NEURAL EPISODIC CONTROL WITH STATE ABSTRACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing Deep Reinforcement Learning (DRL) algorithms suffer from sample inefficiency. Generally, episodic control-based approaches are those solutions that leverage highly-rewarded past experiences to improve the DRL algorithms' sample efficiency. However, previous episodic control-based approaches have failed to utilize the latent information from the historical behaviors (*e.g.*, state transitions, topological similarities, *etc.*) and lack scalability during DRL training. This work introduces Neural Episodic Control with State Abstraction (NECSA), a simple but effective state abstraction-based episodic control containing a more comprehensive episodic memory, a novel state evaluation, and a multi-step state analysis. We evaluate our approach to the MuJoCo environments in OpenAI gym domains. The experimental results indicate that NECSA achieve higher sample efficiency than the state-of-the-art episodic control-based approaches. Our data and code are available at the website: https://sites.google.com/view/drl-necsa.

1 INTRODUCTION

Deep reinforcement learning (DRL) has garnered much attention in both research and industry, with applications in various fields related to artificial intelligence (AI) such as games (Mnih et al., 2013; Silver et al., 2018), autonomous driving (Xu et al., 2020), and robotics (Thomaz & Breazeal, 2008). DRL usually achieves excellent performance for many tasks and sometimes outperforms human beings. However, human-level DRL policies usually require a tremendous amount of data and millions of training steps, which are demonstrated to be sample inefficient (Arulkumaran et al., 2017; Tsividis et al., 2017). To mitigate this problem, many approaches have been proposed, such as improving the exploration (Yu, 2018; Burda et al., 2018), modeling the environment (Moerland et al., 2020), state abstraction (Vezhnevets et al., 2017) and knowledge transfer (Lazaric et al., 2008). However, this paper focuses on resolving the problem of sample inefficiency through episodic control.

Episodic control is designed to assist DRL agents in making the appropriate decisions in unseen environments using past experiences. The idea is inspired by a biological mechanism, hippocampus (Lengyel & Dayan, 2007). Moreover, episodic control has been adopted to tackle the sample inefficiency in DRL (Blundell et al., 2016; Pritzel et al., 2017). Previous neural episodic control-based approaches have usually stored past experiences in a tabular memory. Therefore, the agent could retrieve the past highly-rewarded experiences by looking up similar cached states from the episodic memory. Then the state (action) values could be estimated based on the similar states retrieved. In this way, the policy can quickly reduce the bias between episodic and model estimated state values and generalize the past highly-rewarded cases.

Although many episodic control-based approaches were proposed to improve the sample efficiency of DRL policy, all of them suffer from obvious limitations (Hu et al., 2021; Pinto, 2020; Kuznetsov & Filchenkov, 2021). In general, they only store the concrete states, actions, and state values (Blundell et al., 2016). On the other hand, their episodic memory does not record information such as time steps and transitions in the traces. As a result, some latent semantics, such as state transitions and topological similarities, cannot be explored and exploited. However, many previous works demonstrate that such latent information can be used to improve sample efficiency (Kuznetsov & Filchenkov, 2021; Zhu et al., 2020). For instance, a DRL model is trained to make decisions continuously, and the influence (*e.g.*, approximation errors) of a state-action pair might be accumulated and affect the forward scenes (Dynkin, 1965). In other words, the root cause of a *bad* decision in the

current state can either come from the latest state or the much earlier states. The topological state transitions can help trace the root cause of the *bad* decisions.

Moreover, the data structure of existing episodic memories does not efficiently support storing and exploring latent semantics. The reason is that the concrete state representations usually consist of float numbers. Thus almost none of the states are the same. Consequently, we cannot directly count and retrieve them from episodic memory, and it is impossible to identify the critical state transitions. In addition, existing episodic control-based approaches, which utilize distance-based measurement to retrieve the k-st most similar concrete states (*e.g.*, k-nearest neighbors, *i.e.*, *kNN* search) and the weighted sum of the state (action) values for the estimation (Pritzel et al., 2017; Lin et al., 2018), are inevitably resource-consuming. Overall, existing episodic control-based approaches lack (1) a more comprehensive episodic memory analysis (*i.e.*, multi-step state transitions) and (2) a more scalable storage and retrieval strategy for episodic data.

We propose NECSA, a state abstraction-based neural episodic control approach that enables a more comprehensive analysis of episodic data and better sample efficiency to address the above issues. Inspired by multi-grid and model-based reinforcement learning (Grześ & Kudenko, 2008; Kaiser et al., 2019), we discretize the continuous state space into finite grids on each dimension, and the states located in the same grid will be labeled with a unique *ID*. Naturally, we conduct a multi-step analysis of the state transitions by treating the consecutive state transitions as a fixed pattern. Finally, we make the policy generalize different patterns, as we infer that analyzing and generalizing such multi-step patterns might result in better performance than just focusing on a single state (Sutton & Barto, 2018). Such abstraction enables the following strengths: (1) based on the abstracted state space, more advanced semantic characteristics, such as state transitions and topological similarities, can be analyzed for improving performance (Grześ & Kudenko, 2008); (2) the complexity of storing and retrieving the episodic data is reduced $\mathcal{O}(N)$ to $\mathcal{O}(1)$ since we can retrieve the episodic memory using an exact match.

Previous episodic controls used average state values as the state measurement to correct the DRL policy estimation (Lin et al., 2018; Kuznetsov & Filchenkov, 2021). Nevertheless, such a state measurement cannot be computed directly for a multi-step pattern. Instead, we propose an intrinsic state measurement based on state abstraction instead of past state values. Specifically, we record the returns of those episodes where they occur in an abstract pattern. Then we compute the average returns to measure the abstract pattern. This measurement can efficiently identify those patterns which can result in higher rewards. By utilizing such intrinsic rewards (Burda et al., 2018), we revise the policy and accelerate the learning by encouraging those states with higher measurements but punishing those with relatively low measurements.

Finally, we evaluate NECSA on MuJoCo (Todorov et al., 2012) in OpenAI gym (Brockman et al., 2016) domains. The evaluation shows that our approach can significantly improve the sample efficiency and outperform state-of-the-art episodic control. In summary, we make the following contributions: (1) we propose a multi-step analysis of state transitions to achieve better policies; (2) we propose a comprehensive episodic memory that enables a more advanced analysis of past experiences; (3) we propose an intrinsic reward-based episodic control method to optimize the policy.

2 Related Work

2.1 Neural Episodic Control

Episodic control (Lengyel & Dayan, 2007) was creatively applied on model-free DRL tasks to retrieve episodic memory-based state values (Blundell et al., 2016) for resolving sample inefficiency. The distance-based measurements were applied for looking up similar episodic data (Pritzel et al., 2017). The episodic memory buffer can be smaller by applying Gaussian Random Projection to reduce the dimension of concrete states (Lin et al., 2018). The retrieved state values from episodic memory can be combined with the predictions of the critic network (Hansen et al., 2018). Novel episodic memory structure such as associative memory and generalized memory efficiently propagates state values to stored memory items (Zhu et al., 2020; Hu et al., 2021). Episodic control was also combined with multi-agent tasks (Zheng et al., 2021) with curiosity-based exploration and model-based reinforcement learning (Le et al., 2021). Recently, the episodic control was also applied to the continuous control (Zhang et al., 2019; Kuznetsov & Filchenkov, 2021), which outperformed

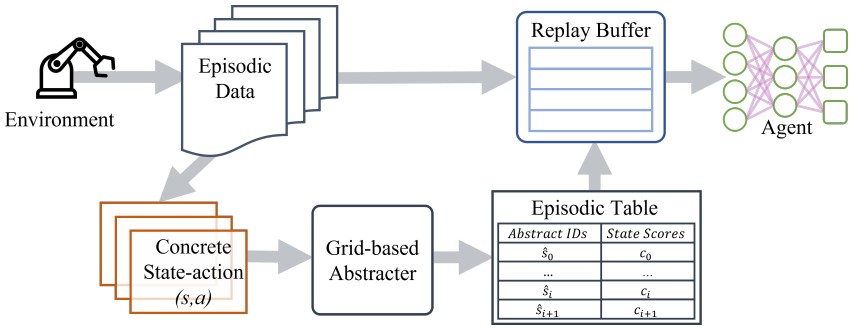


Figure 1: The overview of NECSA. We abstract the concrete state-action pairs, measure the abstract states and combine the state measurement with the traces. Finally, the revised traces will be stored in the replay buffer for the agent to sample.

the state-of-the-art DRL algorithms. The universal episodic control (Hu et al., 2021; Pinto, 2020) was proved effective on both discrete and concrete action space by adopting a generalized episodic memory (*e.g.*, neural network) to fit the past state values. Offline tasks can be solved by the state value-based episodic buffer and avoid the over-generalization of actions in the dataset (Ma et al., 2021). Episodic memory was also used to search the optimal hyperparameter for policy gradient methods (Le et al., 2022). This paper proposes a novel and simple episodic control, NECSA, which adopts a grid-based state abstraction instead of distance-based measurement to store and retrieve episodic memory more efficiently. Moreover, unlike previous works, our episodic memory is more comprehensive since the state abstraction strategy enables us to explore abstract state transitions and topological information, which can be used to improve performance (Yin & Li, 2020).

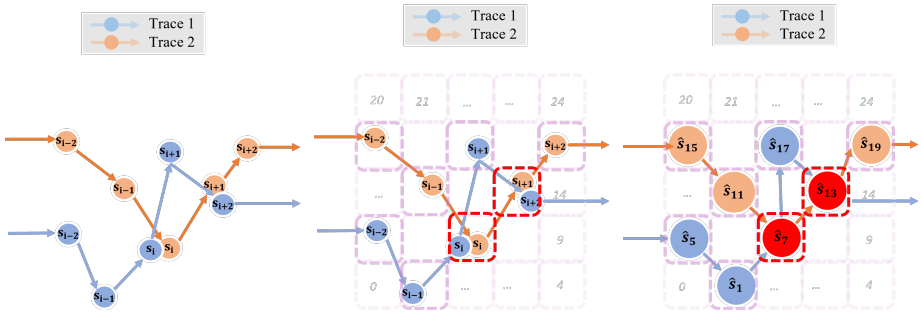
2.2 State Abstraction

The purpose of state abstraction is to group those states which share the same characteristics into a single cluster. One solution is discretizing the continuous domains (Dougherty et al., 1995). In model-based RL tasks (Kaiser et al., 2019), the state abstraction (Jiang et al., 2015; Burda et al., 2018) plays an essential role in reducing the data scales, although such abstraction requires strong domain knowledge of the environment or assumptions. In model-free reinforcement learning, the grid-based state abstraction has proved effective (Anderson & Crawford-Hines, 1994; Grześ & Kudenko, 2008). Particularly in continuous control, the continuous action space can be divided into equal intervals, then those concrete actions in the same value scales can be labeled by a common abstraction (Pazis & Lagoudakis, 2009; Tang & Agrawal, 2020). Another work of state abstraction in reinforcement learning is DreamerV2 (Hafner et al., 2020) which encodes the observations directly as discrete state variables. The idea of smoothing the state space (Gangwani et al., 2020) shares similarities with our work exploring topological information on state transitions. Dynamic state clustering (Mannor et al., 2004) is proved to be effective, although it is hard to be applied to the multi-step analysis. Moreover, previous works partially focus on state or action space, and few works have been done on abstracting state-action pairs. Inspired by the above works, we use the grid-based abstraction to group the state-action pairs simultaneously.

3 Background

Generally, the process of reinforcement learning (RL) can be defined as a Markov Decision Process (MDP) (Sutton & Barto, 2018). MDP is a four-tuple $\langle S, A, R, P \rangle$, where S and A represent the sets of states and actions respectively. The agent interacts with the environment at each time step t by observing the current state $s_t \in S$, choosing an action $a_t \in A$ to execute, receiving an immediate reward $r_t = R(s_t, a_t)$ after doing a_t, and transferring to a new state $s_{t+1} \sim P(s_t, a_t)$. The $R(\cdot)$ and $P(\cdot)$ are the reward function and state transition function respectively.

The agent selects an action $a \sim \pi(s)$ to execute according to a policy $\pi(\cdot)$ and interacts with the environment to generate a trace, $[(s_0, a_0, r_0), ..., (s_t, a_t, r_t), ...]$, where the subscripts denote different



(a) Concrete state transitions. (b) Grid-based state abstraction. (c) Abstract state transitions.

Figure 2: Take a 2-dimensional concrete state space as an example. Concrete states can be labeled by the *ID* of grids. For example, In trace 1, state s_{i-1} is abstract state $\hat{s}_1$. Concrete state s_{i+2} in trace 1 and s_{i+1} in trace 2 are located in the same grid, thus they share the same abstract state $\hat{s}_13$ (red ones). In this way, trace 1 can be represented as $(\hat{s}_5, \hat{s}_1, \hat{s}_7, \hat{s}_{17}, \hat{s}_{13})$.

time steps. The return of each trace is defined by $\sum_{t=0}^{T} \gamma^t r_t$, where rewards are discounted by a factor $\gamma \in [0, 1)$. RL generally aims to find an optimal policy to maximize the returns.

The state-action value function: $Q_\pi(s_t, a_t) = E_\pi\big[\sum_{t=0}^{T-1} \gamma^t r_t | s_t, a_t\big]$ is widely used in many RL algorithms. In value-based approaches (e.g., DQN (Mnih et al., 2013)), action with maximal $Q_\pi(s_t, a_t)$ value will be selected, and these approaches have achieved great success in discrete action space environments. The policy-based approaches such as A3C (Mnih et al., 2016) are more efficient and suitable for continuous action spaces than value-based approaches, which generate a policy distribution and sample actions from it instead of iterating in the infinite action space. DDPG (Lillicrap et al., 2016) is used in a continuous action setting to improve the sample efficiency over the vanilla actor-critic, in which the *deterministic* means the Actor can directly output the actions instead of computing a probability distribution over actions. Twin Delayed DDPG (TD3) (Fujimoto et al., 2018) tackles the over-estimation of state values. In this work, we build our episodic control in continuous action space based on TD3 (Fujimoto et al., 2018).

4 Methodology

This paper proposes an effective and efficient episodic control framework, NECSA. Figure 1 shows the main procedure of our framework. We have two additional modules: (1) a grid-based abstraction to convert the concrete states to abstract *ID* (*i.e.*, the $\hat{s}_i$ as noted later); (2) a key-value-manner episodic memory module to store the abstract states with the state measurement (*i.e.*, the reward confidence scores c_i as mentioned later). The traces are revised based on the state measurement and further flow to the replay buffer for policy optimization. NECSA is highly supplementary, which can be applied to the general reinforcement learning paradigm.

4.1 Grid-based State Abstraction

The K-dimensional state space R^K contains infinite concrete states, each state s_i is represented as $(s_i^0, ..., s_i^K)$. Each component s_i^m in the m-th dimension satisfies $s_i^m \in [l_m, u_m]$, where the l_m and u_m are the lower and upper boundaries, respectively. We split each dimension $[l_m, u_m]$ into N equal intervals. Thus the concrete state space R^K will be divided into N^K grids as follows:

$$e_n^m = [l_m + n \times \frac{u_m - l_m}{N}, l_m + (n+1) \times \frac{u_m - l_m}{N}] \tag{1}$$

where e_n^m represents the n-th interval on the m-th dimension in the state space. In this way, (1) each concrete will fall into a grid; (2) all the concrete states s_i which fall into the same grid can be identified as in the same cluster; (3) each cluster (*i.e.* each grid) can be represented by a unique abstract state $\hat{s}$:

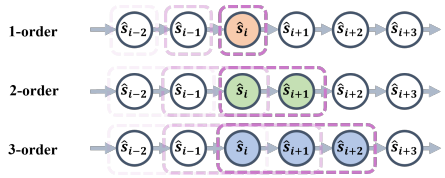


Figure 3: A schematic diagram of the multi-step analysis. We slide windows with different lengths to extract patterns.

$$\hat{s} = \{s_i | s_i^m \in e_n^m, n \in [0, N], m \in [0, K]\} \tag{2}$$

Figure 2 is a schematic diagram of grid-based clustering, in which the infinite concrete state space is converted into the finite discrete state space. The environment determines the l_m and u_m. It is worth mentioning that we can either perform the abstraction on concrete states or state-action pairs. As the experimental results suggest, abstracting state-action pairs can perform better than focusing on concrete states. We use a random projection to reduce the dimension of the concrete state-action pairs and make our method more scalable for high-dimensional tasks.

4.2 Episodic Memory with Multi-step Analysis and State Measurement

Figure 3 shows the logic of extending the state abstraction to multi-step patterns. The abstract patterns are extracted by sliding a N-step window on each trace. We treat N-step continuous abstract states as a whole pattern. Moreover, we count the patterns iteratively and orderly, where $\{\hat{s}_i, \hat{s}_{i+1}, \hat{s}_{i+2}\}$ and $\{\hat{s}_{i+1}, \hat{s}_{i+2}, \hat{s}_{i+3}\}$ are treated as different patterns.

To revise the state (action) values (*i.e.* Q) estimation, previous episodic control-based approaches usually leverage a shift of episodic returns to correct the critic Q-value. Such approaches can achieve better performance while requiring domain-specific knowledge of corresponding DRL algorithms, model structure reform, and engineering work efforts. Furthermore, measuring states by the statistics of Q-values is only applicable in one-step episodic memory. With multi-step patterns, we propose a state measurement named reward confidence scores. The score c_i represents the expectation of the episode return when $\hat{s}_i$ occurs. Although the scores are similar to state (-action) values in terms of the definitions, they stand for different semantics. State (action) values represent the forwarding rewards after the state $\hat{s}_i$, but the scores measure the episode reward over the whole trace.

We design a simple key-value-manner episodic memory C based on state abstraction and measurement. Each item in the episodic memory is indexed by the $\hat{s}_i$ with three values: the current score c_i, the historical total episodic rewards $\mathcal{E}_i$ that earned by the traces which contain the $\hat{s}_i$, and the total number of occurrences η_i of $\hat{s}_i$. The following equation is the implementation of computing scores:

$$\begin{cases} \mathcal{E}_i = \sum r, & \eta_i = 1, & c_i = \mathcal{E}_i, & \text{if } \hat{s}_i \notin C \\ \mathcal{E}_i = \mathcal{E}_i + \sum r, & \eta_i = \eta_i + 1, & c_i = \dfrac{\mathcal{E}_i}{\eta_i}, & \text{otherwise,} \end{cases} \tag{3}$$

Where the $\sum r$ is the final episodic return of the trace, which contains $\hat{s}_i$, in this way, our memory buffer can efficiently record past experiences and update the scores. Our episodic memory enables three operations: *ADD*, *LOOKUP* and *UPDATE*. *ADD* is the operation to store the $\hat{s}_i$ into C. *LOOKUP* returns score c_i for $\hat{s}_i$ by taking $\mathcal{O}(1)$. *UPDATE* is to revise the c_i of $\hat{s}_i$. Moreover, the episodic memory enables the normalizing of the scores to [0,1], and the average of the score c_i is adopted as the intrinsic reward for abstract pattern $\hat{s}_i$.

4.3 Episodic Control-Based Learning

As inspired by previous episodic control-based approaches, we infer that a statistic-based measurement of state-action pairs can be a helpful reference to revise the estimation of state (-action) values by the critic network (Lin et al., 2018; Kuznetsov & Filchenkov, 2021). Therefore, we use an intrinsic

Algorithm 1 NECSA.

Input: $\mathcal{M}$: DRL model, E: Environment, $\mathcal{B}$: Replay Buffer
Input: C: An episodic memory to record each $\hat{s}_i$ and the score c_i
Output: $\mathcal{M}'$: A tuned DRL model

1: Initialization: $i \leftarrow 0$, $S, \hat{S}, A, R, D, \mathcal{B} \leftarrow \emptyset$
2: **while** $i < Total_steps$ **do**
3: $\quad s_i \leftarrow state, a_i \leftarrow action, r_i \leftarrow reward, d_i \leftarrow done$
4: $\quad S.append(s_i), A.append(a_i), D.append(d_i)$
5: $\quad \hat{s}_i \leftarrow State_Abstract(s_i, a_i)$
6: $\quad \hat{S}.append(\hat{s}_i)$
7: $\quad c_i \leftarrow Inquire_Score(\hat{s}_i, C)$
8: $\quad \hat{r}_i \leftarrow Reward_Revision(r_i, c_i)$
9: $\quad R.append(r_i)$
10: $\quad \mathcal{B}.add(s_i, a_i, \hat{r}_i)$
11: $\quad$ **if** d_i is true **then**
12: $\quad\quad C \leftarrow Update_Score(\hat{S}, R, C)$
13: $\quad\quad S, \hat{S}, A, R, D \leftarrow \emptyset$
14: $\quad$ **end if**
15: $\quad i \leftarrow i + 1$
16: $\quad \mathcal{M} \leftarrow Train(\mathcal{M}, \mathcal{B}.sample())$
17: **end while**

reward (Burda et al., 2018) to revise the policy. Formally, the revised reward $\hat{r}_i = r_t + \Delta$, where r_t is the return of the original reward function, Δ is the intrinsic reward (computed based on the reward confidence scores) which represents the semantics for improving the rewarding mechanism. We perform reward revision for each pattern as follows:

$$\hat{r}_i = r_i + \underbrace{(c_i - \frac{\sum_{m=0}^{M} c_m}{M})}_{\Delta} \times \epsilon, \tag{4}$$

where the $\hat{r}_i$ is the revised reward based on r_i. M is the total number of abstract patterns. $\frac{\sum_{m=0}^{M} c_m}{M}$ is the average of all the scores.

The term Δ will be less than 0 if the $\hat{s}_i$ score is less than the average. Thus the corresponding concrete state will be punished. The state will be encouraged if Δ is greater than 0. The agent will earn more rewards if it passes through $\hat{s}_i$. In a word, the lower the score c_i is, the more the pattern is punished. ϵ is a hyperparameter that helps control the magnitude of punishment and encouragement.

We implement NECSA as Algorithm 1. We use a module named *State_Abstract* in line 5 which can covert the concrete states to abstract patterns in run-time. While tuning the model, scores can be inquired from the episodic memory C in line 7, and the reward will be revised in line 8. We also added a module named *Update_Score* in line 12, which helps calculate the scores for the abstract patterns in $\hat{S}$ and update the scores in C as each termination of an episode. The revised rewards will be stored in the replay buffer $\mathcal{B}$. Finally, the policy will be updated with the sampled data from the $\mathcal{B}$ in line 16.

5 EXPERIMENT

5.1 EXPERIMENT SETUP

We conduct the experiments on nine MuJoCo environments in OpenAI gym (Brockman et al., 2016) domains, including Walker2d-v3, Hopper-v3, Humanoid-v3, Ant-v3, HalfCheetah-v3, Swimmer-v3, Reacher-v2, InvertedPendulum-v2, and InvertedDoublePendulum-v2. The selected environments are typical continuous control tasks. We select four baselines include EMAC (Kuznetsov & Filchenkov, 2021), GEM (Hu et al., 2021) DDPG (Lillicrap et al., 2016) and TD3 (Fujimoto et al., 2018) for the comparison. EMAC and GEM are state-of-the-art episodic control approaches and proved effective

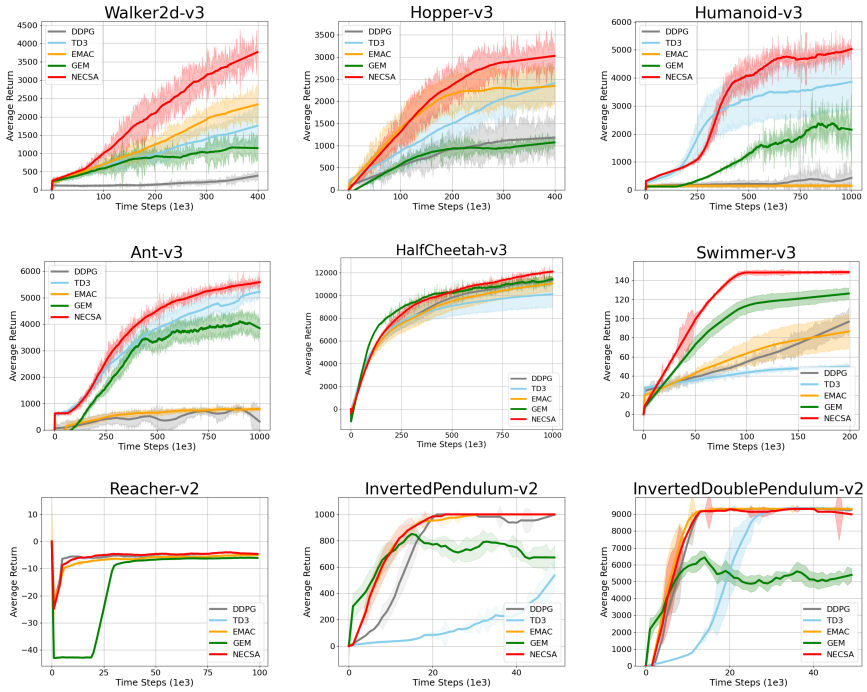


Figure 4: The evaluation of each approach on MuJoCo tasks.

on continuous control tasks. EMAC retrieves experiences from a non-parametric episodic memory to minimize the gap between the estimation and statistics of state (-action) values via a novel critic loss. GEM integrates a neural network to generalize episodic experiences, enabling fast episodic memory retrieval. We adopt the benchmark of DDPG and TD3 in Tianshou (Weng et al., 2021). The implementation of NECSA is based on TD3.

We evaluate our approach on InvertedPendulum-v2 and InvertedDoublePendulum-v2 for 50,000 time steps, Reacher-v2 for 100,000 steps, Swimmer-v3 for 200,000 steps, Walker2d-v3 and Hopper-v3 for 400,000 steps, Humanoid-v3, Ant-v3 and HalfCheetah-v3 for 1,000,000 steps, respectively. We run each experiment with five different random seeds to counteract the randomness and compare the average performance over respective training steps. It takes 4 to 8 GPU hours to finish five seeds on each task. All the experiments were run on the powerful servers with CPU (Intel(R) Core(TM) i9-10940X CPU @ 3.30GHz), GPU(NVIDIA Corporation GA102GL [RTX A6000]) with 128GB RAM.

5.2 Evaluation Results

Figure 4 shows the main evaluation of our approaches. In our experiments, we set the length of the multi-step abstract pattern m=3. As shown in Figure 4, our approach NECSA significantly outperforms the baselines on six tasks, including Walker2d-v3, Hopper-v3, Humanoid-v3 and Ant-v3, HalfCheetah-v3, and Swimmer-v3, in terms of sample efficiency. Furthermore, NECSA reaches (slightly outperforms) state-of-the-art performance in Reacher-v2, InvertedPendulum-v2 and InvertedDoublePendulum-v2. Table 1 is the average return of the learned policies over the respective training steps. The bold entries in Table 1 mark the best average returns of each method. NECSA achieves the best results on all tasks in the limited training steps. Overall, we found that (1) NECSA is the most sample efficient algorithm on all tasks while other algorithms only establish good performance on the part of tasks; (2) NECSA performs better on relatively complex tasks (*e.g.*, high-dimensional tasks). Such results prove that NECSA is a scalable episodic control on continuous control tasks.

Moreover, EMAC and GEM struggle on Humanoid-v3 (a relatively complex task in MuJoCo since the observation space is large), while NECSA can perform better than all the baselines on this task. Note

Table 1: The best average returns of each approach on MuJoCo tasks.

Task Name	DDPG	TD3	EMAC	GEM	NECSA
Walker2d-v3	387.73	1756.09	2338.24	2024.94	**3768.08**
Hopper-v3	1180.64	2411.15	2347.98	1450.24	**3026.57**
Humanoid-v3	431.69	3858.07	3296.64	3846.88	**4540.12**
Ant-v3	811.53	5223.94	789.99	4107.28	**5588.44**
HalfCheetah-v3	11314.36	10120.34	11052.82	11458.70	**12097.91**
Swimmer-v3	138.42	60.10	101.71	146.65	**149.63**
Reacher-v2	-4.05	-4.01	-5.01	-5.99	**-3.97**
InvertedPendulum-v2	1000.0	535.97	1000.0	955.48	**1000.0**
InvertedDoublePendulum-v2	9316.83	9333.19	9327.17	6858.88	**9359.33**

that in Humanoid-v3, a 376-dimensions state space with five grids on each dimension will generate a 5^{376} abstract state space, which is almost impossible for efficient storage and retrieving in the episodic memory. Gaussian random projection can be a promising solution to make the concrete states smaller (Pritzel et al., 2017; Kuznetsov & Filchenkov, 2021). Therefore, we perform a dimension reduction of the concrete state vectors to 1×25 by a random projection and divide each dimension into eight intervals. The results show that the random projection reduces dimension without significant information loss, thus making NECSA a scalable episodic control on high-dimensional tasks.

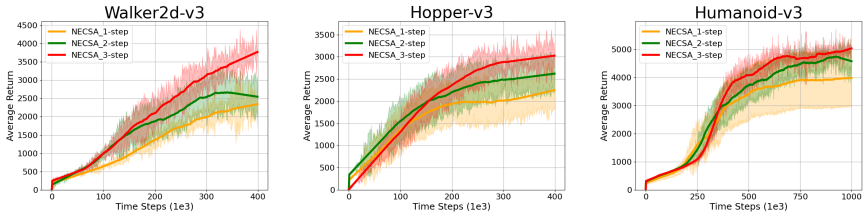


Figure 5: The evaluation of performance between one-step and m-step NECSA.

5.3 Ablation Study

One of the most critical parts of this paper is to adopt a multi-step analysis of past experiences. To better demonstrate the effectiveness of the multi-step episodic control, we compare the performance with one-step and two-step settings. The results in Figure 5 show that the three-step NECSA outperforms one-step and two-step settings. Another observation is that the two-step NECSA usually establishes better sample efficiency than the one-step settings. Such results prove that multi-step analysis of past traces can significantly improve performance on various tasks. However, longer steps are not always better. In our evaluation, the default setting of NECSA multi-step (m) analysis is set to 3. The reason is that longer patterns cause the exact matches to become exceedingly rare, which makes episodic control less helpful for improving performance.

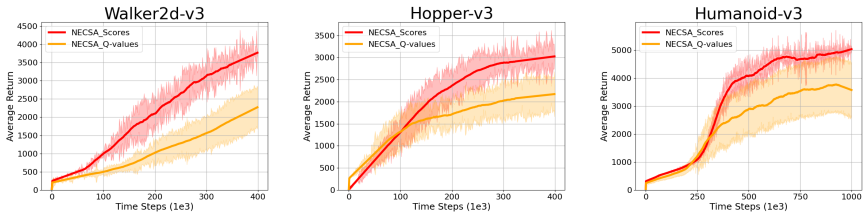


Figure 6: The evaluation of performance between measuring the abstract state by reward confidence scores and Q values.

We also study the effectiveness of reward confidence scores. We use scores to measure each state-action pair, although previous works mainly focus on the state-action values (*i.e.*, the Q-values).

Specifically, we use the average of scores and Q-values (Lin et al., 2018; Kuznetsov & Filchenkov, 2021) for each abstract state (pattern) to compute the intrinsic reward. Note that we cannot directly attain the Q-value of a multi-step abstract pattern. We instead use the average Q value of the corresponding concrete states (Lin et al., 2018). The results in Figure 6 show that measuring the state-action pairs through scores (NECSA_Scores) can lead to better sample efficiency than using Q-values (NECS_Q-values). We claim that such state measurement is practical and necessary since the scores contain the overall impact of an abstract on a whole trace. At the same time, Q-values are the expectations of forwarding rewards. Furthermore, such a difference makes the scores more comprehensive since we reference the historical impact on the abstract states.

We conclude that NECSA can perform better than the state-of-the-art episodic control-based approaches based on the experiments. State abstraction-based episodic memory enables more advanced analysis of past experiences. The scores of state abstractions are valuable, and the rationale for state measurement and the multi-step analysis of state transitions can help significantly improve the performance.

6 Discussion

We adopt grid-based abstraction instead of *kNN*. Although *kNN* can group concrete states into unique IDs, it cannot satisfactorily update episodic memory in real time. For instance, if we perform the clustering by *kNN* in 10,000 steps, then in the 11,000 steps, the clustering in 10,000 steps may lose generality and accuracy. However, performing new clustering costs extra computation time. Grid-based state abstraction can avoid this since it can handle any newcomer concrete states without altering the previous clustering. Moreover, *kNN* does not support inquiries about the state measurement of multi-step patterns. We extend the episodic control to multi-step analysis, which is one of the most important contributions of NECSA. Overall, we select a grid-based abstraction (clustering) instead of *kNN* for (1) more efficient state abstraction and (2) multi-step abstract pattern analysis.

Although NECSA establishes promising sample efficiency, it still has much room for improvement. First of all, the state abstraction strategy might cause variances or errors. For example, we divide each dimension by the same number of intervals, though we can not ensure that the value scale of each dimension is the same. Therefore, some existing concrete states might not be labeled by abstract *ID*. Based on prior knowledge of the states' space, we infer that a possible solution may be to divide each dimension into appropriate but not fixed intervals automatically.

The motivation behind NECSA is to construct a more comprehensive episodic memory. We consider that past experiences need to be further explored, and more valuable information will be found and exploited. Furthermore, exploring latent semantics based on the abstraction-based episodic memory promises to improve the sample efficiency. In addition, NECSA is highly supplementary since it does not require reforming the models or loss functions. Therefore, we believe that NECSA can help improve the performance of more tasks.

7 Conclusion and Future Work

We propose NECSA, a novel state abstraction-based neural episodic control approach to improve DRL's sample efficiency. We transform dense concrete state-action pairs into abstract patterns, which enables efficient storage and retrieval of episodic data. We also perform a multi-step analysis of the pattern transitions to enhance their performance. Furthermore, we propose a novel state measurement metric, reward confidence scores, which can be incorporated with extrinsic rewards to accelerate policy optimization. The evaluation results prove that our approach is more sample efficient than state-of-the-art episodic control approaches.

We believe that more advanced research can be conducted based on the state abstraction, state measurement, and multi-step analysis of state transitions, including (1) abstracting the behaviors of the policy via state abstraction since we can build an automaton base on the abstract states; (2) analyzing the state transitions and topological relationships of multi-agent tasks; (3) applying our approach to model-based reinforcement learning with on-the-fly planning. We leave these topics for future works.

REFERENCES

Charles W. Anderson and Stewart Crawford-Hines. Multigrid q-learning. 1994.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL http://arxiv.org/abs/1606.01540.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Machine learning proceedings 1995*, pp. 194–202. Elsevier, 1995.

E. B. Dynkin. *Markov processes*, pp. 77–104. Springer Berlin Heidelberg, Berlin, Heidelberg, 1965. ISBN 978-3-662-00031-1. doi: 10.1007/978-3-662-00031-1_4. URL https://doi.org/10.1007/978-3-662-00031-1_4.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Tanmay Gangwani, Yuan Zhou, and Jian Peng. Learning guidance rewards with trajectory-space smoothing. *Advances in Neural Information Processing Systems*, 33:822–832, 2020.

Marek Grześ and Daniel Kudenko. Multigrid reinforcement learning with reward shaping. In *International Conference on Artificial Neural Networks*, pp. 357–366. Springer, 2008.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Steven Hansen, Alexander Pritzel, Pablo Sprechmann, André Barreto, and Charles Blundell. Fast deep reinforcement learning using online adjustments from the past. *Advances in Neural Information Processing Systems*, 31, 2018.

Hao Hu, Jianing Ye, Guangxiang Zhu, Zhizhou Ren, and Chongjie Zhang. Generalizable episodic memory for deep reinforcement learning. *arXiv preprint arXiv:2103.06469*, 2021.

Nan Jiang, Alex Kulesza, and Satinder Singh. Abstraction selection in model-based reinforcement learning. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 179–188, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/jiang15.html.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

Igor Kuznetsov and Andrey Filchenkov. Solving continuous control with episodic memory. *arXiv preprint arXiv:2106.08832*, 2021.

Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 544–551, 2008.

Hung Le, Thommen Karimpanal George, Majid Abdolshah, Truyen Tran, and Svetha Venkatesh. Model-based episodic memory induces dynamic hybrid controls. *Advances in Neural Information Processing Systems*, 34:30313–30325, 2021.

Hung Le, Majid Abdolshah, Thommen K George, Kien Do, Dung Nguyen, and Svetha Venkatesh. Episodic policy gradient training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7317–7325, 2022.

Máté Lengyel and Peter Dayan. Hippocampal contributions to control: the third way. *Advances in neural information processing systems*, 20, 2007.

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 *ICLR (Poster)*, 2016.

Zichuan Lin, Tianqi Zhao, Guangwen Yang, and Lintao Zhang. Episodic memory deep q-networks. *arXiv preprint arXiv:1805.07603*, 2018.

Xiaoteng Ma, Yiqin Yang, Hao Hu, Qihan Liu, Jun Yang, Chongjie Zhang, Qianchuan Zhao, and Bin Liang. Offline reinforcement learning with value-based episodic memory. *arXiv preprint arXiv:2110.09796*, 2021.

Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 71, 2004.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

Jason Pazis and Michail G Lagoudakis. Binary action search for learning continuous-action control policies. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 793–800, 2009.

Rafael Pinto. Model-free episodic control with state aggregation. *arXiv preprint arXiv:2008.09685*, 2020.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *International Conference on Machine Learning*, pp. 2827–2836. PMLR, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419): 1140–1144, 2018.

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

Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5981–5988, 2020.

Andrea L Thomaz and Cynthia Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6-7):716–737, 2008.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Pedro A Tsividis, Thomas Pouncy, Jaqueline L Xu, Joshua B Tenenbaum, and Samuel J Gershman. Human learning in atari. In *2017 AAAI spring symposium series*, 2017.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 3540–3549. PMLR, 2017.

Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Yi Su, Hang Su, and Jun Zhu. Tianshou: A highly modularized deep reinforcement learning library. *arXiv preprint arXiv:2107.14171*, 2021.

Zhuo Xu, Jianyu Chen, and Masayoshi Tomizuka. Guided policy search model-based reinforcement learning for urban autonomous driving. *arXiv preprint arXiv:2005.03076*, 2020.

Zhao-Heng Yin and Wu-Jun Li. Toma: Topological map abstraction for reinforcement learning. *arXiv preprint arXiv:2005.06061*, 2020.

Yang Yu. Towards sample efficient reinforcement learning. In *IJCAI*, pp. 5739–5743, 2018.

Zhizheng Zhang, Jiale Chen, Zhibo Chen, and Weiping Li. Asynchronous episodic deep deterministic policy gradient: Toward continuous control in computationally complex environments. *IEEE transactions on cybernetics*, 51(2):604–613, 2019.

Lulu Zheng, Jiarui Chen, Jianhao Wang, Jiamin He, Yujing Hu, Yingfeng Chen, Changjie Fan, Yang Gao, and Chongjie Zhang. Episodic multi-agent reinforcement learning with curiosity-driven exploration. *Advances in Neural Information Processing Systems*, 34, 2021.

Guangxiang Zhu, Zichuan Lin, Guangwen Yang, and Chongjie Zhang. Episodic reinforcement learning with associative memory. In *ICLR*, 2020.

A APPENDIX

A.1 COMMON HYPERPARAMETER AND SETTINGS

Our hyperparameter settings are available in Table 2. In addition, we directly evaluate the program released by the authors of EMAC (Kuznetsov & Filchenkov, 2021) and GEM (Hu et al., 2021). We also adopt the implementation of DDPG and TD3 in (Weng et al., 2021). We build NECSA based on TD3. The neural network consists of two hidden layers. The size of each layer is 256. The activation unit is ReLU. Both the Actor and Critic networks share the same structure.

Table 2: List of main hyperparameters used in NECSA.

Hyper-paramater	NECSA
Critic Learning Rate	3e-4
Actor Learning Rate	3e-4
Optimizer	Adam
Memory Size	100000
Batch Size	100
Discount Factor	0.99
Episode Length	1000
Exploration Policy	N(0, 0.1)

A.2 ANALYSIS OF STATE VISITING DENSITY

We plot the density of state visiting times in Figure 7. Take Walker2d-v3 as an example. The 1-order NECSA generated 198,613 abstract states. 163,273 of them were visited one time, and 35,340 were visited more than one time. All the 35,340 multi-times visited states were visited 236,727 times in 400,000 training steps, which is more than 1-time visit. The m-order NECSA generated 377,629 abstract patterns. 372,253 of them were visited one time, and 5,376 of them were visited more than one time. All the 5,376 multi-times visited patterns were visited 27,747 times, which is more than the one -time visited pattern.

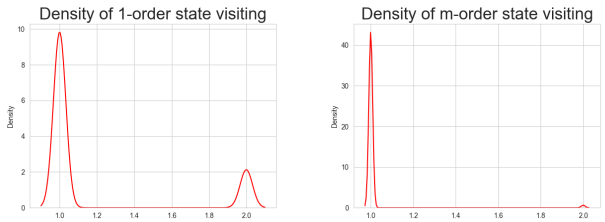


Figure 7: Density of abstract state visiting times on Walker2d-v3 (m=3).

The density plot shows that 1-time visits are more in multi-step analysis. However, the performance in Figure 4 shows that multi-step analysis can still achieve the best sample efficiency in our settings. Therefore, we assume that the multi-times visited patterns played an essential role in enhancing the sample efficiency.

A.3 SPECIAL HYPERPARAMETERS AND SETTINGS IN NECSA

During the experiments, we evaluated different settings of several hyperparameters, which can help us better understand the effectiveness of NECSA. We discuss four settings: (1) Should we abstract the state or state-action pairs? (2) What is the effect of discrete state space grid numbers on performance? (3) How to control the magnitude of revising environment rewards via intrinsic reward (*i.e.*, reward confidence scores)? (4) How do the backbone algorithms affect the performance of NECSA? We answer the above questions through the following experiments.

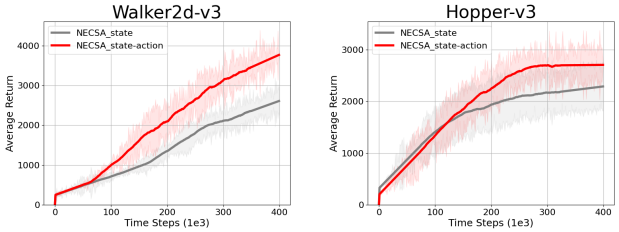


Figure 8: Comparison between abstraction of the state and state-action pairs.

Figure 8 shows the performance of abstracting state and state-action pairs. NECSA_state is the curve that we only abstract the concrete states. On the other hand, NECSA_state-action means that we abstract state-action pairs. It reveals that abstracting the state-action pairs can significantly improve performance. The reason is that abstracting state-action pairs can involve more information for modeling the policy than just focusing on states. For example, state and state-action values in DRL can be directly converted to each other, and intuitively, the scores between states and state-action pairs should also be the same. However, we found that although a set of concrete states can be located in the same grid, their corresponding state-action pairs might be divided into different grids after abstracting the state-action space. Therefore variances of the scores between states and state-action pairs are amplified. Due to such variances, abstraction with state-action pairs can improve evaluation accuracy.

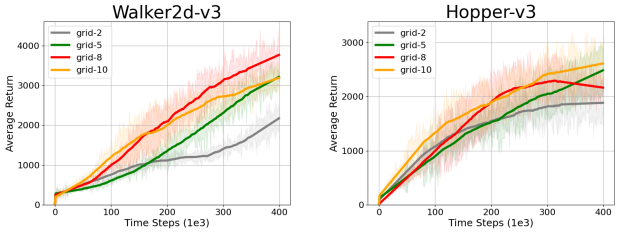


Figure 9: Comparison between different granularities of grid-based abstraction.

Figure 9 shows the performance under different abstraction granularities. Grid-N means we divide each dimension of state-action pairs into N equal intervals. The results show that grid numbers 8 and 10 are the best abstraction choices in Walker2d-v3 and Hopper-v3. Our evaluation is limited to 10 grids since a greater grid number causes the exponential growth of the abstract number, which is very memory-consuming. We found that more grids can not always achieve better performance. For example, in Walker2d-v3, 8-grids abstraction outperforms 10-grids. The reason is that although more grids can cluster similar concrete states more accurately, as the grid number of the abstraction becomes smaller, the scores of abstract states and patterns are less representative. We set the grid numbers to 10, 5, and 5 in Swimmer-v3, InvertedPendulum-v2, and InvertedDoublePendulum-v2, respectively.

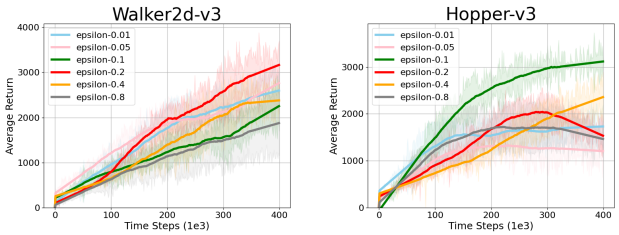


Figure 10: Comparison of performance between different granularities of revising rewards.

Figure 10 shows how the parameter ϵ in the equation Eq. equation 4 influences the performance. The results suggest two major conclusions: (1) $\epsilon = 0.2$ and $\epsilon = 0.1$ are the best for controlling

reward shaping in Walker2d-v3 and Hopper-v3, respectively; (2) a value for ϵ that is greater or smaller will reduce performance. The reason is that a smaller ϵ reduces the impact of scores, and a greater ϵ will increase the weight of state evaluation and completely reform the rewarding mechanism. In addition, we set $\epsilon = 0.15$ for Swimmer-v3 and $\epsilon = 0.2$ for InvertedPendulum-v2 and InvertedDoublePendulum-v2.

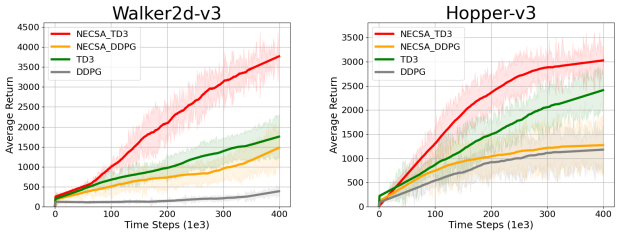


Figure 11: Comparison of performance between building NECSA on different backbones.

Figure 11 shows how the backbone algorithms affect the performance of NECSA. In this evaluation, we build NECSA on TD3 and DDPG and compare the performance with the primary algorithms. The results show that (1) algorithms can achieve better sample efficiency after incorporating with NECSA and (2) the backbone algorithms affect the performance of NECSA. For example, in Walker2d-v3 and Hopper-v3, TD3 is more sample efficient than DDPG. Such a conclusion can also be drawn after implementing NECSA, demonstrating that NECSA is scalable and effective on different backbone algorithms.