesirable results. This is because, when the predictor size is bigger than prior on state-based tasks (not image-based as in original work by Burda et al. (2018)), we observe that it can sometimes overgeneralize to OOD prior embeddings.

According to Section 4, for the prior, we use FiLM conditioning on penultimate layer before nonlinearity. In principle, the predictor can be arbitrary (Ciosek et al., 2019), but in practice, its architecture and conditioning type can also affect performance. We conduct a preliminary study on a small subset of the D4RL Gym tasks to select the best-performing conditioning. Based on the results in Table 1, we chose a predictor with bilinear conditioning in the first layer, as it showed the best performance.

Anti-Exploration Bonus. We define the anti-exploration bonus similarly to RND loss as

$$b(s,a) = \|f_{\psi}(s,a) - \bar{f}_{\bar{\psi}}(s,a)\|_2^2 \tag{3}$$

and additionally divide it by RND loss running standard deviation (which is tracked during pretraining phase) to increase its scale uniformly among environments. Such



(a) RND bonus for dataset actions (b) RND bonus for actor actions (c) Distance to dataset actions

Figure 3. Effect of different state-action conditioning in the prior of RND on actor training. We use the halfcheetah, walker2d and hopper medium datasets, with 3 seeds each. For training procedure, see Algorithm 2 in the Appendix. **(a)** Anti-exploration bonus for in-dataset actions during RND pretraining. We additionally divide the bonus by the RND loss running standard deviation to increase its scale (see Section 5) so the anti-exploration bonus increases slightly over time as standard deviation decreases. However, this does not affect minimization by the actor and is needed to highlight the differences. **(b)** Anti-exploration bonus for actor actions during training. Ideally, it should converge to values close to the final values in (a). **(c)** Distance of actor actions to true in-dataset actions during training. Ideally, it should decrease, as actions closer to the behavioral policy have the lowest bonus by design.

scaling simplifies hyperparameter search, shrinking the possible range of useful α coefficients that control the level of conservatism during training.

For detailed training procedure and full SAC losses, we refer to Algorithm 1 in the Appendix (differences with the original SAC algorithm are highlighted in blue).

6. Experiments

In this section, we present an empirical evaluation of our method using the D4RL benchmark on the Gym domain (Section 6.1) and the more challenging AntMaze domain (Section 6.2). Next, we provide additional analysis and visual insight into why FiLM conditioning in the prior might be beneficial (Section 6.3). Finally, we present an ablation that compares more variations of conditioning for predictor and prior (Section 6.4). For each experiment, we also list the exact hyperparameters in Appendix E and implementation details in Appendix D. Additionally, we analyse sensitivity to hyperparameters in Appendix G.

6.1. Evaluation on the Gym Domain

Setup. We evaluate our method on all available datasets for the HalfCheetah, Walker2d and Hopper tasks in the Gym domain of the D4RL benchmark. For ensemble-free baselines, we chose CQL (Kumar et al., 2020), IQL (Kostrikov et al., 2021), TD3+BC (Fujimoto & Gu, 2021), which show good results and are widely used in practice. We also report scores for vanilla SAC (Haarnoja et al., 2018). For ensemble-based baselines, we chose SAC-N & EDAC (An et al., 2021) and the more recent RORL (Yang et al., 2022), which currently achieve state-of-the-art scores in this domain. We follow the An et al. (2021) and train for 3M gradient steps, evaluating on 10 episodes.

Results. The resulting scores are presented in Table 2. We see that SAC-RND stands out from the ensemble-free methods and outperforms them by a wide margin, achieving a mean score comparable to EDAC and only slightly behind RORL. Note that we do not use ensembles, whereas SAC-N can require up to 500 critics, EDAC up to 50 and RORL up to 20. In addition, we compare our proposed changes with the naive predictor and prior, confirming that our modifications are essential for achieving good performance (see Figure 1).

6.2. Evaluation on the AntMaze Domain

Setup. We evaluate our method on all datasets available for the AntMaze domain of the D4RL benchmark. Ensemble-free baselines are the same as in Section 6.1. For ensemble-based baselines, we chose RORL (Yang et al., 2022) and MSG (Ghasemipour et al., 2022), the latter of which, to our knowledge, currently has the best mean score for this domain. We do not include SAC-N and EDAC, as there are no public results for them on this domain, and we were also unable to obtain a non-zero result. We follow the An et al. (2021) and train for 3M gradient steps, evaluating on 100 episodes.

Results. The resulting scores are presented in Table 3. Kostrikov et al. (2021) has shown that many offline RL methods that perform well on the Gym domain fail on the AntMaze domain. It can be seen that, on the AntMaze domain, SAC-RND shows good results that are on par with ensembles, and outperforms ensemble-free methods. This also shows that our choice of predictor and prior generalises well to new domains. Note that, in addition to ensembles,

Table 2. SAC-RND evaluation on the Gym domain. We report the final normalized score averaged over 4 random seeds on v2 datasets. TD3 + BC and IQL scores are taken from Lyu et al. (2022). CQL, SAC, SAC-N and EDAC scores are taken from An et al. (2021). RORL scores are taken from Yang et al. (2022).

	Ensemble-free				Ensemble-based			
Task Name	**SAC**	**TD3+BC**	**IQL**	**CQL**	**SAC-N**	**EDAC**	**RORL**	**SAC-RND**
halfcheetah-random	29.7 ± 1.4	11.0 ± 1.1	13.1 ± 1.3	31.1 ± 3.5	28.0 ± 0.9	28.4 ± 1.0	28.5 ± 0.8	29.0 ± 1.5
halfcheetah-medium	55.2 ± 27.8	48.3 ± 0.3	47.4 ± 0.2	46.9 ± 0.4	67.5 ± 1.2	65.9 ± 0.6	66.8 ± 0.7	66.6 ± 1.6
halfcheetah-expert	-0.8 ± 1.8	96.7 ± 1.1	95.0 ± 0.5	97.3 ± 1.1	105.2 ± 2.6	106.8 ± 3.4	105.2 ± 0.7	105.8 ± 1.9
halfcheetah-medium-expert	28.4 ± 19.4	90.7 ± 4.3	86.7 ± 5.3	95.0 ± 1.4	107.1 ± 2.0	106.3 ± 1.9	107.8 ± 1.1	107.6 ± 2.8
halfcheetah-medium-replay	0.8 ± 1.0	44.6 ± 0.5	44.2 ± 1.2	45.3 ± 0.3	63.9 ± 0.8	61.3 ± 1.9	61.9 ± 1.5	54.9 ± 0.6
halfcheetah-full-replay	86.8 ± 1.0	-	-	76.9 ± 0.9	84.5 ± 1.2	84.6 ± 0.9	-	82.7 ± 0.9
hopper-random	9.9 ± 1.5	8.5 ± 0.6	7.9 ± 0.2	5.3 ± 0.6	31.3 ± 0.0	25.3 ± 10.4	31.4 ± 0.1	31.3 ± 0.1
hopper-medium	0.8 ± 0.0	59.3 ± 4.2	66.2 ± 5.7	61.9 ± 6.4	100.3 ± 0.3	101.6 ± 0.6	104.8 ± 0.1	97.8 ± 2.3
hopper-expert	0.7 ± 0.0	107.8 ± 7.0	109.4 ± 0.5	106.5 ± 9.1	110.3 ± 0.3	110.1 ± 0.1	112.8 ± 0.2	109.7 ± 0.3
hopper-medium-expert	0.7 ± 0.0	98.0 ± 9.4	91.5 ± 14.3	96.9 ± 15.1	110.1 ± 0.3	110.7 ± 0.1	112.7 ± 0.2	109.8 ± 0.6
hopper-medium-replay	7.4 ± 0.5	60.9 ± 18.8	94.7 ± 8.6	86.3 ± 7.3	101.8 ± 0.5	101.0 ± 0.5	102.8 ± 0.5	100.5 ± 1.0
hopper-full-replay	41.1 ± 17.9	-	-	101.9 ± 0.6	102.9 ± 0.3	105.4 ± 0.7	-	107.3 ± 0.1
walker2d-random	0.9 ± 0.8	1.6 ± 1.7	5.4 ± 1.2	5.1 ± 1.7	21.7 ± 0.0	16.6 ± 7.0	21.4 ± 0.2	21.5 ± 0.1
walker2d-medium	-0.3 ± 0.2	83.7 ± 2.1	78.3 ± 8.7	79.5 ± 3.2	87.9 ± 0.2	92.5 ± 0.8	102.4 ± 1.4	91.6 ± 2.8
walker2d-expert	0.7 ± 0.3	110.2 ± 0.3	109.9 ± 1.2	109.3 ± 0.1	107.4 ± 2.4	115.1 ± 1.9	115.4 ± 0.5	114.3 ± 0.6
walker2d-medium-expert	1.9 ± 3.9	110.1 ± 0.5	109.6 ± 1.0	109.1 ± 0.2	116.7 ± 0.4	114.7 ± 0.9	121.2 ± 1.5	105.0 ± 7.9
walker2d-medium-replay	-0.4 ± 0.3	81.8 ± 5.5	73.8 ± 7.1	76.8 ± 10.0	78.7 ± 0.7	87.1 ± 2.4	90.4 ± 0.5	88.7 ± 7.7
walker2d-full-replay	27.9 ± 47.3	-	-	94.2 ± 1.9	94.6 ± 0.5	99.8 ± 0.7	-	109.2 ± 1.8
Average	16.2	67.5	68.9	73.6	84.4	**85.2**	**85.7**	**85.2**

both MSG and RORL require pre-training or supervision with behavioural cloning in order to achieve reported results, while our method does not require any additional modifications.

Table 3. SAC-RND evaluation on AntMaze domain. We report the final normalized score averaged over 4 random seeds on v1 datasets. IQL, CQL, MSG scores are taken from Ghasemipour et al. (2022) and SAC from (Kumar et al., 2020). TD3+BC, RORL scores are taken from Yang et al. (2022).

	Ensemble-free				Ensemble-based		
Task Name	**SAC**	**TD3+BC**	**IQL**	**CQL**	**RORL**	**MSG**	**SAC-RND**
antmaze-umaze	0.0	78.6	87.5	74.0	97.7 ± 1.9	97.8 ± 1.2	97.2 ± 1.2
antmaze-umaze-diverse	0.0	71.4	62.2	84.0	90.7 ± 2.9	81.8 ± 3.0	83.5 ± 7.7
antmaze-medium-play	0.0	10.6	71.2	61.2	76.3 ± 2.5	89.6 ± 2.2	65.5 ± 35.7
antmaze-medium-diverse	0.0	3.0	70.0	53.7	69.3 ± 3.3	88.6 ± 2.6	88.5 ± 9.2
antmaze-large-play	0.0	0.2	39.6	15.8	16.3 ± 11.1	72.6 ± 7.0	67.2 ± 6.1
antmaze-large-diverse	0.0	0.0	47.5	14.9	41.0 ± 10.7	71.4 ± 12.2	57.6 ± 22.7
Average	0.0	27.3	63.0	50.6	65.2	**83.6**	**76.6**

6.3. Why is FiLM Conditioning Beneficial for Bonus Minimization?

In Section 4, we showed that FiLM conditioning in the RND prior significantly improved the actors' ability to minimize the anti-exploration bonus. Since the issue occurred during actor training, we hypothesize that this may be related to the anti-exploration bonus optimization landscape. In this section, we analyze the anti-gradient fields for conditioning with concatenation or FiLM for the prior network.

For the purpose of analysis, we design a toy dataset with only four categorical states and two-dimensional actions sampled uniformly in each corner of the grid (see Appendix C for dataset visualization and generation details).

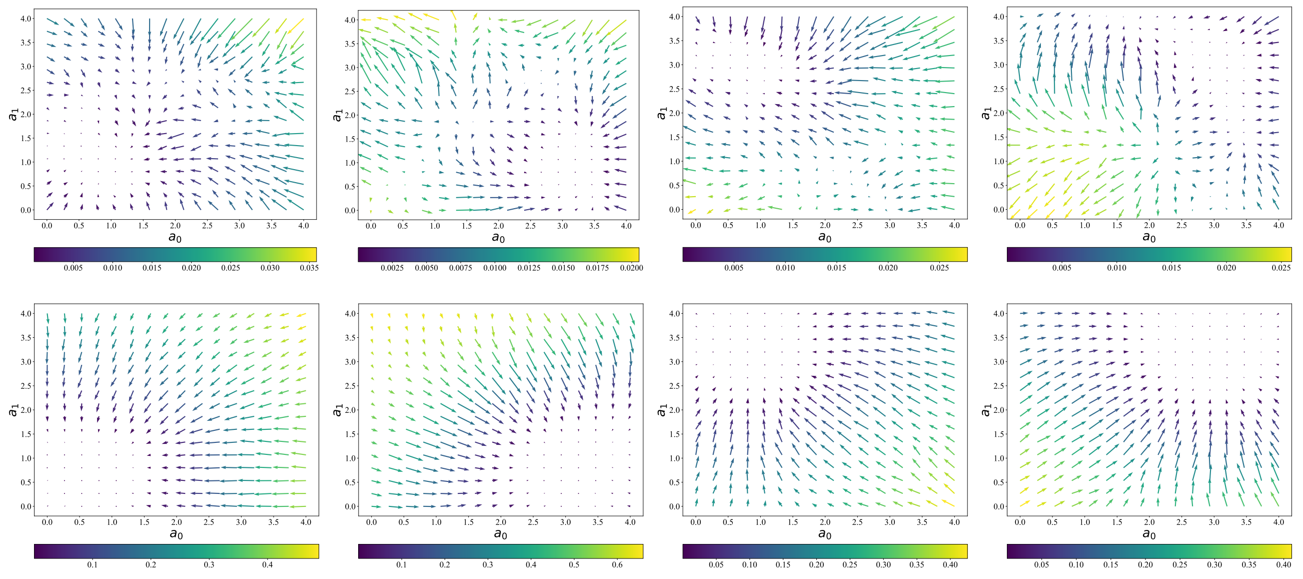
We fix the hyperparameters and pretrain two RNDs that differ only in the type of prior conditioning. The predictor uses simple concatenation. Next, in Figure 4, we plot the two-dimensional anti-gradient field for the anti-exploration bonus conditioned on each state. The effect of FiLM becomes more apparent in these plots. While the resulting anti-gradients for concatenation are noisy and only point in the direction of the minimum in a small neighbourhood, the directions for FiLM are smooth over the entire available action space and point to the correct global minimum for each state. While we cannot draw general conclusions from such a demonstration, based on the results of Section 4, we hypothesize that a similar phenomenon might exist in high-dimensional problems as well.

6.4. Exploring More Conditioning Pairs

One might wonder (1) how different types of conditioning for predictor and prior interact with each other and (2) where to introduce conditioning in terms of depth for it to be most beneficial.

To answer these questions, we return to the experiment from Section 4 and generate more variations for each type (where it is possible): conditioning on the first layer, on the last layer, and on all layers. We also look at two variations of the bilinear layer: full, as presented in Jayakumar et al. (2020), and simplified, which is used by default in PyTorch. In Figure 5 we plot the final MSE between the resulting policy and the behavioural one on the training data. Two interesting observations can be made from these findings.

First, FiLM may not be the only architecture with the right inductive biases for the prior, and both bilinear types with



(a) State 0 (b) State 1 (c) State 2 (d) State 3

Figure 4. Actions' anti-gradient field for the anti-exploration bonus conditioned on four categorical states at each corner for the toy problem introduced in Section 6.3. We visualize the dataset in Figure 7 in the appendix. The top row corresponds to RND with concatenation conditioning in the prior, while the bottom row corresponds to FiLM conditioning. As can be seen, the resulting anti-gradients for concatenation are noisy, while the directions for FiLM are smooth over the entire available action space.

conditioning on all layers can also achieve similar results. However, compared to FiLM, inner bilinear layers are much more computationally expensive, as they involve at least one 3D weight tensor and two additional 2D weight tensors, and the hidden dimensions are usually much higher than the input dimensions.

Second, it appears that conditioning on the last layer is most beneficial for the predictor, while conditioning on all layers is beneficial for the prior. In spite of that, it is difficult to draw broad conclusions, as different types may work well for new problems and domains.

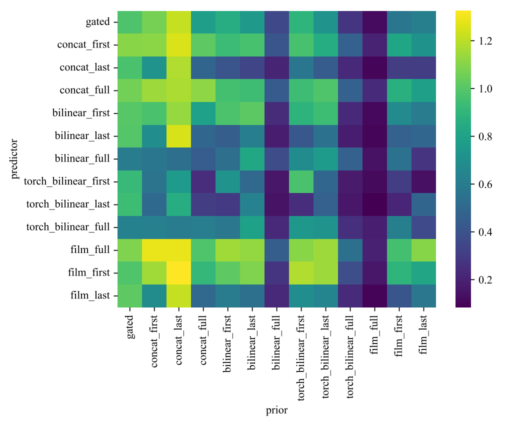


Figure 5. Mean squared error between actions of the actor trained with different conditioning for the predictor & prior and actions of the behavioral policy. We use the halfcheetah, walker2d and hopper medium datasets, with 3 seeds each. It can be seen that conditioning on each layer is beneficial for the priors, while for the predictors, it is better to condition on the last layer. Note that this experiment is in the setting of Section 4, that is, without a critic.

7. Related Work

Model-free offline RL. Most offline RL approaches focus on the distribution shift problem and overestimation bias of Q-values for OOD actions. Some researchers address this by imposing strict constraints for policy updates, penalizing the divergence from the behavioral policy with KL divergence, maximum mean discrepancy (MMD) distance (Kumar et al., 2019; Wu et al., 2019), simple mean squared error (MSE) (Fujimoto & Gu, 2021), or by re-weighting behavioral policy actions with the estimated advantages (Nair et al., 2020). Others directly regularize Q-values by lowering return estimates for OOD actions, preventing overestimation for unseen actions. For instance, Kumar et al. (2020), Ghasemipour et al. (2022) and Rezaeifar et al. (2022) explicitly introduce an optimization term that lowers Q-values for OOD actions, while An et al. (2021) penalizes implicitly by utilizing the lower-confidence bound (LCB) of Q-values. Alternatively, the evaluation of OOD actions can be avoided altogether by using the upper expectile value function (Kostrikov et al., 2021) or by policy optimization within a latent action space (Chen et al., 2022; Zhou et al., 2021; Akimov et al., 2022).

In our work, we follow the anti-exploration approach (Rezaeifar et al., 2022). In contrast to An et al. (2021); Ghasemipour et al. (2022); Yang et al. (2022), we completely eliminate ensembles for uncertainty estimation, thus reducing computational overhead without sacrificing performance. Moreover, unlike Rezaeifar et al. (2022), we have succeeded in using an RND for novelty detection in offline RL for continuous action spaces.

Estimation bias in Q-learning. In both offline and online reinforcement learning, off-policy Q-learning methods suffer from an overestimation bias in the temporal difference learning target (Van Hasselt et al., 2016; Fujimoto et al., 2018). This phenomenon is orthogonal to overestimation due to unseen actions in offline RL, as it occurs even in the presence of strong conservatism constraints. It is mainly caused by target prediction errors for seen transitions and their propagation due to the maximum operation $max_{a' \in A} Q(s', a')$. To address this problem, Fujimoto et al. (2018) introduced clipped double Q learning (Van Hasselt et al., 2016) in TD3, which uses a minimum of two critics. This approach was later used by Haarnoja et al. (2018) in SAC to improve stability and accelerate convergence.

In our work, we use clipped double Q-learning (Fujimoto et al., 2018), since SAC-RND is based on SAC (Haarnoja et al., 2018), and found it beneficial for stability. However, to ensure that it does not introduce additional conservatism, which can be a confounding factor for the impact of RND, we always set the number of critics to two.

Uncertainty estimation in offline RL. Uncertainty estimation is a popular technique in reinforcement learning and is used for a variety of purposes such as exploration, planning, and robustness. In offline RL, its use is mostly limited to modeling epistemic uncertainty (Clements et al., 2019), including measuring the prediction confidence of dynamics models (Yu et al., 2020; Kidambi et al., 2020) or critics (An et al., 2021; Rezaeifar et al., 2022). This approach can be further used to induce uncertainty-aware penalization during training.

Alternatively, uncertainty can help overcome suboptimal conservatism by designing more flexible offline approaches, e.g., conditioning on different levels of confidence to dynamically adjust the level of conservatism during evaluation (Hong et al., 2022) or using Bayesian perspective to design an optimal adaptive offline RL policy (Ghosh et al., 2022).

In our work, we estimate epistemic uncertainty and use it as an anti-exploration bonus to induce conservatism. Unlike previous approaches, we do not use ensembles to estimate epistemic uncertainty.

Efficient ensembles Ensembles are a powerful and simple non-Bayesian baseline for uncertainty estimation that outperform Bayesian neural networks in practice (Lakshminarayanan et al., 2017). However, training deep ensembles can be both memory intensive and computationally demanding, making the design of efficient ensembles an attractive research direction for which numerous methods have been developed. For example, Gal & Ghahramani (2016) proposed to use dropout to approximate Bayesian inference in deep Gaussian processes, and Durasov et al. (2021) derived a method to interpolate between dropout and full ensembles with fixed masks and controllable overlap between them. Meanwhile, Wen et al. (2020) significantly reduced the cost by defining each weight matrix as the Hadamard product of a shared weight among all ensemble members and a rank-one matrix per member.

Recently, Ghasemipour et al. (2022) showed that, in offline RL, none of the most popular approaches for efficient ensembles are capable of delivering performance that is comparable to naive ensembles, and that more work is needed in this research direction. In our work, we chose an alternative path for uncertainty estimation with RND, which was shown to a fast and competitive counterpart to ensembles (Ciosek et al., 2019).

8. Conclusion

In this work, we revisited the results from previous research (Rezaeifar et al., 2022), showing that with a naive choice of conditioning for the RND prior, it becomes infeasible for the actor to effectively minimize the anti-exploration bonus and discriminativity is not an issue. To solve this, we proposed conditioning based on FiLM, which led us to a new ensemble-free method called SAC-RND. We empirically validated that it achieves results comparable to ensemble-based methods and outperforms its ensemble-free counterparts. As such, we believe that our work is a valuable contribution to anti-exploration and uncertainty estimation in offline RL. In addition, we further outline the possible limitations of this work in the Appendix A.

References

Akimov, D., Kurenkov, V., Nikulin, A., Tarasov, D., and Kolesnikov, S. Let offline rl flow: Training conservative agents in the latent space of normalizing flows. In *3rd Offline RL Workshop: Offline RL as a"Launchpad"*, 2022.

An, G., Moon, S., Kim, J.-H., and Song, H. O. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization.

arXiv preprint arXiv:1607.06450, 2016.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., and Blundell, C. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, pp. 507–517. PMLR, 2020.

Baker, B., Akkaya, I., Zhokhov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *arXiv preprint arXiv:2206.11795*, 2022.

Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Chen, X., Ghadirzadeh, A., Yu, T., Gao, Y., Wang, J., Li, W., Liang, B., Finn, C., and Zhang, C. Latent-variable advantage-weighted policy optimization for offline rl. *arXiv preprint arXiv:2203.08949*, 2022.

Ciosek, K., Fortuin, V., Tomioka, R., Hofmann, K., and Turner, R. Conservative uncertainty estimation by fitting prior networks. In *International Conference on Learning Representations*, 2019.

Clements, W. R., Van Delft, B., Robaglia, B.-M., Slaoui, R. B., and Toth, S. Estimating risk and uncertainty in deep reinforcement learning. *arXiv preprint arXiv:1905.09638*, 2019.

Dumoulin, V., Perez, E., Schucher, N., Strub, F., Vries, H. d., Courville, A., and Bengio, Y. Feature-wise transformations. *Distill*, 2018. doi: 10.23915/distill.00011. https://distill.pub/2018/feature-wise-transformations.

Durasov, N., Bagautdinov, T., Baque, P., and Fua, P. Masksembles for uncertainty estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13539–13548, 2021.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Ghasemipour, S. K. S., Gu, S. S., and Nachum, O. Why so pessimistic? estimating uncertainties for offline rl through ensembles, and why their independence matters. *arXiv preprint arXiv:2205.13703*, 2022.

Ghosh, D., Ajay, A., Agrawal, P., and Levine, S. Offline rl policies should be trained to be adaptive. In *International Conference on Machine Learning*, pp. 7513–7530. PMLR, 2022.

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

Hong, J., Kumar, A., and Levine, S. Confidence-conditioned value functions for offline reinforcement learning. *arXiv preprint arXiv:2212.04607*, 2022.

Jarrett, D., Tallec, C., Altché, F., Mesnard, T., Munos, R., and Valko, M. Curiosity in hindsight. *arXiv preprint arXiv:2211.10515*, 2022.

Jayakumar, S. M., Czarnecki, W. M., Menick, J., Schwarz, J., Rae, J., Osindero, S., Teh, Y. W., Harley, T., and Pascanu, R. Multiplicative interactions and where to find them. 2020.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Kumar, A., Agarwal, R., Geng, X., Tucker, G., and Levine, S. Offline q-learning on diverse multi-task data both scales and generalizes. *arXiv preprint arXiv:2211.15144*, 2022.

Kurenkov, V. and Kolesnikov, S. Showing your offline reinforcement learning work: Online evaluation budget matters. In *International Conference on Machine Learning*, pp. 11729–11752. PMLR, 2022.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Lee, K.-H., Nachum, O., Yang, M., Lee, L., Freeman, D., Xu, W., Guadarrama, S., Fischer, I., Jang, E., Michalewski, H., et al. Multi-game decision transformers. *arXiv preprint arXiv:2205.15241*, 2022.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Lyu, J., Ma, X., Li, X., and Lu, Z. Mildly conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2206.04745*, 2022.

Nair, A., Gupta, A., Dalal, M., and Levine, S. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Nikulin, A., Kurenkov, V., Tarasov, D., Akimov, D., and Kolesnikov, S. Q-ensemble for offline rl: Don't scale the ensemble, scale the batch size. *arXiv preprint arXiv:2211.11092*, 2022.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Rezaeifar, S., Dadashi, R., Vieillard, N., Hussenot, L., Bachem, O., Pietquin, O., and Geist, M. Offline reinforcement learning as anti-exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8106–8114, 2022.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Smith, L., Kostrikov, I., and Levine, S. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.

Srivastava, R. K., Shyam, P., Mutz, F., Jaśkowski, W., and Schmidhuber, J. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

Tarasov, D., Nikulin, A., Akimov, D., Kurenkov, V., and Kolesnikov, S. CORL: Research-oriented deep offline reinforcement learning library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022. URL `https://openreview.net/forum?id=SyAS49bBcv`.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Wen, Y., Tran, D., and Ba, J. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*, 2020.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Yang, R., Bai, C., Ma, X., Wang, Z., Zhang, C., and Han, L. Rorl: Robust offline reinforcement learning via conservative smoothing. *arXiv preprint arXiv:2206.02829*, 2022.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Zhou, W., Bajracharya, S., and Held, D. Plas: Latent action space for offline reinforcement learning. In *Conference on Robot Learning*, pp. 1719–1735. PMLR, 2021.

A. Limitations

Several limitations of this work should be noted. While we provide extensive experiments and ablations on the well-established D4RL benchmark for continuous control, we lack the benchmarks for discrete actions and visual state spaces. Thus, we cannot be absolutely certain that the same pair of predictor and prior will generalize to the new domains, and additional exploration on new problems may be needed. However, the general properties of the good RND prior for offline RL uncovered in this work remain the same and should guide practitioners in the new applications of our method. Furthermore, we have explored only a limited set of the many available conditioning variations. Finally, our method inherits the limitations of anti-exploration style algorithms (Rezaeifar et al., 2022) — sensitivity to the reward and RND bonus scale, as this greatly affects the level of conservatism, thus requiring the α sweep for each problem separately. To alleviate this limitation, we additionally explicitly checked the sensitivity to this parameter using the expected online performance metric in the Appendix G.

B. Previous Research Results

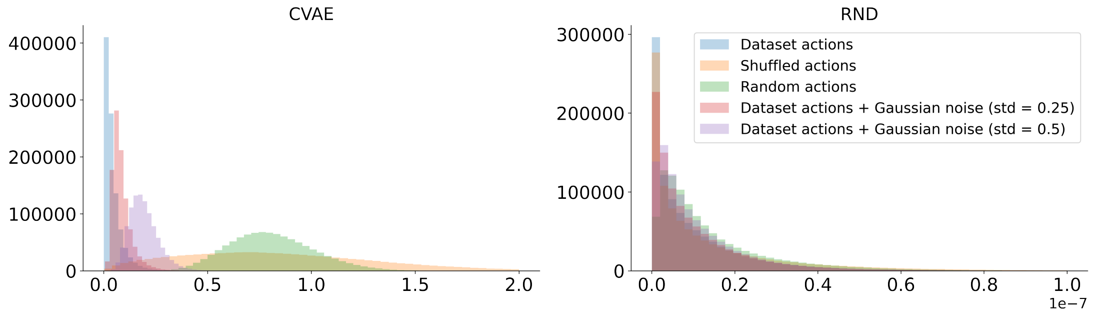


Figure 6. Anti-exploration bonus on walker2d-medium dataset for RND and CVAE. Note that figure taken from Rezaeifar et al. (2022) for a convenient comparison with our results in Figure 2.

C. Toy Dataset

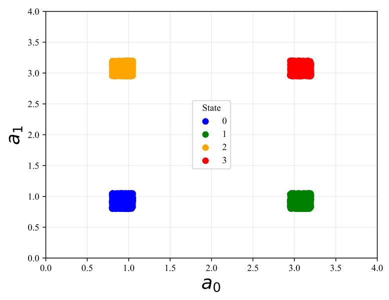


Figure 7. Toy dataset visualization introduced in Section 6.3. This toy dataset consists of four categorical states for each corner of the limited 2D actions grid. For each state, we uniformly sample 4096 two-dimensional actions within a limited square. We use one-hot encoding for the states during RND training.

D. Implementation Details

In our experiments, we use hyperparameters from Table 4 where possible and sweep over α to pick the best value for each dataset. We implement all of our models using the Jax (Bradbury et al., 2018) framework. For the exact implementation of conditioning variants for predictor and prior networks, refer to our code at `https://github.com/tinkoff-ai/sac-rnd`. Similarly to Nikulin et al. (2022); Kumar et al. (2022); Smith et al. (2022), we add Layer Normalization (Ba et al., 2016) to the critic after each layer as it greatly improves stability and convergence speed. For SAC-N in Section 4 we use the implementation from the CORL library (Tarasov et al., 2022). All experiments were performed on V100 and A100

GPUs. With our implementation, each training for 3 million training steps usually takes ~ 40 minutes to run (~ 15 minutes for the typical 1 million steps).

Gym Domain. We use the v2 version of each dataset. We follow the An et al. (2021) approach and run our algorithms for 3 million training steps and report the final normalized average score over 10 evaluation episodes. For the final experiments, we use 4 seeds, while using less for hyperparameter tuning. We tune the α coefficient over the $\{1.0, 3.0, 4.0, 5.0, 8.0, 9.0, 10.0, 13.0, 15.0, 20.0, 25.0\}$ range for the walker and hopper datasets. We found that the halfcheetah datasets require a lower level of conservatism, which is why we tune over the $\{0.001, 0.1, 0.3, 0.5, 0.8, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0\}$ range for these datasets while keeping the same number of candidates. We follow the Ghasemipour et al. (2022) approach and choose the best α for each dataset (see Table 5).

AntMaze Domain. We use the v1 version of each dataset due to the fact that the v0 version has major problems and bugs during generation (e.g., some trajectories have discontinuities where the agent teleports from one part of the maze to another [2]). We follow the An et al. (2021) approach and run our algorithms for 3 million training steps and report the final normalized average score over 100 evaluation episodes. Same as Chen et al. (2022), we scale the reward by 100.0. We found that actor and critic require different levels of conservatism in these tasks, which is why we chose to decouple α and use separate values (the same approach was used in Rezaeifar et al. (2022)). We tune the α for the actor in the $\{0.5, 1.0, 1.5\}$ range, and α for the critic in the $\{0.001, 0.01, 0.1\}$ range. We follow the Ghasemipour et al. (2022) approach and choose the best α for each dataset (see Table 7).

E. Hyperparameters

Table 4. SAC-RND general hyperparameters.

Parameter	**Value**
optimizer	Adam (Kingma & Ba, 2014)
batch size	1024 (256 on antmaze-*)
learning rate (all networks)	1e-3 (3e-4 on antmaze-*)
tau (τ)	5e-3
hidden dim (all networks)	256
num layers (all networks)	4
RND embedding dim (all tasks)	32
target entropy	-action_dim
gamma (γ)	0.99 (0.999 on antmaze-*)
nonlinearity	ReLU

Table 5. SAC-RND best hyperparameters used in D4RL Gym domain.

Task Name	α
halfcheetah-random	0.1
halfcheetah-medium	0.3
halfcheetah-expert	6.0
halfcheetah-medium-expert	0.1
halfcheetah-medium-replay	0.1
halfcheetah-full-replay	3.0
hopper-random	5.0
hopper-medium	25.0
hopper-expert	20.0
hopper-medium-expert	15.0
hopper-medium-replay	8.0
hopper-full-replay	3.0
walker2d-random	1.0
walker2d-medium	8.0
walker2d-expert	4.0
walker2d-medium-expert	25.0
walker2d-medium-replay	8.0
walker2d-full-replay	3.0

[2]https://github.com/Farama-Foundation/D4RL/issues/77

Table 6. SAC-RND best hyperparameters used in D4RL AntMaze domain.

Task Name	α (actor)	α (critic)
antmaze-umaze	1.0	0.1
antmaze-umaze-diverse	1.0	0.1
antmaze-medium-play	0.5	0.001
antmaze-medium-diverse	1.0	0.01
antmaze-large-play	1.0	0.01
antmaze-large-diverse	0.5	0.01

F. Runtime Comparison

Table 7. Runtime comparison for different algorithms on halfcheetah-medium-v2 dataset. We measure time on V100 GPU for standard 1M updates with batch size 256 and identical network sizes. For SAC-N (An et al., 2021) we report times for all ensemble size configurations from the original publication. Note that, as we discussed in Appendix D, all algorithms were implemented in Jax (Bradbury et al., 2018), which is typically much faster for small networks than PyTorch (Paszke et al., 2019) alternatives.

Algorithm	**Updates / s** $\uparrow$	**Runtime (m)** $\downarrow$
TD3+BC	1485	11.2
SAC-2	1285	12.9
SAC-RND	850	19.6
SAC-10	809	20.5
SAC-20	559	29.7
SAC-100	171	97.3
SAC-200	93	178.3
SAC-500	39	424.0

G. Sensitivty to Hyperparameters

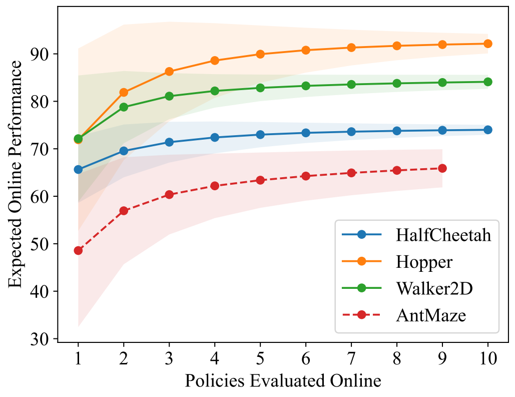
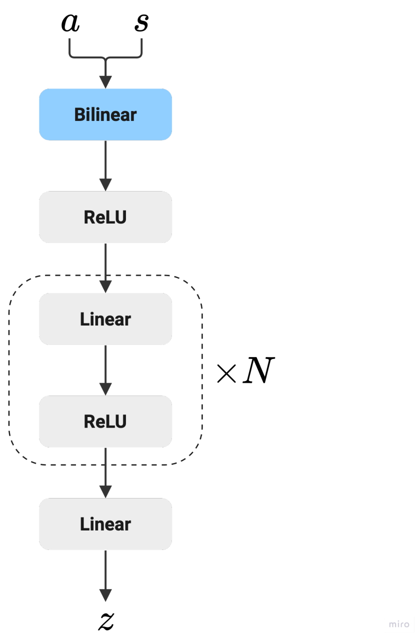
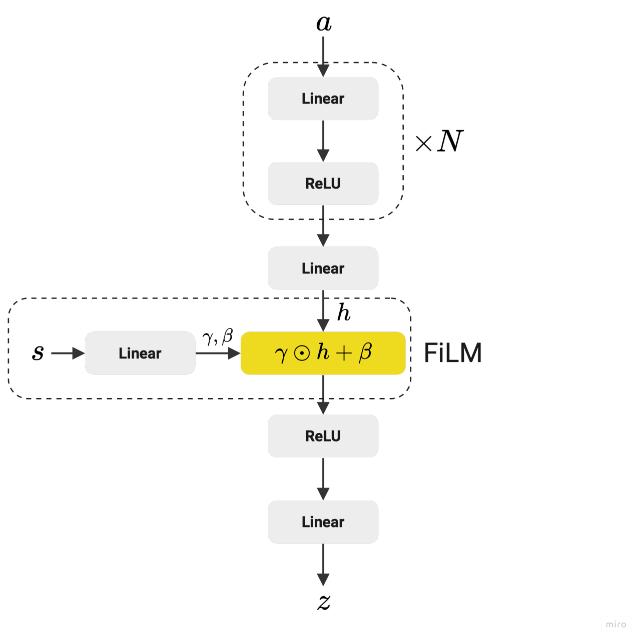


Figure 8. Expected Online Performance (Kurenkov & Kolesnikov, 2022) under uniform offline policy selection. It can be seen, that for satisfactory results in all domains a budget of at least five policies for online evaluations is needed.

H. Architecture Visualization



(a) Predictor



(b) Prior

Figure 9. Visualization of the final RND architecture described in the section 5. For the predictor we use bilinear (Jayakumar et al., 2020) conditioning on the first layer. For the prior we use FiLM (Perez et al., 2018) conditioning on the penultimate layer before nonlinearity. States are encoded with a single linear layer of double hidden size, which output is then divided into equal parts for the γ and β. While the number of linear layers can be arbitrary, in our work we use $N = 2$, so the network size is 4 layers in total (see Table 4).

I. Pseudocode

Algorithm 1 Soft Actor-Critic with Random Network Distillation (SAC-RND)

```
Initialize policy parameters $\theta$, Double Q-function parameters $\{\phi_1, \phi_2\}$, RND predictor and prior parameters $\{\psi, \psi'\}$, and offline replay buffer $\mathcal{D}$
for desired number of pretraining steps do
    Sample a mini-batch $B = \{(s, a)\}$ from $\mathcal{D}$
    Update RND predictor weights $\psi$ with gradient descent using
        $$\nabla_\psi \frac{1}{|B|} \sum_{s \in B} \Big[ \|f_\psi(s, a) - \bar{f}_{\bar{\psi}}(s, a)\|_2^2 \Big]$$
end for
for desired number of training steps do
    Sample a mini-batch $B = \{(s, a, r, s')\}$ from $\mathcal{D}$
    Compute target Q-values (shared by all Q-functions):
        $$y(r, s') = r + \gamma \Big[ \min_{j=1,2} Q_{\bar{\phi}_i}(s', a') - \beta \log \pi_\theta(a'|s') - \alpha b(s', a') \Big]$$
    where $a' \sim \pi_\theta(\cdot|s')$ and $b(s', a')$ is an anti-exploration bonus defined by Eq. (3).
    Update each Q-function $Q_{\phi_i}$ with gradient descent using
        $$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s') \in B} \Big[ Q_{\phi_i}(s, a) - y(r, s') \Big]^2$$
    Update policy with gradient ascent using
        $$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \Big[ \min_{j=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \beta \log \pi(\tilde{a}_\theta(s)|s) - \alpha b(s, \tilde{a}_\theta(s)) \Big]$$
    where $\tilde{a}_\theta(s)$ is a sample from $\pi(\cdot|s)$ which is differentiable w.r.t. $\theta$ via the reparametrization trick.
    Update target networks with $\bar{\phi}_i \leftarrow (1 - \rho)\bar{\phi}_i + \rho \phi_i$
end for
```

Algorithm 2 Simplified SAC-RND (without a critic) used in experiments for Section 4 and Section 6.4.

```
Initialize policy parameters $\theta$, RND predictor and prior parameters $\{\psi, \psi'\}$, and offline replay buffer $\mathcal{D}$
for desired number of pretraining steps do
    Sample a mini-batch $B = \{(s, a)\}$ from $\mathcal{D}$
    Update RND predictor weights $\psi$ with gradient descent using
        $$\nabla_{\psi} \frac{1}{|B|} \sum_{s \in B} \Big[ \| f_{\psi}(s, a) - \bar{f}_{\bar{\psi}}(s, a) \|_2^2 \Big]$$
end for
for desired number of training steps do
    Sample a mini-batch $B = \{(s, a, r, s')\}$ from $\mathcal{D}$
    Update policy with gradient descent using
        $$\nabla_{\theta} \frac{1}{|B|} \sum_{s \in B} \Big[ \beta \log \pi(\tilde{a}_{\theta}(s)|s) + b(s, \tilde{a}_{\theta}(s)) \Big]$$
    where $\tilde{a}_{\theta}(s)$ is a sample from $\pi(\cdot|s)$ which is differentiable w.r.t. $\theta$ via the reparametrization trick and $b(s', a')$ is an
    anti-exploration bonus defined by Eq. (3).
end for
```