Learning to Explore in Diverse Reward Settings via Temporal-Difference-Error Maximization

Sebastian Griesbach[1], **Carlo D'Eramo**[1,2,3]
[1] Center for Artificial Intelligence and Data Science, University of Würzburg
[2] Department of Computer Science, Technical University of Darmstadt
[3] Hessian Center for Artificial Intelligence (Hessian.ai), Germany

Abstract

Numerous heuristics and advanced approaches have been proposed for exploration in different settings for deep reinforcement learning. Noise-based exploration generally fares well with dense-shaped rewards and bonus-based exploration with sparse rewards. However, these methods usually require additional tuning to deal with undesirable reward settings by adjusting hyperparameters and noise distributions. Rewards that actively discourage exploration, i.e., with an action cost and no other dense signal to follow, can pose a major challenge. We propose a novel exploration method, Stable Error-seeking Exploration (SEE)[1], that is robust across dense, sparse, and exploration-adverse reward settings. To this endeavor, we revisit the idea of maximizing the TD-error as a separate objective. Our method introduces three design choices to mitigate instability caused by far-off-policy learning, the conflict of interest of maximizing the cumulative TD-error in an episodic setting, and the non-stationary nature of TD-errors. SEE can be combined with off-policy algorithms without modifying the optimization pipeline of the original objective. In our experimental analysis, we show that a Soft-Actor Critic agent with the addition of SEE performs robustly across three diverse reward settings in a variety of tasks without hyperparameter adjustments.

1 Introduction

Exploration is widely recognized as a crucial and distinctive aspect of reinforcement learning (RL). Being such a fundamental problem, a plethora of theoretical studies and empirical works have been produced over the years. Practical solutions to tackle exploration are numerous, with a prevalence of methods based on the injection of stochastic noise into policies either directly [11, 10, 16, 6] or via entropy maximization [24, 14]. Noise-based exploration has proven to be highly effective in settings where the reward function is informative and well-shaped. For less favorable reward settings, e.g., sparse rewards, typically more targeted methods are employed. For example, bonus-based methods, also referred to as intrinsic motivation, are a class of reward-shaping mechanisms that aim to substitute sparse rewards such that the reward signal is transformed back to a shaped reward scheme again [2, 3, 27]. Such methods often substitute the reward but still rely entirely on random noise for the actual action selection for exploration. The reshaped rewards can potentially dilute the original reward function of the MDP, changing the optimal policy or causing over-exploration of irrelevant state regions, which may harm performance in other reward settings [22]. If in a sparse reward setting one tries to enforce efficient behavior, it is reasonable to apply action costs. The resulting reward may actively discourage exploration as every action that does not immediately reach the goal is punished, adding complication to this setting. Adapting the exploration of a given RL algorithm from one setting to another often demands a costly tuning procedure, which might be

[1]Code to reproduce all experiments is available at: `https://github.com/Sebastian-Griesbach/SEE`

infeasible in complex problems. Therefore, making RL algorithms more robustly applicable is an ongoing effort of the research community.

To this endeavor, we propose Stable Error-seeking Exploration (SEE), a novel approach for exploration that robustly handles diverse reward settings. SEE is a directed action selection mechanism that can be combined with any off-policy RL algorithm and does not introduce relevant additional hyperparameters. Our approach separates exploration from exploitation by optimizing for two disentangled objectives and training two individual policies. On the one hand, the *exploitation* objective remains the unchanged RL objective of maximizing the (discounted) cumulative reward, and, thus, it can be combined with any existing RL optimization pipeline, including reward shaping mechanisms. On the other hand, our *exploration* objective maximizes the absolute temporal-difference-error (TD-error) encountered during training rollouts. The intuition behind this choice is that a high TD-error, especially in deterministic environments, indicates the presence of potentially relevant information not yet learned. We point out that decoupling exploration and exploitation into two separate objectives is not a new idea, as shown in prior works [26, 19, 15]. However, till now, it has been challenging to devise effective and stable learning procedures when maximizing the TD-error. In this paper, we identify three distinct causes of instability affecting this setting, namely (i.) far off-policy learning, when a behavior policy is too out of distribution w.r.t. the target policy; (ii.) the conflict of interest of terminating an episode while the exploration objective follows an always positive reward signal; and (iii.) the non-stationary nature of the TD-error as a target. The **contribution** of this work is the formulation of a methodological solution to tackle each of these issues, i.e., (1.) combining the exploration and exploitation policies into a single behavior policy that bridges both distributions; (2.) using a maximum reward update, which is agnostic towards the length of an episode; (3.) conditioning the exploration value function on current estimates of the exploitation value function to inform it about the cause of change in the TD-error target. We integrate these solutions into one algorithmic approach, resulting in SEE.

We empirically analyze SEE by combining it with Soft Actor-Critic (SAC) [24, 8] and TD3 [16]. We show that incorporating SEE enhances robustness across diverse reward settings, without the need for hyperparameter tuning. Especially SAC+SEE shows strong performance across all tested environments. Furthermore, we conduct ablation studies to analytically evince the positive effect of our proposed design choices.

2 Preliminaries

We consider a reinforcement learning (RL) setting [21] in which the environment is modeled as a Markov decision process (MDP), defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{P}(s'|s,a)$ the state transition function, $\mathcal{R}(s,a,s')$ the reward function, and $\gamma \in [0,1)$ the discount factor. A stochastic policy $\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ defines $\pi_\theta(a|s)$, the probability of taking action a in state s. The objective is to maximize the expected return $J(\pi_\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where the expectation is over trajectories induced by π_θ. In the actor-critic framework, the actor represents the policy, while the critic estimates either the state-value function $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s]$ or the action-value function $Q^\pi(s,a) = \mathbb{E}_\pi[r(s,a) + \gamma Q^\pi(s',a') | s,a]$. The critic is typically updated by minimizing the temporal-difference (TD) error $\delta = r + \gamma Q^\pi(s',a') - Q^\pi(s,a)$, while the actor is updated via the policy gradient $\nabla_\theta J(\pi_\theta) \approx \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)]$. This framework leads to deep RL algorithms such as TD3 [16] and SAC [24], which we are using as the basis of our work.

3 Stable Error-seeking Exploration

We propose an off-policy exploration method that performs robustly in different reward scenarios without hyperparameter adjustments. To this end, we use separate objectives for *exploitation* and *exploration*. On the one hand, the *exploitation* objective remains the regular cumulative discounted reward. On the other hand, the *exploration* objective aims to maximize the absolute value of the encountered TD-error of the *exploitation* objective throughout training rollouts. This choice entails using two separate policies, one for each objective. It is worth noting that prior works have already established this separation of objectives [26, 19, 15]. However, despite its simple formulation, the exploration objective is challenging to optimize due to what we identify to be three main causes of instability in the learning process. In the following, we describe in detail these causes of instability and propose targeted solutions for each of them, eventually devising an algorithmic solution that mitigates them and enables stable and effective learning.

3.1 Mixing of policies

In common off-policy deep RL approaches, the behavior policy used during rollouts is chosen to be similar to the learned target policy. For example, it can be a noisy and past variant of the target policy, e.g., when using ε-greedy exploration with a replay buffer. Ideally, it would suffice to use an arbitrary exploration policy during training rollouts to gather relevant information about the environment. In practice, it has been shown that off-policy deep RL techniques perform poorly if the behavior policy is too out-of-distribution w.r.t. the target policy [17]. Importantly, by pursuing different objectives, our exploration and exploitation policies can be arbitrarily far apart, thus creating instability. To tackle this issue, we propose to combine both exploration and exploitation policies into a single *behavior* policy μ which is used during training rollouts. Assume two arbitrary policies π_1, π_2 and their respective action-value functions $Q_1(s,a), Q_2(s,a)$. Let the two actions given by the policies in state s be denoted as

$$a_1 \sim \pi_1\left(\cdot|s\right) \qquad\qquad a_2 \sim \pi_2\left(\cdot|s\right). \tag{1}$$

The behavior policy μ selects one of the candidate actions under a Boltzmann distribution

$$\forall a \in \{a_1, a_2\} : \mu(a|s) \propto \exp\left(A\left(s,a\right)\right), \tag{2}$$

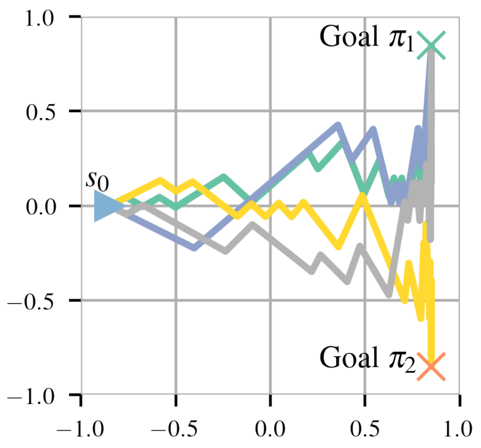
where A is the relative advantage of both actions w.r.t. their action-value functions defined as

$$A(s,a_1) = Q_1(s,a_1) - Q_1(s,a_2) \qquad\qquad A(s,a_2) = Q_2(s,a_2) - Q_2(s,a_1). \tag{3}$$

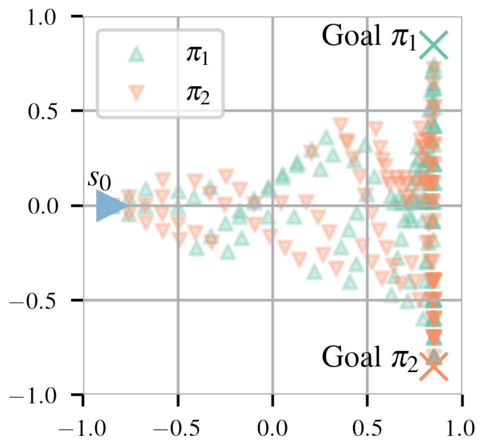
Using a Boltzmann distribution enables the mixing of both policies in regions where their expected values are close while focusing on one particular policy when a significantly higher value is expected. Intuitively, it explores the use of both policies and gravitates towards "goals" of any of the two. We found that a temperature of $\tau = 1$ works reliably and therefore do not consider it as a relevant hyperparameter.
In general, the scale of Q_1 and Q_2 needs to be considered to achieve a balanced mixing. However, in this case, the two relative advantages are estimates of 1.) the regular advantage following a specific action which is defined as the temporal-difference (not the error) and 2.) the temporal-difference-error of the same transition. As both values are based on the same reward function, they have a similar magnitude and thus no balancing is required. Figure 1 shows an example of a mixed policy. The data generated by the behavior policy μ creates a bridge between the two policy distributions.

3.2 Maximum reward formulation

Typical RL approaches optimize the cumulated discounted future reward. Differently, our exploration objective optimizes the absolute value of the TD-error. This introduces a conflict of interest between the exploitation and exploration policies, especially in problems where the agent is tasked to reach a terminal state as soon as possible. The TD-error of the transition to a terminal state is likely large if the agent has not yet learned to reliably seek it. However, in the remaining state-space there might be many small TD-errors. When optimizing for the accumulated absolute TD-error, it might be suboptimal to reach the terminal state as no further rewards can be gathered thereafter. This is the same problem described by [3], who tackle it by making the exploration reward non-episodic, such that the accumulation considers the future beyond an episode termination. However, this formulation discounts the transition of interest based on its distance to the initial state, thus neglecting potentially



(a) Four sampled trajectories of a mixed policy.



(b) Distribution of states where an action of the two underlying policies is selected during the rollout.

Figure 1: Example of a mixed behavior policy of two deterministic policies π_1, π_2 going towards their respective goals. An action moves a maximum distance of 0.05 on a plane of the size of 2×2. The respective action-value functions assume a reward of 1 and termination at their target position, 0 elsewhere. We use a discount factor of 0.9.

relevant transitions that are far away from the initial state.
On the contrary, we propose to use a formulation that is agnostic to the length of the episode. A first viable option would be to use an average reward formulation, as it circumvents the discounting issue altogether. Although we think this would be the most natural choice, it is not clear how to estimate the average reward of the current policy purely from off-policy data. Therefore, we opt for the maximum reward formulation where the single highest discounted reward along a trajectory is maximized [7, 25]. This simple to realize and agnostic towards the length of the episode. Thus, the Bellman update of the exploration objective is replaced with the maximum reward update:

$$Q^{k+1}(s,a) = \max\left(r, \gamma \max_{a'} Q^k(s',a')\right), \tag{4}$$

where r is the reward received during the transition from state s with action a to state s', γ is the discount factor, a' is the next action of the underlying policy, Q^k is the state-action value function and Q^{k+1} its next iteration after the update. Intuitively, this means that the exploration agent is seeking the single largest misconception in the value function approximation along a trajectory.

3.3 Conditioning of value function

During training, the exploitation action-value approximation changes, causing in turn the TD-error to change and thus the reward function of the exploration objective. Non-stationary rewards might destabilize training as the agent has no information on why, how, and when the reward function changes, and it is limited to adapting to the distributional shift as it appears. We propose to inform the exploration objective about the cause of changes. To achieve this, we add an embedding of the *exploitation* value function parameters to the arguments of the *exploration* value function.
To obtain such embedding, we resort to fingerprinting, a method originally developed by [9], which has been later shown to be effective in deep RL settings [4, 5]. Here, we adopt fingerprinting as follows. Let there be two action-value functions Q^θ and Q^ω where θ and ω are the parameters of the respective value functions. The goal is to condition Q^ω such that it takes Q^θ into account $Q^\omega(s, a|Q^\theta)$. A number of probe states $\hat{s} \in \mathcal{S}$ and probe action $\hat{a} \in \mathcal{A}$ are randomly initialized. The embedding is simply the concatenated result of passing the probe states and actions through Q^θ:

$$\phi(\theta) = (Q^\theta(\hat{s}_1, \hat{a}_1), Q^\theta(\hat{s}_2, \hat{a}_2), \cdots, Q^\theta(\hat{s}_n, \hat{a}_n)), \tag{5}$$

where $\phi(\theta)$ is a vector that embeds Q^θ. The embedding is then given as an additional input to Q^ω. Figure 2 shows a conceptual visualization of fingerprinting. As the embedding process is fully

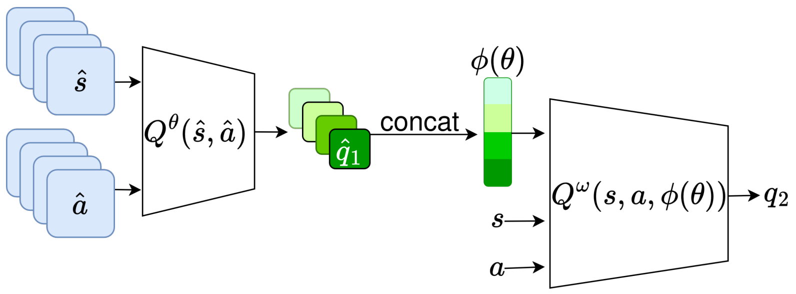


Figure 2: Fingerprinting ϕ is used to condition a value function Q^{ω} on another value function Q^{θ}.

differentiable, the probe states $\hat{s}$ and actions $\hat{a}$ can be optimized based on the objective along with all other parameters. For ease of notation, the probe states $\hat{s}$ and actions $\hat{a}$ are from here on considered to be part of the conditioned value function parameters $\hat{s}, \hat{a} \in \omega$. Thus, θ becomes directly an additional input $Q^{\omega}(s, a, \theta)$. For our method, we use fingerprinting to condition the exploration value function on the exploitation value function to inform it about the cause of change in the TD-errors. The number of used probe states and actions is an additional hyperparameter. However, [5] has shown that as little as 10 probe-states are sufficient to inform a critic about an actor network to solve MuJoCo environments. We do not tune for this hyperparameter but simply pick a slightly higher number of 16 for all tasks.

3.4 Exploration objective

Following the three design choices described above, we can formulate our *exploration* objective, which solves the MDP of the *exploitation* objective with two modifications:

- The parameters of the exploitation network $\theta \in \Theta$ are included in the state space (conditioning):
$$\mathcal{S}_{\Delta} = \mathcal{S} \cup \Theta. \tag{6}$$
- The exploration reward function $\mathcal{R}_{\Delta}(s, a, s', \theta)$ is defined as the absolute TD-error of the exploitation action-value estimation:
$$\mathcal{R}_{\Delta}(s, a, s', \theta) = \left| \mathcal{R}(s, a, s') + \gamma \max_{a'} \hat{Q}^{\theta}(s', a') - \hat{Q}^{\theta}(s, a) \right|, \tag{7}$$
  where $\mathcal{R}(s, a, s')$ is the reward function of the exploitation MDP and $\hat{Q}^{\theta}$ is the approximation of the exploitation state-action value function with parameters θ.

We denote the state-action value function approximation of the *exploration* objective with parameters ω as $\hat{\Delta}^{\omega}(s, a, \theta)$. The fingerprinting probe inputs are contained in ω. Using the maximum update formulation and conditioning, we obtain the *exploration* objective

$$\mathcal{J}_{\omega} = \mathbb{E}_{s,a,s' \sim \mathcal{D}} \left[\left(\max \left(\mathcal{R}_{\Delta}(s, a, s', \theta), \gamma \max_{a'} \hat{\Delta}^{\omega'}(s', a', \theta) \right) - \hat{\Delta}^{\omega}(s, a, \theta) \right)^2 \right], \tag{8}$$

where $\mathcal{D}$ is the transition replay buffer. During training rollouts, a mixed behavior policy μ samples actions to collect data, as described in Section 3.1.
The *exploitation* objective remains the unchanged objective of the underlying base RL algorithm. It is only affected by the gathered data from the replay buffer during updates. Algorithm 1 shows how SEE can be combined with an arbitrary off-policy actor-critic RL method. For the sake of clarity, some details like target networks are omitted. We refer to Appendix D for implementation details.

Algorithm 1 Generalized actor-critic + SEE

```
Input: Regular inputs for actor-critic algorithm: $\hat{Q}^{\theta_1} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}, \hat{\pi}_Q^{\theta_2} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d, \theta_1 \in \Theta$;
Exploration critic and actor $\hat{\Delta}^{\omega_1} : \mathcal{S} \times \mathcal{A} \times \Theta \to \mathbb{R}, \hat{\pi}_\Delta^{\omega_2} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$;
Empty replay buffer $\mathcal{D}$;
for each iteration do
  for each environment interaction do
    $a_Q \sim \hat{\pi}_Q^{\theta_2}(\cdot|s), a_\Delta \sim \hat{\pi}_\Delta^{\omega_2}(\cdot|s)$    ▷ sample candidate actions
    $A(s, a_Q) \leftarrow \hat{Q}^{\theta_1}(s, a_Q) - \hat{Q}^{\theta_1}(s, a_\Delta)$    ▷ calculate relative advantages
    $A(s, a_\Delta) \leftarrow \hat{\Delta}^{\omega_1}(s, a_\Delta, \theta_1) - \hat{\Delta}^{\omega_1}(s, a_Q, \theta_1)$
    $a \sim \mu(a|s) \propto \exp(A(s,a)), \forall a \in \{a_Q, a_\Delta\}$    ▷ sample action from behavior policy
    $s' \sim p(\cdot|s,a), \mathcal{D} \leftarrow \mathcal{D} \cup \{s, a, r, s'\}$    ▷ step and record transition
  for each gradient step do
    $s, a, r, s' \sim \mathcal{D}$    ▷ sample mini-batch
    $\theta_1 \leftarrow \arg\min_{\theta_1} \mathcal{J}_{\theta_1}, \theta_2 \leftarrow \arg\min_{\theta_2} \mathcal{J}_{\theta_2}$    ▷ unmodified off-policy actor-critic update
    $a'_Q \sim \hat{\pi}_Q^{\theta_2}(\cdot|s'), a'_\Delta \sim \hat{\pi}_\Delta^{\omega_2}(\cdot|s')$    ▷ sample next actions
    $r_\Delta \leftarrow \left| r + \gamma \hat{Q}^{\theta_1}(s', a'_Q) - \hat{Q}^{\theta_1}(s, a) \right|$    ▷ calculate exploration rewards
    $\omega_1 \leftarrow \arg\min_{\omega_1} \left( \max\left( r_\Delta, \gamma \hat{\Delta}^{\omega_1}(s', a'_\Delta, \theta_1) \right) - \hat{\Delta}^{\omega_1}(s, a, \theta_1) \right)^2$    ▷ maximum update
    $\omega_2 \leftarrow \arg\min_{\omega_2} \left( -\hat{\Delta}^{\omega_1}(s, a_\Delta \sim \hat{\pi}_\Delta^{\omega_2}(\cdot|s), \theta_1) \right)$    ▷ update exploration actor
Output: $\hat{Q}^{\theta_1}, \hat{\pi}_Q^{\theta_2}, \hat{\Delta}^{\omega_1}, \hat{\pi}_\Delta^{\omega_2}$
```

4 Experiments

We empirically validate our approach by analyzing its effect when used in off-policy deep RL algorithms. We select Twin-Delayed Deep Deterministic Policy Gradient (TD3) [16] and Soft Actor-Critic (SAC) [24, 8] as base algorithms. All additions to the regular update of these base algorithms, e.g., two critic networks and entropy maximization for SAC, are also used for the exploration objective update. This means that for SAC, both the exploration and exploitation policies perform separate entropy maximization for their respective objectives. In TD3, a static noise is added on top of the policy for exploration purposes. This static nature of the exploration noise does not align well with the idea of dynamical exploration of SEE. Therefore, we opted to use no noise in the TD3+SEE implementation, making the action selection of the exploration and exploitation policy deterministic. However, the behavior policy stochastically selects one of the two candidate actions. Further implementation details are described in Appendix D. We opt to not compare to any other methods due to the lack of comparability. We are not aware of other works that focus on robustness across different reward settings in deep RL. The most related class of algorithms is intrinsic motivation, which usually substitutes the reward with novelty bonuses for very sparse rewards, for which we do not make any claims. In all used environment variants, it is possible to occasionally find a positive reward with uniform random actions. Furthermore, our method can be combined with such reward-shaping methods, as it is agnostic to the exploitation optimization pipeline and its reward function. SEE only affects the action selection during training rollouts; thus, it is not competing with methods that augment the reward or the optimization pipeline. To our knowledge, the approach of [19] is the closest to our work. However, a comparison is not sensible as they employ an additional method that affects the optimization pipeline to stabilize their approach [18].

4.1 Comparison to base algorithms

For the sake of conciseness, Figure 3 highlights results of four of the total eight environments. Remaining results are shown in Figure 6. We show that SEE performs robustly in diverse reward settings by creating three reward variants per environment. The three reward settings are dense rewards, sparse rewards, and exploration-adverse rewards. In the following, we describe what exactly these settings entail in our experimental setup.

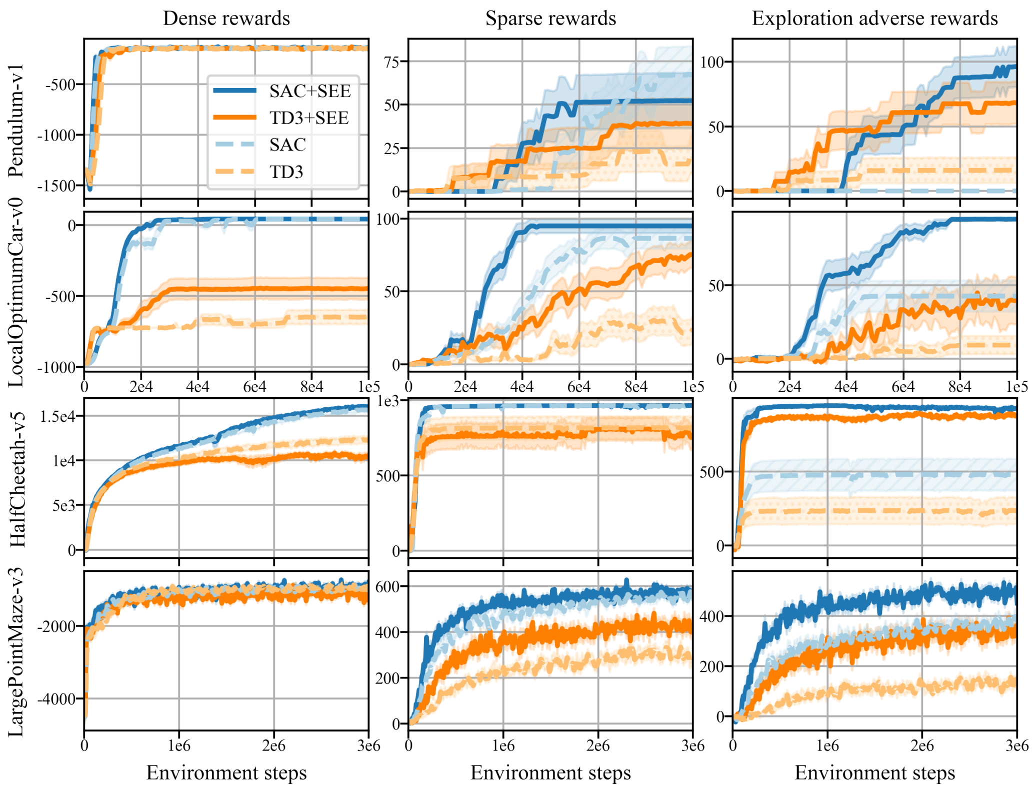


Figure 3: Comparing SAC+SEE and TD3+SEE to their respective base algorithms across multiple environments in different reward settings. The plots show the average evaluation return across 20 seeds per environment variant. The shaded regions indicate the standard error.

- **Dense rewards**: The reward function is well-shaped. Every step towards the desired behavior is reflected in the reward signal.
- **Sparse rewards**: There is a goal that must be reached. Only when the goal is reached does the reward become a positive value; otherwise, it is 0. All goals in our environments can be occasionally reached by uniform random actions or random environment initialization.
- **Exploration adverse rewards**: There is a goal that must be reached. The reward function actively discourages exploration by applying an action cost to every action. A positive reward is only given upon reaching the goal. Inactivity is always a local optimum.

We select eight environments of varying difficulty from Pendulum-v1 which is quickly solved by most RL algorithms, to FetchPickAndPlace-v4 which was originally developed for Hindsight Experience Replay [1] and is quite challenging to solve without it. For each of these environments, we created three variants of the previously established reward settings. Detailed descriptions of the environments are listed in Appendix B. For our experimental results, we run the two base algorithms and their respective SEE extension on all 24 environment variants. We do not change hyperparameters in between the reward settings. Notably, the hyperparameters have **not** been tuned for our extension. Instead, we use the pre-tuned hyperparameters of the base algorithms in their respective extensions. Appendix C contains a list of all used hyperparameters. Generally, it can be observed that in the dense reward setting, the addition of SEE does perform comparably to their base algorithm. In the sparse reward setting, SEE seems to have some advantage, and in the exploration adverse setting, SEE improves performance significantly. SAC performs surprisingly well in the sparse reward setting and even has a slightly higher final performance on sparse reward Pendulum-v1. However, in the adverse reward Pendulum-v1 not a single run of SAC was capable of reaching the goal. Especially SAC+SEE performs robustly across all environments and reward settings. TD3+SEE generally improves the performance of TD3 with some exceptions.

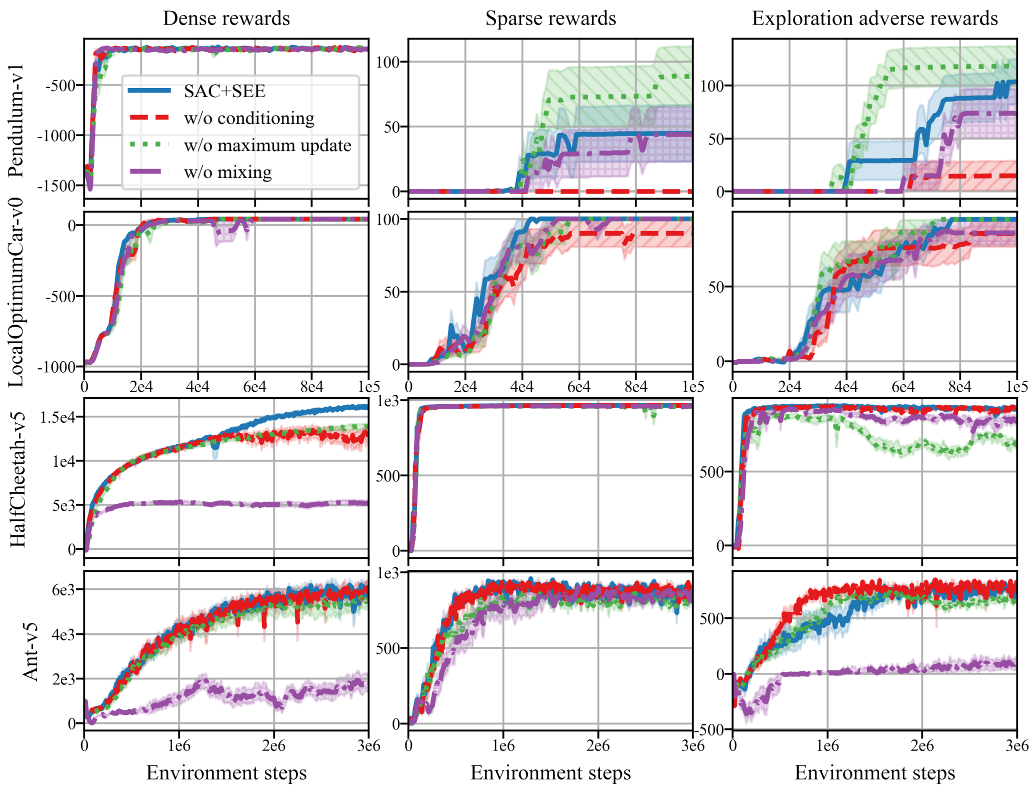


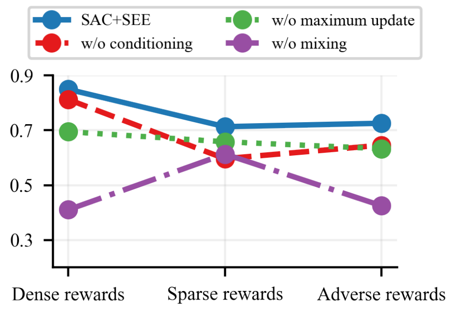
Figure 4: Comparing SAC+SEE to ablations where one of the design choices is replaced. The graphs show the average evaluation return across 10 seeds per environment variant. The shaded regions indicate the standard error.

4.2 Ablations

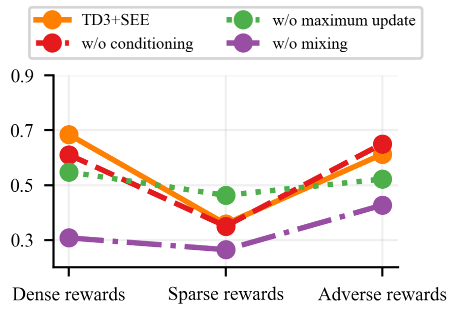
To measure the impact of our three design choices, we conducted ablation studies on a subset of environments. To avoid cluttering, we highlight in Figure 4 the ablation studies of combining SEE with SAC. The results of the same ablation but with TD3+SEE are presented in Figure 7. The three conducted ablations replace one of the three design choices each.

- **w/o conditioning:** The exploration value function does not receive an embedding of the exploitation value function.
- **w/o maximum update:** The update of the exploration objective uses a regular Bellman update instead of the maximum update.
- **w/o mixing:** Actions during training rollouts are selected alternately by the exploitation and exploration policy instead of using the proposed behavior policy.

We observe that some additions only have a positive effect in specific settings. The maximum update even decreases performance in the Pendulum-v1 environment. However, the combination of all three additions always performs robustly without a failure case. To evince this, Figure 5 shows normalized accumulated results across all environments grouped by reward setting. SAC+SEE performs best on average with all active design choices. However, this is not the case for TD3+SEE. Here, especially the maximum update has a negative impact on performance in sparse reward settings. The most positively impactful addition is the mixing of exploration and exploitation policy.



(a) Accumulated average performances of SAC+SEE ablation runs.



(b) Accumulated average performances of TD3+SEE ablation runs.

Figure 5: The graphs show the normalized aggregated average returns grouped by reward setting for all ablations. The normalization assigns the value 0 to the single worst run in an environment variant and 1 to the single best run.

5 Related works

Exploration in RL is an active field of research. While naive undirected exploration methods like ε-greedy or random noise are effective when exposed to well-shaped dense rewards, they often struggle in other settings. One common idea to mitigate this effect is to substitute unfavorably shaped rewards, such that the resulting reward ends up being well-shaped again. To this end, bonus-based methods, also referred to as intrinsic reward methods, are used. While we do not think that it is sensible to compare our method to intrinsic motivation methods, we highlight some works in that field that are conceptually related to this work. Random network distillation (RND) [3] popularized the idea of using a prediction error as a bonus reward. They also encountered the conflict of interests of strictly positive exploration rewards in an episodic setting. Many similar ideas followed, addressing several issues of this method. One general issue of bonus-based exploration is that the bonus objective is changing the MDP. A changed reward function also potentially changes the optimal policy. [27] propose a method that substitutes sparse rewards without changing the optimal policy. For this, they employ a potential function that measures the distance of states where the distance is a bisimulation metric. Intuitively, this means that the distance of states is measured as their distance in the value space instead of the feature space. As a result, their method is more likely to uncover states with high TD-errors. Due to the bisimulation metric relying on the current policy and the respective value function, this method is limited to on-policy settings. [20] propose an intrinsic motivation method, where the bonus objective is defined as the upper bound of an information gain approximation. Their method uses a learned world model for this approximation and can be combined with any off-policy reinforcement learning method. They integrate it into SAC in the same way as the entropy maximization is and are able to show strong empirical results in sparse and dense reward environments. The previously mentioned methods disregard the non-stationary nature of their exploration bonus. [26] recognize this and decouples exploration from exploitation. The exploration agent here follows an optimistic pseudo-count intrinsic reward. To better deal with the non-stationarity, the exploration reward is calculated during the update and not stored in the replay buffer. Additionally, the exploration agent is updated more aggressively than the exploitation agent. To overcome the far-off-policy instability, they use a product of exploration and exploitation policy as a behavior policy, which has the downside that these policies require a significant overlap to work in practice. [19] is similar to our method; they defined a separate exploration objective for maximizing the TD-error incurred by the exploitation objective. To solve the far-off-policy instability, they roll out both policies individually and collect the data in separate buffers. During the update, the data is mixed with a specific ratio. They had trouble stabilizing this approach with DDPG-style parametric policies and therefore augmented the optimization pipeline with an additional method [18]. Our approach roughly follows the idea of Riedmiller et al. [13] who propose to see exploration and exploitation as two separate phases in the training refereed to as 'Collect and Infer'. During the collection phase, the objective is to collect the optimal dataset such that during the inference phase, the best possible policy based on that amount of data can be learned. This requires the data collection to be aware of what data has already been collected. While our method still balances exploration and exploitation during training, it is informed about collected data through the conditioning.

6 Discussion

After empirically showing the strength of our proposed method SEE, we also want to discuss limitations of the approach. Due to the additional exploration objective, we double the amount of required function approximations. Thus, in turn with the additional updates, roughly doubling the required compute compared to the base algorithm. Bonus-based exploration methods often suffer from the so-called noisy-TV problem. It describes a setting where a novelty-seeking agent gets stuck in stochastic transitions that constantly produce novel states, such as a noisy TV. SEE is not directly affected by this as it does not seek novel states; however, stochastic rewards may produce a similar effect. For some transitions with a stochastic reward, the exploitation agent might learn the expected reward, but due to the stochasticity, there will be a consistent TD-error related to this transition on which the exploration policy might get stuck on.
We have shown that SAC+SEE works reliably in a diverse set of environments and reward settings, notably without specifically tuning hyperparameters. Therefore, we consider SAC+SEE an interesting candidate for challenging reward settings where precise hyperparameter tuning might not be feasible. Furthermore, we think that the individual design choices may also be applicable in other settings, such as when a behavior policy needs to reflect multiple objectives (mixing), a decision-making agent should be indifferent to the length of an episode (maximum update), or where an objective depends on a non-stationary target (conditioning).

Acknowledgments

This work was funded by the German Federal Ministry of Education and Research (BMBF) (Project: 01IS22078).

The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the NHR project b187cb. NHR funding is provided by federal and Bavarian state authorities. NHR@FAU hardware is partially funded by the German Research Foundation (DFG) – 440719683.

The authors also gratefully acknowledge the "Julia 2" HPC provided by the Julius-Maximilians-University Würzburg. "Julia 2" was funded as DFG project as "Forschungsgroßgerät nach Art 91b GG" under INST 93/1145-1 FUGG.

References

[1] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, December 4–9, 2017.*, volume 31, 2017. URL `https://papers.nips.cc/paper_files/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html`.

[2] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS 2016), Barcelona, Spain, December 5–10, 2016.*, volume 30, 2016. URL `https://papers.nips.cc/paper_files/paper/2016/hash/afda332245e2af431fb7b672a68b659d-Abstract.html`.

[3] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, ICLR 2019, 2019.

[4] Francesco Faccio and Louis Kirsch. Parameter-Based Value Functions. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Conference, May 3–7, 2021.*, 2021.

[5] Francesco Faccio, Aditya Ramesh, Vincent Herrmann, Jean Harb, and Jürgen Schmidhuber. General Policy Evaluation and Improvement by Learning to Identify Few But Crucial States, July 2022. URL `http://arxiv.org/abs/2207.01566`. arXiv:2207.01566 [cs, stat].

[6] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy Networks for Exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30–May 3, 2018.*, 2018.

[7] Sai Krishna Gottipati, Yashaswi Pathak, Rohan Nuttall, Sahir, Raviteja Chunduru, Ahmed Touati, Sriram Ganapathi Subramanian, Matthew E. Taylor, and Sarath Chandar. Maximum Reward Formulation In Reinforcement Learning, December 2023. URL `http://arxiv.org/abs/2010.03744`. arXiv:2010.03744 [cs, stat].

[8] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic Algorithms and Applications, January 2019. URL `http://arxiv.org/abs/1812.05905`. arXiv:1812.05905 [cs].

[9] Jean Harb, Tom Schaul, Doina Precup, and Pierre-Luc Bacon. Policy Evaluation Networks, February 2020. URL `http://arxiv.org/abs/2002.11833`. arXiv:2002.11833 [cs, stat].

[10] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016.*, 2016.

[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, December 2013. URL `http://arxiv.org/abs/1312.5602`. arXiv:1312.5602 [cs].

[12] Antonin Raffin. RL Baselines3 Zoo, 2020. URL `https://github.com/DLR-RM/rl-baselines3-zoo`. Published on GitHub.

[13] Martin Riedmiller, Jost Tobias Springenberg, Roland Hafner, and Nicolas Heess. Collect & Infer – a fresh look at data-efficient Reinforcement Learning. In *Proceedings of the 5th Conference on Robot Learning, PMLR 164:1736–1744, London, UK, November 8–11, 2021.*, 2021. URL `https://proceedings.mlr.press/v164/riedmiller22a.html`.

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. URL `http://arxiv.org/abs/1707.06347`. arXiv:1707.06347 [cs].

[15] Lukas Schäfer, Filippos Christianos, Josiah P. Hanna, and Stefano V. Albrecht. Decoupled Reinforcement Learning to Stabilise Intrinsically-Motivated Exploration, February 2022. URL `http://arxiv.org/abs/2107.08966`. arXiv:2107.08966 [cs].

[16] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *ICML 2018*, pages 1582–1591. PMLR, 2018. URL `https://proceedings.mlr.press/v80/fujimoto18a.html`.

[17] Scott Fujimoto, David Meger, and Doina Precup. Off-Policy Deep Reinforcement Learning without Exploration. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *ICML 2019*, pages 2052–2062. PMLR, 2019. URL `https://proceedings.mlr.press/v97/fujimoto19a.html`.

[18] Riley Simmons-Edler, Ben Eisner, Eric Mitchell, Sebastian Seung, and Daniel Lee. Q-Learning for Continuous Actions with Cross-Entropy Guided Policies. In *Reinforcement Learning for Real Life (RL4RealLife) Workshop in the 36 th International Conference on Machine Learning, Long Beach, California, USA, 2019.*, 2019.

[19] Riley Simmons-Edler, Ben Eisner, Daniel Yang, Anthony Bisulco, Eric Mitchell, Sebastian Seung, and Daniel Lee. Reward Prediction Error as an Exploration Objective in Deep RL. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI-PRICAI-2020, pages 2816–2823, Yokohama, Japan, July 2020. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-6-5. doi: 10.24963/ijcai.2020/390. URL `https://doi.org/10.24963/ijcai.2020/390`.

[20] Bhavya Sukhija, Stelian Coros, Andreas Krause, Pieter Abbeel, and Carmelo Sferrazza. Max-InfoRL: Boosting Exploration in Reinforcement Learning Through Information Gain Maximization. *In 9th International Conference on Learning Representations, ICLR 2025, Virtual Conference, May 5–9, 2025.*, 2025.

[21] Richard S. Sutton and Andrew Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts London, England, second edition edition, 2020. ISBN 978-0-262-03924-6.

[22] Adrien Ali Taıga, William Fedus, Marlos C Machado, Aaron Courville, and Marc G Bellemare. On Bonus-Based Exploration Methods in the Arcade Learning Environment. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–May 1*, 2020. URL `https://iclr.cc/virtual_2020/poster_BJewlyStDr.html`.

[23] Mark Towers, Jordan K Terry, Ariel Kwiatkowski, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL `https://github.com/Farama-Foundation/Gymnasium/tree/v0.28.1`. Published on GitHub.

[24] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *ICML 2018*, pages 1856–1865. PMLR, 2018. URL `https://proceedings.mlr.press/v80/haarnoja18b.html`.

[25] Grigorii Veviurko, Wendelin Boehmer, and Mathijs de Weerdt. To the Max: Reinventing Reward in Reinforcement Learning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, ICML 2024. OpenReview.net, 2024.

[26] William F. Whitney, Michael Bloesch, Jost Tobias Springenberg, Abbas Abdolmaleki, Kyunghyun Cho, and Martin Riedmiller. Decoupled Exploration and Exploitation Policies for Sample-Efficient Reinforcement Learning, July 2021. URL `http://arxiv.org/abs/2101.09458`. arXiv:2101.09458 [cs].

[27] Yiming Wang, Ming Yang, Renzhi Dong, Binbin Sun, Furui Liu, and Leong Hou U. Efficient Potential-based Exploration in Reinforcement Learning using Inverse Dynamic Bisimulation Metric. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023.*, volume 36 of *NeurIPS 2023*, pages 38786–38797. Curran Associates, Inc., 2023. URL `https://papers.nips.cc/paper_files/paper/2023/hash/79f7f00cbe3003cea4d0c2326b4c0b42-Abstract-Conference.html`.

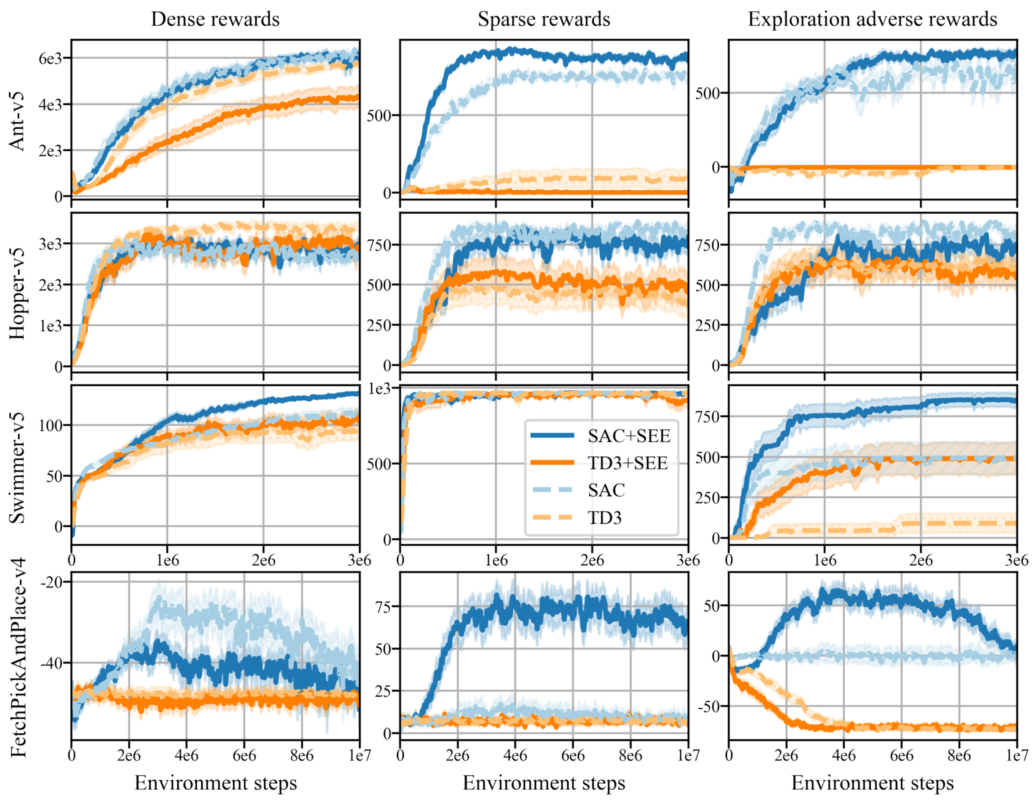


Figure 6: Comparing SAC+SEE and TD3+SEE to their respective base algorithms across multiple environments with different reward settings. The graphs show the average evaluation return across 20 seeds per environment variant. The shaded regions indicate the standard error.

A Additional results

In this appendix we present the omitted results from the main paper for the sake of conciseness. Figure 6 depicts the remaining environments of the comparison of SEE to the base algorithms. Noteworthy is that in the Hopper-v5 environment, the addition of SEE is unable to improve performance even in difficult reward settings. In Swimmer-v5 SAC+SEE outperforms SAC in the dense reward setting. We think this is due to the fact that usually for this environment a high discount of $\gamma = 0.9999$ is used. However, in our experiments, we use the same hyperparameters for all Mujoco environments (see Appendix C) and SEE seems to perform robustly in the absence of specific tuning. In the sparse FetchPickAndPlace-v4, all methods regularly encounter a positive reward as the box sometimes is initialized on top of the target position. But only SAC+SEE was capable of picking up on that signal. Figure 7 shows the individual ablation studies of TD3+SEE. It can be seen that in combination with TD3 these additions are not as effective. As previously mentioned, for TD3+SEE we do not add stochastic noise to our actions. This result therefore might hint towards stochastic exploration being beneficial even in the presence of a dedicated exploration objective. As a reminder, in SAC+SEE we employ entropy maximization on both the exploitation and exploration objectives.

The authors also gratefully acknowledge the "Julia 2" HPC provided by the Julius-Maximilians-University Würzburg. "Julia 2" was funded as DFG project as "Forschungsgroßgerät nach Art 91b GG" under INST 93/1145-1 FUGG.

B Environments

In this work we use modified versions of well know environments. Table 1 gives details on the modification that was done for each environment. All environments are original or modified versions

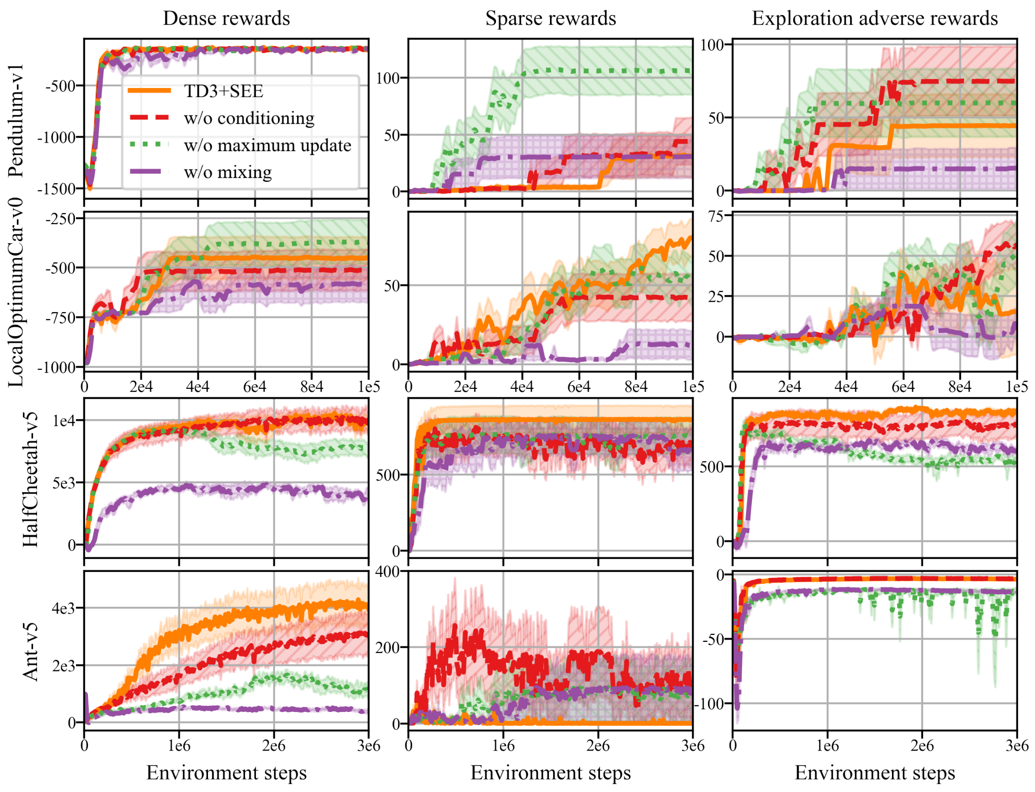


Figure 7: Comparing TD3+SEE to ablations where one of the design choices is replaced. The graphs show the average evaluation return across 10 seeds per environment variant. The shaded regions indicate the standard error.

of the environments found in Gymnasium [23] and Gymnasium-Robotics. Refer to our provided code for more details.

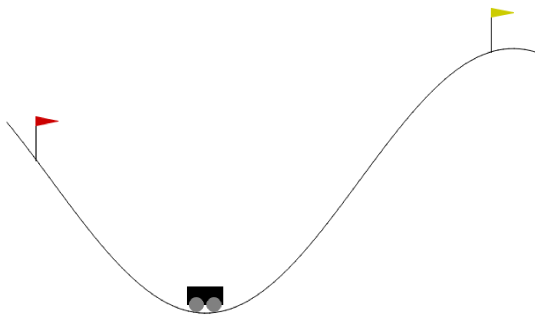


Figure 8: Depiction of LocalOptimumCar-v0 a modified version of MountainCarContinuous-v0 with an additional easier to reach goal on the left. Unlike the goal on the right the additional goal only yields a reward of 10 when reached instead of 100.

C Hyperparameters

Table 2 shows used hyper parameters. These were taken from the RL Baseline3 Zoo [12]. The parameter sets for the Classic environments were taken from the tuned version of Pendulum-v1 for TD3 and SAC respectively, and the MuJoCo sets were taken from HalfCheetah-v4 hyperparameters.

Name	**General**	**Dense**	**Sparse**	**Adverse**
Pendulum-v1	unmodified	unmodified	+1 if Pendulum within upper 10 degree else 0, Pendulum always starts pointing down with 0 velocity	sparse reward + action cost of unmodified version
LocalOptimum-Car-v0	MountainCar-Continuous-v0 with an additional lesser goal on the left at position –1.1 giving a reward of +10 (see Figure 8)	Original reward with additional negative distance to the optimal goal	original reward without action cost	original reward
HalfCheetah-v5	for sparse and adverse setting observation includes position of the agent	unmodified	+1 for x-position > 5 else 0	sparse reward + action cost of unmodified version
Ant-v5	for sparse and adverse setting observation includes position of the agent	unmodified	+1 for x-position > 4 else 0	sparse reward + action cost of unmodified version
Hopper-v5	for sparse and adverse setting observation includes position of the agent	unmodified	+1 for x-position > 1 else 0	sparse reward + action cost of unmodified version
Swimmer-v5	for sparse and adverse setting observation includes position of the agent	unmodified	+1 for x-position > 1 else 0	sparse reward + action cost of unmodified version
FetchPickAnd-Place-v4	Adding goal position to observation, episode truncated at 200 steps	using reward of FetchPickAnd-PlaceDense-v4	$+1$ if distance to goal $<= 0.05$ else 0	sparse reward - $0.1 \sum_{i=0}^{dim(\mathcal{A})} a_i^2$
LargePoint-Maze-v3	PointMaze_-Large_Diverse_-GR-v3, adding goal position to observation	reward is negative distance to goal position	$+1$ if distance to goal $<= 0.45$ else 0	sparse reward - $0.1 \sum_{i=0}^{dim(\mathcal{A})} a_i^2$

Table 1: List of all used environments with descriptions of general modifications and modifications made for each of the three setting namely dense-rewards, sparse-rewards and exploration adverse-rewards.

The discount factor is always set to 0.99. SEE does not introduce relevant new hyperparameters but doubles them as we separately optimize for two objectives. All hyperparameters for the *exploration* objective are copied from the *exploitation* objective. For the FetchPickAndPlace environments, we use an additional hidden layer such that the hidden dims are $[400, 300, 200]$. All remaining hyperparameters are the same as the MuJoCo setting.

Hyperparameter	Classic	MuJoCo
Learning rate	0.001	0.0003
Batch size	256	256
Discount γ	0.99	0.99
Replay buffer size	200000	1000000
Target networks τ	0.005	0.005
Update freq	1	1
Warm-up steps	1000	10000
Hidden dims	$[400, 300]$	$[400, 300]$
Temperature	auto	auto
Initial temperature	1.0	1.0
Target entropy	auto	auto

(a) Hyperparameters used for SAC

Hyperparameter	Classic	MuJoCo
Learning rate	0.001	0.001
Batch size	256	256
Discount γ	0.99	0.99
Replay buffer size	200000	1000000
Target networks τ	0.005	0.005
Critic update freq	1	1
Actor update freq	2	2
Target update freq	2	2
Warm-up steps	1000	10000
Hidden dims	$[400, 300]$	$[400, 300]$
Action noise std	0.1	0.1
Target noise std	0.2	0.2
Target noise clip	0.5	0.5

(b) Hyperparameters used for TD3

Hyperparameter	Classic	MuJoCo
Probe states (3.3)	16	16

(c) Hyperparameters used for SAC+SEE that differ from SAC

Hyperparameter	Classic	MuJoCo
Action noise std	0.0	0.0
Probe states (3.3)	16	16

(d) Hyperparameters used for TD3+SEE that differ from TD3

Table 2: Overview of used hyperparameters by methods and environments. Classic includes the environment Pendulum-v1 and LocalOptimumCar-v0. MuJoCo includes all remaining environments. For the FetchPickAndPlace-v4 environment all methods used one additional hidden layer and thus had hidden dims of $[400, 300, 200]$.

D Implementation details

For the sake of clarity, some details of SEE are omitted from the main paper that concern the implementation.

Policy mixing balance In section 3.1 it is mentioned that in our case the two objectives do not require scaling to achieve a balanced mixing due to a similar magnitude. In practice we use two mixing factors λ and $(1-\lambda)$ for the relative advantage values during mixing. For all our experiments lambda is set to $\lambda = 0.5$. Therefore, this does not change their relative magnitude. However, it does change their absolute magnitude, which could affect the sampling probabilities of the Boltzmann distribution.

Exploration Update The SEE extensions make use of all update methods used in the respective base algorithms. Including target networks, multiple critics, soft critics, minimum over multiple critics for target calculation, etc. This means that if the base algorithm uses the minimum over two target networks for its next value estimate, so does the update of our exploration objective.

Relative advantage calculation To calculate the relative advantage, we take the difference of two state-action values. The exact calculation of these state-action values imitates the calculation of the state-action value for the actor update of the base algorithm. This means that in TD3+SEE this is simply the state-action value given by the first of the two critic networks. In SAC+SEE this is the minimum across both state-action values.

Use of online parameters for exploration reward For calculating the exploration reward, only the online parameters of the exploitation objective are used (not of the target network), as these are also the only parameters used for the conditioning of the exploration objective.

Reproducibility Our full implementation to reproduce all results is available at: `https://github.com/Sebastian-Griesbach/SEE`.