Neural Computing and Applications (2023) 35:22563–22576
https://doi.org/10.1007/s00521-023-08845-x

ORIGINAL ARTICLE

ACRE: Actor-Critic with Reward-Preserving Exploration

Athanasios Ch. Kapoutsis[1] · Dimitrios I. Koutras[1] · Christos D. Korkas[1] · Elias B. Kosmatopoulos[1]

Received: 27 January 2023 / Accepted: 28 June 2023 / Published online: 14 August 2023

Abstract
While reinforcement learning (RL) algorithms have generated impressive strategies for a wide range of tasks, the performance improvements in continuous-domain, real-world problems do not follow the same trend. Poor exploration and quick convergence to locally optimal solutions play a dominant role. Advanced RL algorithms attempt to mitigate this issue by introducing exploration signals during the training procedure. This successful integration has paved the way to introduce signals from the intrinsic exploration branch. ACRE algorithm is a framework that concretely describes the conditions for such an integration, avoiding transforming the Markov decision process into time varying, and as a result, making the whole optimization scheme brittle and susceptible to instability. The key distinction of ACRE lies in the way of handling and storing both extrinsic and intrinsic rewards. ACRE is an off-policy, actor-critic style RL algorithm that separately approximates the forward novelty return. ACRE is shipped with a Gaussian mixture model to calculate the instantaneous novelty; however, different options could also be integrated. Using such an effective early exploration, ACRE results in substantial improvements over alternative RL methods, in a range of continuous control RL environments, such as learning from policy-misleading reward signals. Open-source implementation is available here: https://github.com/athakapo/ACRE.

Keywords Reinforcement learning · Actor-critic · GMM · Exploration

1 Introduction

Reinforcement learning (RL) has been the main driving force for developing the minds of our future robots. Breakthroughs in the hardware (e.g., GPU acceleration) and software (e.g., powerful RL algorithms) have led to extraordinary results, such as interacting with humans in a conversational way [25], surpassing human intelligence at playing Atari games [2], performing dexterous manipulation tasks [10], navigating using self-supervised learning [13], etc. However recent promising attempts, e.g., [33] and [39], the level of performance achieved in the discrete and deterministic world, mainly in video or board games (e.g., [4, 31]), seems out of reach in the continuous and stochastic domain of real-life robots. On top of that, the available simulators do not provide engines with arbitrary accuracy, rendering the interaction with the real-world mandatory. All the above reinforce the quick convergence to sub-optimal solutions or the acquisition of brittle policies that do not generalize well on a wide variety of real-world situations [8].

To mitigate the aforementioned issues, almost all model-free RL algorithms attempt to encourage exploration somehow, e.g., ϵ-greedy [19], noise-corrupted actions [17], stochastic policies [30], entropy-regulated learning [12], etc. The most advanced off-policy RL algorithms, e.g., SAC [12] and TD3 [9], own their premium performances, to some extent, in the extra introduced exploration signals during the learning procedure (in addition to the exploration noise during the interaction with the environment). Such an observation paves the way for

✉ Athanasios Ch. Kapoutsis
athakapo@iti.gr

Dimitrios I. Koutras
dkoutras@iti.gr

Christos D. Korkas
chriskorkas@iti.gr

Elias B. Kosmatopoulos
kosmatop@iti.gr

[1] Information Technologies Institute, The Centre for Research & Technology, Hellas, 6th km Charilaou-Thermi Rd, 57001 Thessaloniki, Thermi, Greece

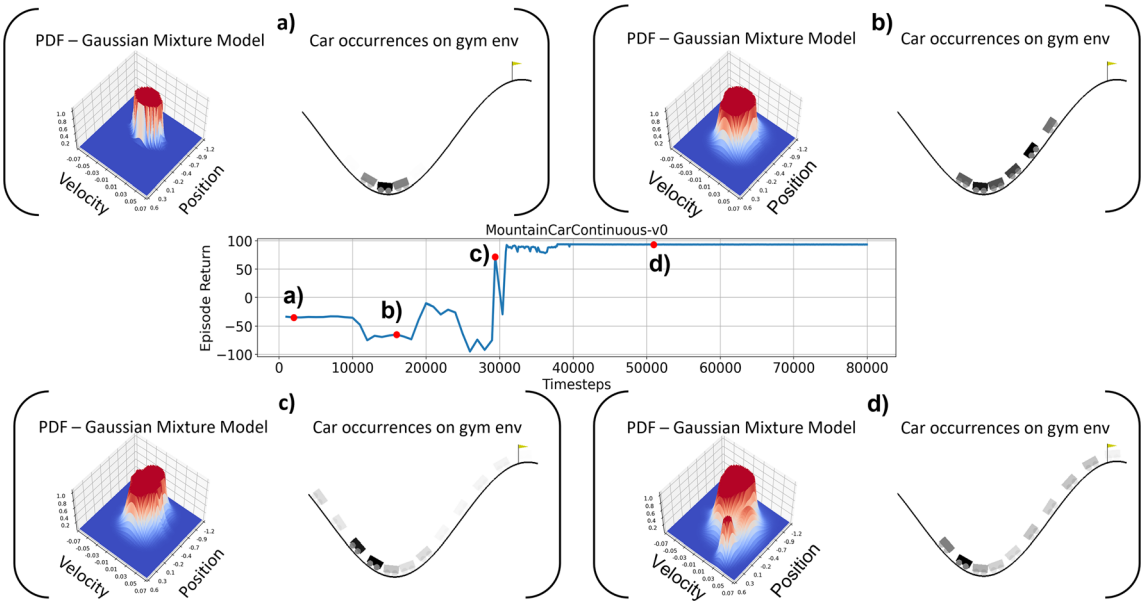


Fig. 1 ACRE performance insights on the MountainCarContinuous OpenAI-gym environment. Each one of the four snapshots depicts (i) on the left-hand side the belief of Gaussian mixture model with respect to states space (inverse novelty) and (ii) on the right-hand side a superimposed visualization of the car's position (the dimmer the picture of the car, the less frequent to find it there) for the corresponding episode. **a** Initial episodes where the car moves randomly, exploring only a tiny subset of the whole state space. **b** Based on the reward feedback, the car attempts to move closer to the flag; however, the environmental dynamics severely limit its performance. **c** Exploration now pays off, and the car has reached even the farthest states. **d** Based on these experiences, ACRE acquires and retains the optimal policy

directly integrating into the learning process exploration signals from the literature of intrinsic exploration strategies (e.g., [1, 28]). Although these application-specific, intrinsic exploration signals can be pretty appealing, their integration with the main body of the RL algorithm requires special attention. A "blind" incorporation with the environmental reward could give rise to several issues that may significantly hinder the additional performance improvements, especially in real-life environments with more complex states/actions, e.g., [32] and [14].

The proposed Actor-Critic with Reward-Preserving Exploration (ACRE) algorithm is a first attempt to formalize the integration of such intrinsic signals into the main body of off-policy actor-critic[1] RL algorithms without jeopardizing the integrity of the learning procedure. A Gaussian mixture model (GMM) is employed on the already visited states to perform a novelty assessment in terms of density estimation. It is crucial to mention that, although the GMM approach was found to be quite efficient, a different mechanism for estimating the novelty of transition could also be utilized to achieve a better or more specific performance of the problem at hand. In the heart of the proposed ACRE algorithm lies a separate module capable of estimating online the forward novelty (till the end of the episode) from each specific state. During the core update of the actor-critic structure, both value (Q) and policy (π) networks are also trained to optimize this forward novelty, stemming from the soft actor-critic framework [11]. In a nutshell, the ACRE algorithm introduces a new way of handling environmental and novelty returns. The environmental reward and the corresponding transition are stored "as is" in a standard replay buffer. On the contrary, the intrinsic, time-varying novelty return is estimated online based on the cumulative novelties of the states since the previous training cycle. Conceptually, ACRE deals with the exploration problem as having a separate time-varying Markov decision process (MDP), simultaneously with the original RL problem.

Figure 1 graphically illustrates the effect of ACRE on the policy-misleading reward environment of MountainCar, to escape local optimum without jeopardizing the gathered environmental transitions. From the 4 snapshots, which depict key insights related to the learning process, we focus on d) to deduce an exciting remark. The generation of new Gaussian curves around the velocity value of 0.025 is not by any means random. Actually, based on the dynamics of the environment, this is the exact velocity that is required to reach the goal position (see Fig. 2b of [15]). The RL agent learns that it only needs to reach this velocity to avoid spending time gaining extra speed that is not required. As a matter of fact, any additional movements would add more

[1] Methods that learn approximations to both policy and value functions are often called actor-critic methods, where "actor" is a reference to the learned policy, and "critic" refers to the learned value function, usually a state-value function [34].

steps to the episode length, and thus, based on the reward scheme of MountainCar, would result in inferior returns.

Beyond this indicative example, Sect. 4 presents a thorough study of the ACRE's performance. Sections 4.1 and 4.2 present an ablation study with respect to different components of the ACRE methodology, analyzing (i) the performance of core ACRE methodology as novelty integration mechanism and (ii) the effectiveness of GMM-based intrinsic reward signal, respectively. Finally, Sect. 4.3 presents an extensive study of the performance of ACRE against 6 state-of-the-art RL approaches on 12 continuous control tasks retrieved from 3 of the most well-established collections of RL benchmarks. All in all, ACRE's ability to explore effectively while retaining the reward signal intact led to premium performance. More specifically, in the majority of the cases, ACRE matched the best-achieved performance—which was not attained by a single RL algorithm alternative—while, at the same time, in two environments, the early exploration resulted in significant performance improvements over the status quo.

2 Preliminaries

2.1 Notation

We consider a standard reinforcement learning format, where an agent is allowed to interact with the environment in discrete timesteps t. The standard approach formulates such a setup into an MDP, described by a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma, d\}$. The state space $\mathcal{S}$ and action space $\mathcal{A}$ are assumed to be continuous, and the transition dynamics $\mathcal{P}(s'||s,a) : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,1]$ represent the environment-related probability density of arriving at a next state $s' \in \mathcal{S}$ given the current state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$. ρ_0 represents the initial state distribution from which s admits its values at the beginning of each episode. Each transition gets assessed by the environment with a scalar reward $r(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The Boolean signal $d \in \{0,1\}$ indicates the termination of an episode (e.g., exceeding the maximum number of iterations or colliding with an obstacle). The ultimate objective is the generation of a policy $\pi(s) : \mathcal{S} \to \mathcal{A}$ that maximizes the expected sum of rewards discounted by a factor $\gamma \in (0,1]$: $\mathbb{E}_{s_t \sim \mathcal{P}, a_t \sim \pi}\left[\sum_t \gamma^t r(s_t, a_t)\right]$.

2.2 Exploration elements in SOTA RL algorithms

While the aforementioned problem setup formally describes the mission of the RL agent, trying to "blindly" optimize such an objective may lead to getting stuck in extremely sub-optimal behavior due to poor exploration of the available $\mathcal{S} \times \mathcal{A}$. More often than not, in model-free, RL algorithms, the policy that gathers the new interaction samples with the environment is a combination of the current best policy π_{best}, fused with some exploration signal, e.g., ϵ-greedy-like exploration strategies (e.g., DQN [19]), policy stochasticity (e.g., PPO [30], TRPO [29]), noise-corrupted actions (e.g., DDPG [17]), etc. Recent state-of-the-art, off-policy RL algorithms introduce Bellman backup updates' exploration elements by implementing either non-deterministic (i.e., TD3 [9]) or entropy-regulated (i.e., SAC [12]) target updates. Table 1 summarizes all these exploration mechanisms in the state-of-the-art RL algorithms.

2.3 Exploration through extra reward signal

A special type of RL exploration strategy that has gained a lot of attention alters the received environmental reward to serve also as an exploration assessment. To accomplish that, these approaches build r_e as an intrinsic/information gain signal. Such an intrinsic reward represents an estimation with respect to the novelty of the new state (or state–action pair), usually in a form of *counting* [35], *diversity* [7], or *curiosity* [27]. This reward is mostly combined linearly with the environment's reward, i.e., $\tilde{r} \triangleq r + r_e$ [1, 24, 28]. In essence, they alter the original MDP setup to $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \tilde{r}, \rho_0, \gamma, d\}$ so as to enable extra exploration capabilities. After doing so, one of the state-of-the-art RL algorithms from Table 1, with all its exploration mechanisms in place, is employed to address the policy construction for this altered MDP. The problem with such an approach is that assuming a replay buffer, the RL agent may attempt to train on corrupted reward signals that probably contain non-relevant exploration signals, due to all the transitions that have been performed afterward. This can lead to instability and also without any guarantee that the achieved policy is optimized for the problem at hand.

3 Actor-Critic Reward-Preserving Exploration (ACRE) algorithm

Within this paper, we discuss why the exploration problem does not fit well inside the original MDP, and building around this idea, we propose ACRE, an off-policy, actor-critic style RL algorithm that takes special care of environmental and exploration returns.

3.1 Exploration is a separate time-varying MDP

Disregarding for the moment the exact method of acquiring a novelty estimation for every state, let us assume that at each timestep t and after observing state s', a novelty reward $r_e(s', t) : \mathcal{S} \to \mathbb{R}$ can be assigned. Please note that $r_e(s', t)$ is time varying, meaning that its form actually depends on all

Table 1 Exploration mechanisms in state-of-the-art RL algorithms

Algorithm	Environment interactions		Learning	
	Action generation	Exploration	Bellman backup update	Exploration
TRPO	$\pi(\cdot\|\|s)$	Policy stochasticity	–	None
PPO	$\pi(\cdot\|\|s)$	Policy stochasticity	–	None
DDPG	$\mu(s)+\mathcal{N}_{ou}$	Ornstein-Uhlenbeck process	$r+\gamma Q(s',\mu(s'))$	None
TD3	$\mu(s)+\mathcal{N}(0,\sigma)$	Zero-mean Gaussian noise	$r_t+\gamma Q\big(s',\tilde{a}'\big)$, $\tilde{a}'=\mu(s')+\mathcal{N}(0,\sigma)$	Zero-mean Gaussian noise
SAC	$\pi(\cdot\|\|s)$	Policy stochasticity	$r+\gamma\Big(Q(s',\tilde{a}')-\alpha\log\pi(\tilde{a}'\|\|s')\Big)$, $\tilde{a}'\sim\pi(\cdot\|\|s')$	Policy entropy

previously visited states. Exploration-wise only, the agent should pick states according to an exploration policy π_e that maximizes $J_e=\mathbb{E}_{s_0\sim\rho_0,a_t\sim\pi_e}\left[\sum_t\gamma^t r_e(\mathcal{P}(s_{t+1}\|s_t,a_t),t)\right]$. Therefore an extra time-varying MDP, described by the tuple $\{\mathcal{S},\mathcal{A},\mathcal{P},r_e,\rho_0,\gamma,d\}$, is now formed with the key difference that reward function r_e is now time varying and actually dependent on all the previously visited states.

Although a policy π_e that only maximizes J_e would not be particularly fruitful (technically would be equivalent to just arriving at all states with a uniform probability), this distinction acknowledges the different nature of these two signals. The reward signal r from the environment that assesses a transition from state s to s' after applying action a is considered to be static[2]. On the other hand, novelty value r_e is a dynamically evolving signal that is directly correlated with the so-far visited states. The latter means that the novelty signal r_e related to the transition from a state s to s' after applying action a is not constant. The above analysis endorses that careful integration of the exploration and reward signal is needed to avoid rendering the problem at hand a time-varying MDP with known challenges [23]. That is of paramount importance in off-policy algorithms (e.g., SAC, TD3, A3C, etc.) that utilize past transitions to update the current policy and value networks.

To retain the information of the original problem intact, we propose two different structures for storing these two sources of information. More specifically, the environmental reward is stored in a traditional replay buffer, while the novelty signal is represented as a dynamic model that changes over time. Similarly to a V-value estimation for the original MDP, we build a novelty return estimator fitted on the calculated $\sum_t\gamma^t r_e$. By doing so, we ensure that the state's future novelty is indeed taken into consideration without jeopardizing the integrity of the environmental reward. In the following subsections, we analyze how these elements can be combined into an actor-critic framework, constituting the ACRE algorithm.

3.2 Estimate novelty returns for exploration MDP

Gaussian mixture models (GMMs) are universal approximators of densities; therefore, they can be used to approximate each state's novelty. Additionally, due to the nonparametric nature of the GMM, we do not need to make any assumption about the distribution that collects the environment's states, rendering them a quite universal independent of the environment and the current policy. Following GMM procedure, each state's novelty will be given by:

$$r_e(s,t)=\sum_{i=1}^{l}w_i p(s||\mu_i,\Sigma_i), \tag{1}$$

where $p(s||\mu_i,\Sigma_i)$ denotes the probability density function of a Gaussian indexed by i, where $i\in\{1,2,\ldots,l\}$ and w_i denotes the corresponding mixing factor. Periodically, a standard expectation–maximization (EM) algorithm is applied to a randomly selected subset of previously visited states to perform the fitting of the mixture of Gaussians, i.e., calculate $\{w_i,\mu_i,\Sigma_i\}\forall i=1,\ldots,l$. Utilizing this GMM novelty estimator of (1), we can now explicitly calculate the forward novelty from each state s_t using:

$$\hat{R}_e(s_t)=\sum_{t'=t}^{T}\gamma^{t'-t}r_e(s_{t'},t'), \tag{2}$$

where s_T denotes the last state of the episode, i.e., $d=1$. To be able to predict these novelty returns, we build a neural network estimator $N_u(s)$ that fits on batches of $\{s_i,\hat{R}_e(s_i)\}$ from the information buffer $\mathcal{I}$, as follows:

$$\nabla_u\frac{1}{\|\mathcal{I}\|}\sum_{s\in\mathcal{I}}\big(N_u(s)-\hat{R}_e(s)\big)^2 \tag{3}$$

In RL terms, the update of $N_u(s)$ is performed "on-policy" (in terms of novelty return estimation), on all the states that

[2] Actually, for stochastic environments, the environmental reward may change. Still, this variation depicts the innate uncertainty of the environmental dynamics $\mathcal{P}(s'\|s,a)$ that can be estimated by a large enough number of samples.

were visited till the previous forward novelty update, i.e., $\hat{R}_e(s_i), \forall i = \{t - T_w, t\}$ where T_w denotes the information buffer $\mathcal{I}$ size. Please note that a state s is considered to have the maximum forward novelty as $N_u(s) \to 0$; therefore, to translate this into the common RL maximization framework the $-N_u(s)$ quantity is going to be used.

3.3 Incorporation of novelty returns estimations inside actor-critic framework

Having defined the updating procedure of $N_u(s)$ (1)–(3), which can estimate the novelty return from s till the end of the episode, now we turn on how this value can be utilized inside the construction of the policy to tackle the original RL problem. The agent interacts with the environment by drawing actions from the so-far learned policy distribution $a \sim \pi_\theta(s)$. After receiving the corresponding reward signal r and the new s', the collected tuple (s, a, r, s', d) is stored in replay buffer $\mathcal{D}$. For the update of both actor and critic, a batch of transitions B is randomly sampled from $\mathcal{D}$. For each ith tuple $(s_i, a_i, r_i, s_i', d_i)$ from B, we calculate the target for the critic update based on the Bellman backup rule regulated by the estimated novelty return, i.e.,

$$y_i = r_i + \gamma(1 - d_i)\left(\min_{j=1,2} Q_{\phi_{\text{targ,j}}}(s_i', \tilde{a_i}') - \beta N_u(s_i')\right), \tag{4}$$

where $\tilde{a_i}' \sim \pi_\theta(s_i')$. The temperature parameter β weights the importance of novelty over the reward returns and thus controls the stochasticity of the converged policy. Apart from the extra term evaluating the novelty of being at state s', the update rule of (4) employs a double Q-learning approach, i.e., $\min_{j=1,2} Q_{\phi_{\text{targ,j}}}(s_i', \tilde{a_i}')$ term, to compensate for the overestimation bias in Q-function learning [37], as it is commonly used in the state-of-the-art Q-learning approaches (e.g., [9, 12]). Additionally, and in line with most stable Q-learning implementations (e.g., [20]) two additional target networks $Q_{\text{targ},1}, Q_{\text{targ},2}$ are utilized for the y_i calculation to bypass convergence issues [17]. Having calculated the target values y_i, both Q_{ϕ_1} and Q_{ϕ_2} are updated by one step of gradient descent for $j = 1, 2$ using:

$$\nabla_{\phi_j} \frac{1}{\|B\|} \sum_{(s,a,r,s',d) \in B} \left(Q_{\phi_j}(s, a) - y(r, s', d)\right)^2 \tag{5}$$

The actor part, responsible for the policy generation, builds around the previously defined critics $Q_{\phi_{\phi,1}}$, $Q_{\phi_{\phi,2}}$ and the novelty return N_u. Conceptually, in the policy improvement step, $\forall s \in \mathcal{B}$ we generate the new policy as follows:

$$\pi_{\text{new}}(\cdot \| s) = \arg\max_{\pi' \in \Pi} \left[\min_{j=1,2} Q_{\phi_{\text{targ,j}}}(s, \pi'(s)) - \beta N_u(s)\right] \tag{6}$$

Following [12], the new actor is being calculated by jointly optimizing Q function and estimated novelty returns $N_u(s)$. The last step of the actor-critic update cycle invokes a polyak-averaged rule for the update of Q target networks, i.e., $\phi_{\text{targ,j}} \leftarrow \rho\phi_{\text{targ,j}} + (1 - \rho)\phi_j$, for $j = 1, 2$. The complete ACRE pseudo-code, combining the estimation of $N_u(s)$ (Sect. 3.2), is provided in Appendix A.

Remark 1 ACRE is agnostic with respect to the nature of the novelty reward signal; thus, instead of the GMM, any other universal approximator scheme (e.g., RBFs)—probably more specific to the problem at hand—can be applied. Actually, in Sect. 4 an alternative methodology that utilized ACRE with RND [7] (instead of GMM) is being described and evaluated.

Remark 2 Although ACRE may assign different novelty signals on the same states at different timesteps, all the transitions are permanently stored "as measured" from the environment. By doing so, ACRE manages to explore online, but in a reward-preserving manner, which is suitable for the off-policy family of RL algorithms.

Remark 3 Each state gets an estimation regarding the cumulative novelty expected to receive until the end of the episode. Thereby, ACRE avoids myopic evaluation of the exploration status while, at the same time, properly tackling the underlying time-varying exploration MDP (as described in Sect. 3.1).

Remark 4 The overall stability of the training is significantly improved (reduced standard deviation in Figs. 2, 6, 7 and 8) by the fact that (3) and (5) are updated using different—tailored to each case characteristics—chunks of data.

4 Experimental evaluation

The objective of the empirical evaluation is to study the following questions: (1) What is the performance of the core ACRE's novelty integration mechanism? (2) Is the GMM alone an adequate intrinsic reward estimator? (3) How does the complete ACRE algorithm compare to state-of-the-art methods on a diverse set of continuous control environments from different suites? Please note that in all forthcoming learning curves (Figs. 2, 4, 6, 7 and 8), the average, over 10 different seeds $(0, 1, 2, \ldots, 9)$, value is depicted with the thick line and the shaded region around it represents the corresponding standard deviation.

4.1 Analysis of ACRE novelty signal integration mechanism

We first investigate the performance of the core ACRE methodology, disregarding the effectiveness of GMM as a novelty estimator for the moment. Thus, we employ a popular variation of PPO methodology capable of incorporating

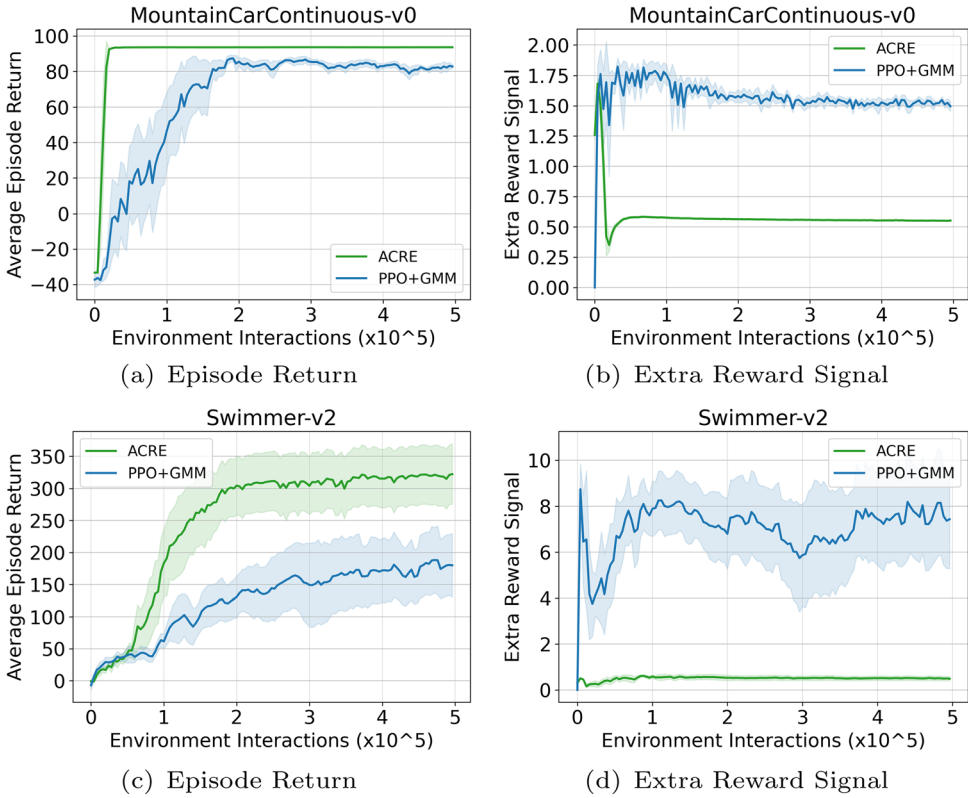


Fig. 2 Performance comparison between ACRE (with GMM) and PPO+GMM for *MountainCarContinuous* & *Swimmer* environments

intrinsic rewards [6, 7, 28]. To constitute a fair comparison, we utilize the same GMM module with all the parameters utilized in ACRE, forming PPO+GMM. As evaluation environments, we utilized the *MountainCarContinuous* and *Swimmer*. Both environments have relatively small state space, allowing quick evaluation of the results. At the same time, quite efficient exploration of the state space is needed; otherwise, the policy networks may early converge to sub-optimal behaviors. All the PPO hyperparameters are chosen according to best practices for these environments. Appendix B extensively covers all hyperparameters' choices for both ACRE and PPO+GMM. Additionally, for PPO+GMM case, a grid search was performed with respect to the factor (β) that weights extrinsic and intrinsic rewards and is presented in Appendix D. Based on that analysis, β is set to 0.05 for *MountainCarContinuous* and 1.0 for *Swimmer*.

Figure 2a, c presents the average episode return for ACRE+GMM and PPO+GMM, for both the environments.

Evidently, ACRE+GMM consistently outperforms PPO+GMM presenting superior performance in integrating the GMM-related intrinsic reward. An important insight can be drawn by examining Fig. 2b, d that depict the average value of the extra GMM-based reward. ACRE novelty integration scheme utilizes the extra reward signal effectively, requiring only a fraction of the magnitude of extra reward that is needed in the best case of PPO+GMM. Figure 3 visualizes the state-space coverage for both environments. Especially for *Swimmer*, a t-SNE transformation [18] is utilized to efficiently project the state space into 2 dimensions. Zooming in on Fig. 3a, we can observe the effective exploration of *MountainCarContinuous* state space of ACRE+GMM compared to PPO+GMM. PPO+GMM exploration (blue dots) seems to follow a normal distribution around the starting state of the car, whereas ACRE+GMM seems to have formed a very specific pattern, that is needed in order to reach the goal position reliably. Switching to Fig. 3b, we can see that the extra states reached by ACRE+GMM have been interpreted as distinct classes in the t-SNE representation.

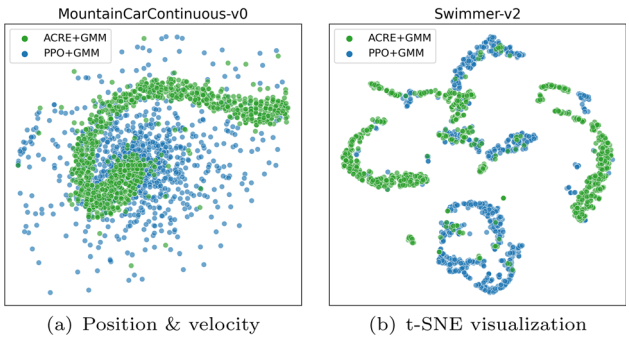
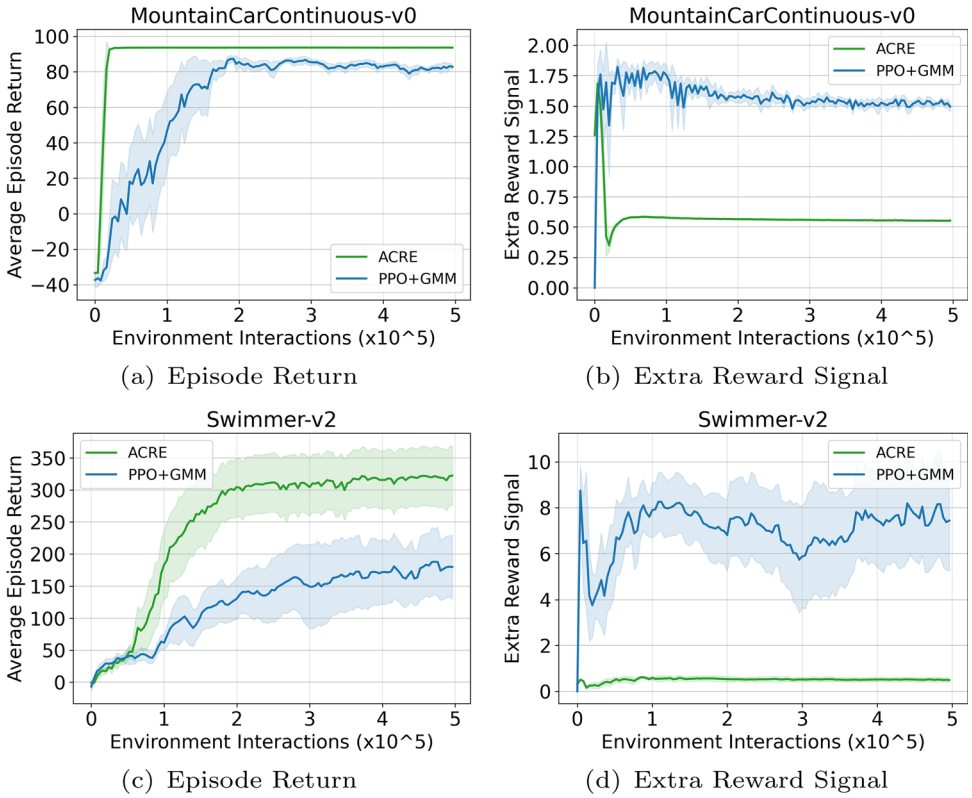


Fig. 3 State-space coverage study between ACRE+GMM and PPO+GMM

4.2 Performance of GMM on novelty estimation

We next evaluate the performance of GMM novelty estimator, as described in (1), with respect to the ability to assist the underlying RL algorithm in escaping local optimal configurations. To do so, we devise a version of ACRE methodology that uses random network distillation (RND) [7] as a novelty estimator. More specifically, instead of deploying a GMM 1 (step 16 of Algorithm 1 in Appendix A), we approximate the novelty as the MSE between RND's *predictor* and *target* networks. Following RND methodology, the *predictor* network is updated to minimize exactly this loss (see Appendix A.2 in [7] for more details). Other than that, all the other functionalities of the ACRE algorithm are intact, as outlined in Appendix A. Intuitively, wider RND networks will be capable of a more fine-grained assessment of the environment's state space at the expense of slower convergence and adaptation to state discovery. To investigate the effect of this novelty reward for different network sizes, we consider 4 versions of this variation with respect to the width of the two hidden layers for the RND networks, i.e., [32, 32], [64, 64], [128, 128], [256, 256] number of neurons, respectively. As in the previous subsection, we are going to evaluate these algorithms on *MountainCarContinuous* and *Swimmer*.

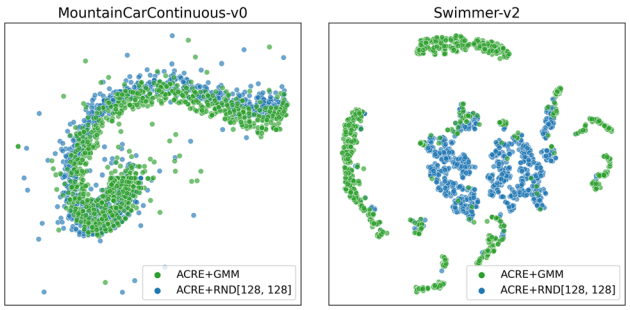
Figure 4 visualizes the results of this study. The ACRE configuration with GMM (green line) consistently achieves the best performance. ACRE+RND[128, 128] seems to match the performance of ACRE+GMM for the *MountainCarContinuous* environment. Overall, by examining both Fig. 4b, d, it is evident that all the RND components quickly lose their "interest" as the predictor network is well-capable of generalizing even for states that have not been seen before. As expected, wider networks for the RND components tend to work better as they can be more detailed on the state-space representation. The pattern seems to break for [256, 256] case (orange line) as the performance degrades seriously, technically misleading the underlying ACRE methodology. Some spikes that appear in the extra reward visualization (Fig. 4b) for ACRE+RND [256, 256] case are probably "too little too late," as both the RL agent is already too confident about its policy and Q-function, and, at the same time, these increased values of the RND component vanish rapidly. Similar to the previous Sect. 4.1, we present in Fig. 5 study of the state-space coverage for ACRE+GMM and the best-performing variant of ACRE+RND, i.e., ACRE+RND[128, 128]. Aligned with the learning curves (green and blue lines) presented in Figs. 4a, 5a depicts an equivalently efficient way of exploring the state space of *MountainCar*. Turning into Fig. 5b, we can see that the performance improvements as reported in Fig. 4c have led to a more effective exploration of the state space, forming spatially disconnected classes of visited states. We devote the last remark of this subsection to pointing out that although GMM has achieved this remarkable performance, its incorporation into an environment with enormous state space (e.g., pixel-based setups) is not trivial. On the contrary, RND-based



(a) Episode Return (b) Extra Reward Signal

(c) Episode Return (d) Extra Reward Signal

Fig. 4 Comparison between ACRE (with GMM) and ACRE+RND for *MountainCarContinuous* & *Swimmer* environments



(a) Position & velocity (b) t-SNE visualization

Fig. 5 State-space coverage study between ACRE+GMM and ACRE+RND[128, 128]

intrinsic reward has achieved human-like performance in several exploration-hard arcade environments [7].

4.3 Extensive analysis on ACRE performance

Finally, we perform extensive testing, evaluating the complete ACRE algorithm on 12 continuous control tasks from the most well-known and used, OpenAI-gym-style collections. We group the evaluation environments into three bundles with four environmental setups each. More precisely, the first bundle contains only standard OpenAI-gym control [5] tasks, namely *BipedalWalker-v3*, *LunarLanderContinuous-v2*, *MountainCarContinuous-v0*, and *Pendulum-v0*. The second bundle focuses on the advanced physics simulator of MuJoCo [38], containing the *Ant-v3*, *Hopper-v3*, *Swimmer-v3* and *Walker2d-v3* environments. The third bundle consists of environments from the DeepMind Control Suite [36], namely *ball_in_cup-catch*, *cartpole-two_poles*, *finger-turn_easy* and *quadruped-walk*.

The performance of ACRE was compared against six (6), well-established RL algorithms: (i) A2C [21], (ii) DDPG, (iii) PPO, iv) SAC, (v) TD3 and (vi) TRPO. The rationale behind choosing such algorithms is to construct a diverse pool of state-of-the-art algorithms, e.g., on-policy/off-policy, stochastic/deterministic policy, etc., to position ACRE accurately and perform a thorough comparison study. The majority of these algorithms have extensions that exploit some kind of parallelism, e.g., DDPG with multiple distributed actors (and other performance improvements) becomes D4PG [3]. However, to construct a comparable pool of algorithms, we focus only on the original algorithms, excluding their parallelism-favored versions, that can also be applied in the proposed ACRE.

To devise a single framework for all the evaluation studies, the recently introduced tonic library [26] was utilized. Also, to avoid contributing to the problem of reporting different evaluation metrics for the same pair of {algorithm, environment}, mainly due to different settings of hyperparameters or implementation details, we run only the simulations related to the proposed ACRE methodology and rely on the already stored learning curves, optimized by the framework designer, for the other state-of-the-art RL approaches. By doing so, we also wanted to remove any ambiguity related to the proper appliance of the aforementioned state-of-the-art RL algorithms. The architectural decisions (e.g., number of layers and neurons inside the neural networks) for ACRE were similar to the alternative state-of-the-art methods (where possible) and were kept constant for all the benchmarks. Figures 6, 7 and 8 report all the evaluated algorithms' learning curves for the 3 bundles, respectively.

Starting from the standard OpenAI-gym environments (Fig. 6), ACRE matches or overcomes the best-achieved value by any of the evaluated state-of-the-art RL algorithms. Additionally, focusing on Fig. 6a, b, we can observe that this performance has been achieved quite rapidly, underlying ACRE's ability to explore the state space quickly. Turning to the original MuJoCo environments (Fig. 7), the proposed ACRE approach once again matches the best-achieved performance. Exceptional performance is achieved in *Swimmer* (Fig. 7c), where ACRE attained almost $\times 3$ the performance of the best performing algorithm (DDPG) for the task at hand. Thanks to the efficient exploration of the available state space, ACRE does not allow both policy and value networks to converge quickly, paving the way for discovering the most efficient policies. Such feature is of

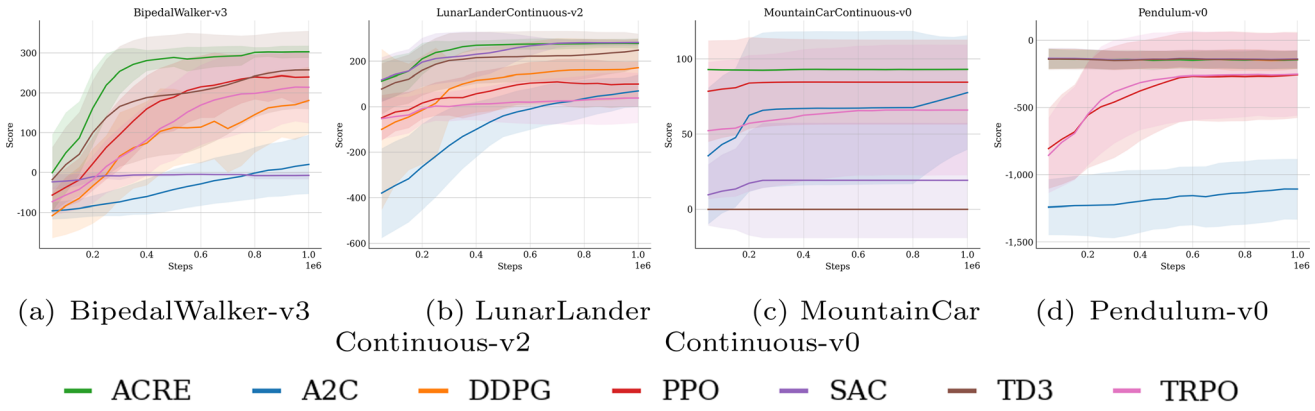


Fig. 6 Standard OpenAI-Gym continuous control environments

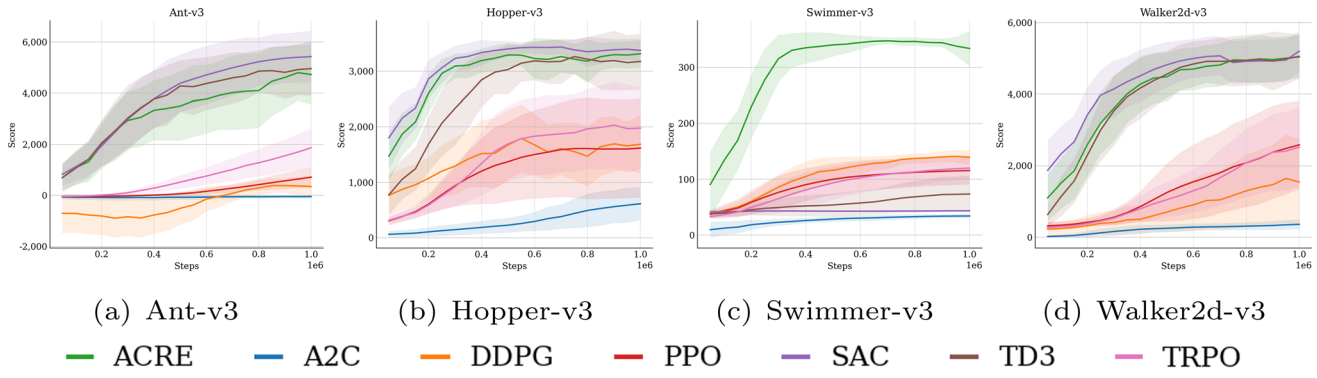


Fig. 7 MuJoCo environments

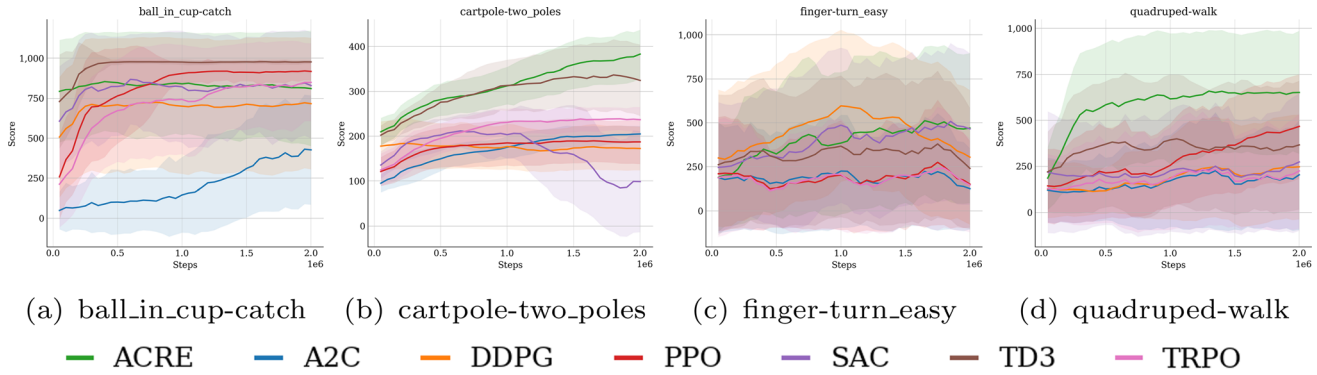


Fig. 8 DeepMind Control Suite

paramount importance, as one of the major issues that cause the current RL algorithm to get stuck in local minima is the learning bias from the initial interactions with the environment [22]. Figure 8 concludes the evaluation study by presenting the results from the DeepMind Control Suite. In the *ball_in_cup-catch* environment, ACRE achieves the average performance, while for *cartpole-two_poles* and *finger-turn_easy* it matches the best-achieved performance. Finally, a strong result is also achieved in the *quadruped-walk* environment, where ACRE achieves an improvement of $\sim 74\%$ over the best-performing algorithm (PPO), even from the first half a million interactions with the environment.

5 Conclusion

This paper studied the integration of extra reward signals in model-free RL algorithms. We started that by observing that a combined tackling of both the problem at hand and the exploration renders the underlying MDP time varying. Directly attempting to establish an optimized policy in such a time-varying MDP can lead to instability and also without any guarantee that the achieved policy is optimized for the problem at hand.

Within this paper we described ACRE algorithm which provides a simple solution to mitigating these harmful effects of extra reward signal integration by having separate handling and storing mechanisms for each signal. On the one hand, the environmental reward is stored persistently in the replay buffer to be ready to be used as many times as needed in its measured form. On the other hand, novelty return is only estimated online during the training procedure to encompass as recent information as possible regarding states' visitation. An ablation simulation study revealed the effectiveness of each component separately. In the follow-up comparison study, ACRE was substantially better than any individual alternative in the simulation trials, exceeding their cumulative returns.

We are only scratching the surface of the many ways to estimate the novelty returns. As a matter of fact, several alternatives in the literature have achieved tremendous performance, e.g., [1], and the majority of them can be directly incorporated into the body of ACRE. A fascinating direction for future work would be to include a size-reduction step on the states, with respect to the novelty return estimation, e.g., a Variational Auto-Encoder (VAE) [16]. Such a size-reduction step could unlock the utilization of the highly accurate GMM-based novelty return estimators in complex control systems with observation vector sizes of several hundred elements (e.g., pixel-based state space).

Appendix A: ACRE Pseudo-code

This appendix section outlines the complete ACRE pseudo-code.

Algorithm 1 Actor-Critic with Reward-preserving Exploration (ACRE)

repeat
  Select action $a \sim \pi_\theta(\cdot\|s)$
  Execute a in the environment
  Observe next state s', reward r, and done signal d
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  if s' is terminal **then** reset environment state
  end if
  if it's time to update **then**
    for i in range(actor-critic updates) **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute targets for the Q functions:

$$y(r, s', d) = r + \gamma(1 - d)\left(\min_{j=1,2} Q_{\phi_{targ,j}}(s', \tilde{a}') - \beta N_u(s')\right),$$

      Update Q-function by one-step of gradient descent for $j = 1, 2$:

$$\nabla_{\phi_i} \frac{1}{\|B\|} \sum_{(s,a,r,s',d)\in B} \left(Q_{\phi_j}(s, a) - y(r, s', d)\right)^2$$

      Update policy by one-step gradient ascent:

$$\nabla_\theta \frac{1}{\|B\|} \sum_{s\in B} \left(\min_{j=1,2} Q_{\phi_{targ,j}}(s, \tilde{a}') - \beta N_u(s)\right),$$

      Update target networks with

$$\phi_{targ,j} \leftarrow \rho\phi_{targ,j} + (1 - \rho)\phi_j, \quad \text{for } j = 1, 2$$

    end for
  end if
  if it's time to update N_u **then**
    Randomly sample a subset of states $\mathcal{D}_e = \{s\}$ from $\mathcal{D}$
    Fit Gaussian Mixture Model $r_e(s)$ to $\mathcal{D}_e$
    Retrieve all states from the last κ episodes and compute exploration rewards-to-go using

$$\hat{R}_e(s_t) = \sum_{t'=t}^{T} \gamma^{t'-t} r_e(s_{t'})$$

    for k in range(exploration updates) **do**
      Update exploration value function N_u by one-step gradient descent:

$$\nabla_u \frac{1}{\|\mathcal{I}\|} \sum_{s\in\mathcal{I}} \left(N_u(s) - \hat{R}_e(s)\right)^2$$

    end for
  end if
until convergence

Appendix B: hyperparameters

See Tables 2 and 3.

Table 2 ACRE hyperparameters

Parameter	Value	Comments
γ	0.99	Discount factor of the MDP
lr	1e-3	Learning rate
Twin Q	True	Use two Q-networks
Q hidden	[256, 256]	Hidden layers size
Policy hidden	[256, 256]	Hidden layers size
Q and Policy models activation	relu	Activation function
Buffer size	1e6	Size of the replay buffer
Batch size	100	Minibatch size for SGD
Actor-critic updates	50	Environment interactions till the next update on Q and π networks
ρ	0.995	Polyak coefficient for target networks update
β	$\propto \|\mathcal{S} \times \mathcal{A}\|^{-1}$	Weights the importance of novelty over the reward returns
κ	1	Number of last episodes the transitions of which are going to be used for the next GMM update
l	7	The number of mixture components
GMM covariance	1e-5	Non-negative regularization added to the diagonal of covariance
Iterations per N_u estimation	80	Number of gradient descent steps to take on value function per novelty estimation

Table 3 PPO+GMM hyperparameters

Parameter	Value	Comments
γ	0.99	Discount factor of the MDP
target_kl	0.01	KL divergence threshold between new and old policies after an update
clip_ratio	0.2	Hyperparameter for clipping in the policy objective
vf_lr	1e-3	Learning rate for value function optimizer
pi_lr	3e-4	Learning rate for policy optimizer
V hidden	[256, 256]	Hidden layers size
π hidden	[256, 256]	Hidden layers size
activation_function	relu	Activation function
train_pi_iters	80	Number of gradient descent steps to take on policy loss per epoch
train_v_iters	80	Number of gradient descent steps to take on value function per epoch
Buffer size	4000	Size of the epoch buffer (=steps_per_epoch)
λ	0.97	Lambda for GAE-Lambda
β	$\propto \|\mathcal{S} \times \mathcal{A}\|^{-1}$	Weights the importance of novelty over the reward returns
κ	1	Number of last episodes the transitions of which are going to be used for the next GMM update
l	7	The number of mixture components
GMM covariance	1e-5	Non-negative regularization added to the diagonal of covariance
Iterations per N_u estimation	80	Number of gradient descent steps to take on value function per novelty estimation

Appendix C: ACRE exploration strategy in accordance with SOTA RL algorithms

Table 4 positions ACRE in accordance with relevant SOTA RL algorithms with respect to the exploration mechanisms both in the environmental interaction and during the learning routines.

Table 4 ACRE positioning with respect to the exploration mechanisms in SOTA RL algorithms

Algorithm	Environment interactions		Learning	
	Action generation	Exploration	Bellman backup update	Exploration
TRPO	$\pi(\cdot\|s)$	Policy stochasticity	–	None
PPO	$\pi(\cdot\|s)$	Policy stochasticity	–	None
DDPG	$\mu(s)+\mathcal{N}_{ou}$	Ornstein-Uhlenbeck process	$r+\gamma Q(s',\mu(s'))$	None
TD3	$\mu(s)+\mathcal{N}(0,\sigma)$	Zero-mean Gaussian noise	$r_t+\gamma Q(s',\tilde{a}'),\ \tilde{a}'=\mu(s')+\mathcal{N}(0,\sigma)$	Zero-mean Gaussian noise
SAC	$\pi(\cdot\|s)$	Policy stochasticity	$r+\gamma\left(Q(s',\tilde{a}')-\alpha\log\pi(\tilde{a}'\|s')\right),\ \tilde{a}'\sim\pi(\cdot\|s')$	Policy entropy
ACRE	$\boldsymbol{\pi}(\cdot\|\boldsymbol{s})$	Policy stochasticity	$\boldsymbol{r}+\boldsymbol{\gamma}\left(\boldsymbol{Q}(\boldsymbol{s}',\boldsymbol{a}')-\boldsymbol{\beta}\boldsymbol{N}_{\boldsymbol{u}}(\boldsymbol{s}')\right),\ \boldsymbol{a}'\sim\boldsymbol{\pi}(\cdot\|\boldsymbol{s}')$, $\nabla_{\boldsymbol{u}}\frac{1}{\|\mathcal{I}\|}\sum_{\boldsymbol{s}\in\mathcal{I}}\left(\boldsymbol{N}_{\boldsymbol{u}}(\boldsymbol{s})-\hat{\boldsymbol{R}}_{\boldsymbol{e}}(\boldsymbol{s})\right)^2$, $\boldsymbol{R}_{\boldsymbol{e}}(\boldsymbol{s})=\sum_{t'=t}^{T}\boldsymbol{\gamma}^{t'-t}\boldsymbol{r}_{\boldsymbol{e}}(\boldsymbol{s}')$, $\boldsymbol{r}_{\boldsymbol{e}}(\boldsymbol{s}')\sim$ ***Gaussian mixture model***	Forward novelty estimated by probability density function of a mixture of Gaussians

Appendix D: grid search on the weighting factor of PPO with GMM

Figure 9 presents a grid search on the weighting factor of PPO+GMM for *MountainCarContinuous-v0* and *Swimmer-v2* environments. The weights that achieved the best value is set to 0.05 for *MountainCarContinuous-v0* and 1.0 for *Swimmer-v2* and are going to be used in all the experimental evaluation of Sect. 4.1.

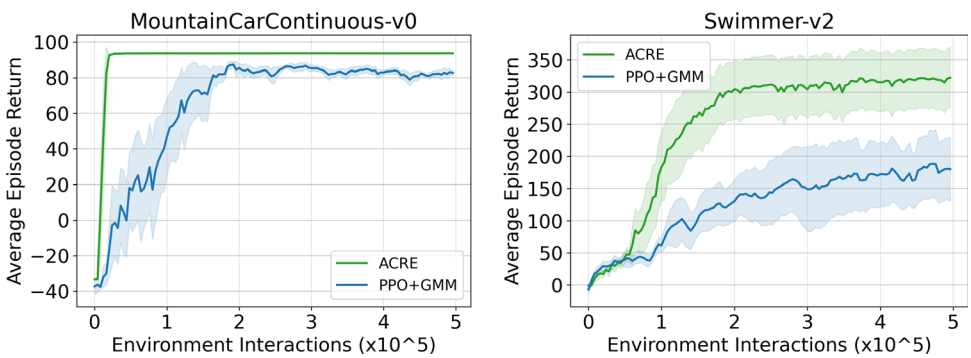


Fig. 9 Grid search on the weighting factor of PPO+GMM

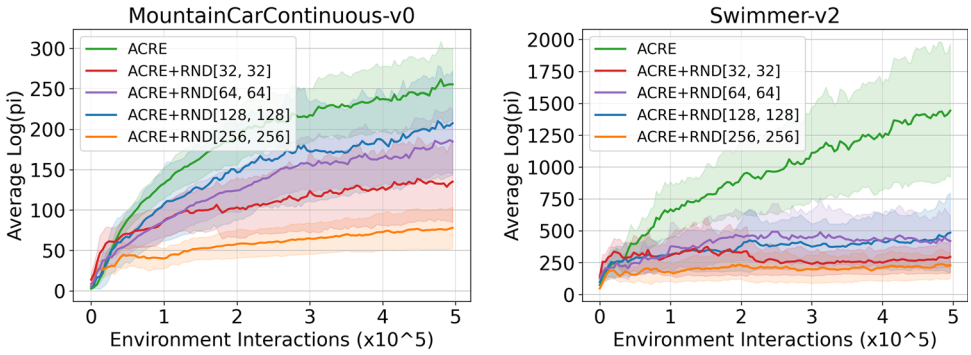


Fig. 10 Comparison between ACRE (with GMM) and ACRE+RND in terms of $\log(\pi)$ evolution for *MountainCarContinuous* & *Swimmer* environments

Appendix E: effect of GMM on the log (π) evolution

Additionally to the analysis presented in Sect. 4.2, an important conclusion can be also drawn by studying the evolution of $\log(\pi)$ for all the evaluating algorithms on both *MountainCarContinuous-v0* & *Swimmer-v2*, as presented in Fig. 10.

Although ACRE with GMM seems to constantly have an average intrinsic reward above 0 (almost like a DC component), the average $\log(\pi)$ is greater, meaning that the RL agent is more certain about its choices.

Acknowledgements We would like to thank NVIDIA Corporation for donating GPUs for this research.

Funding Open access funding provided by HEAL-Link Greece. This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH-CREATE-INNOVATE (Project Code:T2EDK-02,743).

Data availability The training/evaluation phase was performed using the following open-source RL framework https://github.com/fabiopardo/tonic using all the pre-stored hyperparameter values. A standalone implementation of the proposed ACRE algorithm is available on GitHub in the following repository https://github.com/athakapo/ACRE.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Badia AP, Sprechmann P, Vitvitskyi A et al. (2019) Never give up: learning directed exploration strategies. In: International conference on learning representations
2. Badia AP, Piot B, Kapturowski S et al. (2020) Agent57: outperforming the atari human benchmark. In: International conference on machine learning. PMLR, pp 507–517
3. Barth-Maron G, Hoffman MW, Budden D et al. (2018) Distributed distributional deterministic policy gradients. In: International conference on learning representations
4. Bellemare MG, Naddaf Y, Veness J et al (2013) The arcade learning environment: an evaluation platform for general agents. J Artif Intell Res 47:253–279
5. Brockman G, Cheung V, Pettersson L et al. (2016) Openai gym. arXiv:1606.01540
6. Burda Y, Edwards H, Pathak D et al. (2018) Large-scale study of curiosity-driven learning. In: International conference on learning representations
7. Burda Y, Edwards H, Storkey A et al. (2019) Exploration by random network distillation. In: Seventh international conference on learning representations, pp 1–17
8. Dulac-Arnold G, Levine N, Mankowitz DJ et al (2021) Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Mach Learn 110:2419–2468
9. Fujimoto S, Hoof H, Meger D (2018) Addressing function approximation error in actor-critic methods. In: International conference on machine learning. PMLR, pp 1587–1596
10. Gu S, Holly E, Lillicrap T et al. (2017) Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE international conference on robotics and automation (ICRA). IEEE, pp 3389–3396
11. Haarnoja T, Tang H, Abbeel P et al. (2017) Reinforcement learning with deep energy-based policies. In: International conference on machine learning. PMLR, pp 1352–1361
12. Haarnoja T, Zhou A, Abbeel P et al. (2018) Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. PMLR, pp 1861–1870
13. Kahn G, Abbeel P, Levine S (2021) Badgr: an autonomous self-supervised learning-based navigation system. IEEE Robot Autom Lett 6(2):1312–1319
14. Karatzinis GD, Michailidis P, Michailidis IT et al (2022) Coordinating heterogeneous mobile sensing platforms for effectively monitoring a dispersed gas plume. Integr Comput-Aided Eng 29(4):411–429. https://doi.org/10.3233/ICA-220690
15. Khorasgani H, Wang H, Gupta C et al. (2021) Deep reinforcement learning with adjustments. In: 2021 IEEE 19th international conference on industrial informatics (INDIN). IEEE, pp 1–8

16. Kingma DP, Welling M (2014) Auto-encoding variational bayes. In: International conference on learning representations
17. Lillicrap TP, Hunt JJ, Pritzel A et al. (2016) Continuous control with deep reinforcement learning. In: ICLR (Poster)
18. Van der Maaten L, Hinton G (2008) Visualizing data using t-sne. J Mach Learn Res 9:2579–2605
19. Mnih V, Kavukcuoglu K, Silver D et al. (2013) Playing atari with deep reinforcement learning. In: NIPS deep learning workshop
20. Mnih V, Kavukcuoglu K, Silver D et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
21. Mnih V, Badia AP, Mirza M et al. (2016) Asynchronous methods for deep reinforcement learning. In: International conference on machine learning. PMLR, pp 1928–1937
22. Nikishin E, Schwarzer M, D'Oro P et al. (2022) The primacy bias in deep reinforcement learning. In: International conference on machine learning. PMLR, pp 16,828–16,847
23. Ornik M, Topcu U (2021) Learning and planning for time-varying mdps using maximum likelihood estimation. J Mach Learn Res 22:35–1
24. Ostrovski G, Bellemare MG, Oord A et al. (2017) Count-based exploration with neural density models. In: International conference on machine learning. PMLR, pp 2721–2730
25. Ouyang L, Wu J, Jiang X et al (2022) Training language models to follow instructions with human feedback. Adv Neural Inf Process Syst 35:27730–27744
26. Pardo F (2020) Tonic: a deep reinforcement learning library for fast prototyping and benchmarking. arXiv:2011.07537
27. Pathak D, Agrawal P, Efros AA et al. (2017) Curiosity-driven exploration by self-supervised prediction. In: International conference on machine learning. PMLR, pp 2778–2787
28. Pathak D, Gandhi D, Gupta A (2019) Self-supervised exploration via disagreement. In: International conference on machine learning. PMLR, pp 5062–5071
29. Schulman J, Levine S, Abbeel P et al. (2015) Trust region policy optimization. In: International conference on machine learning. PMLR, pp 1889–1897
30. Schulman J, Wolski F, Dhariwal P et al. (2017) Proximal policy optimization algorithms. arXiv:1707.06347
31. Silver D, Huang A, Maddison CJ et al (2016) Mastering the game of go with deep neural networks and tree search. Nature 529(7587):484–489
32. Smith L, Kew JC, Peng XB et al. (2021) Legged robots that keep on learning: fine-tuning locomotion policies in the real world. arXiv:2110.05457
33. Smith L, Kostrikov I, Levine S (2022) A walk in the park: learning to walk in 20 minutes with model-free reinforcement learning. arXiv:2208.07860
34. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT Press, Cambridge
35. Tang H, Houthooft R, Foote D et al. (2017) # exploration: a study of count-based exploration for deep reinforcement learning. In: 31st conference on neural information processing systems (NIPS), pp 1–18
36. Tassa Y, Doron Y, Muldal A et al. (2018) Deepmind control suite. arXiv:1801.00690
37. Thrun S, Schwartz A (1993) Issues in using function approximation for reinforcement learning. In: Proceedings of the fourth connectionist models summer school, Hillsdale, NJ, pp 255–263
38. Todorov E, Erez T, Tassa Y (2012) Mujoco: a physics engine for model-based control. In: 2012 IEEE/RSJ international conference on intelligent robots and systems. IEEE, pp 5026–5033
39. Wu P, Escontrela A, Hafner D et al. (2023) Daydreamer: world models for physical robot learning. In: Conference on robot learning. PMLR, pp 2226–2240