Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Embedding expert demonstrations into clustering buffer for effective deep reinforcement learning*

Shihmin WANG†1, Binqi ZHAO†1, Zhengfeng ZHANG1, Junping ZHANG†1, Jian PU†‡2

1Shanghai Key Laboratory of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai 200433, China

2Institute of Science and Technology for Brain-inspired Intelligence, Fudan University, Shanghai 200433, China

†E-mail: wangshimin20@fudan.edu.cn; bqzhao20@fudan.edu.cn; jpzhang@fudan.edu.cn; jianpu@fudan.edu.cn

Received Feb. 12, 2023; Revision accepted May 19, 2023; Crosschecked Nov. 3, 2023

Abstract: As one of the most fundamental topics in reinforcement learning (RL), sample efficiency is essential to the deployment of deep RL algorithms. Unlike most existing exploration methods that sample an action from different types of posterior distributions, we focus on the policy sampling process and propose an efficient selective sampling approach to improve sample efficiency by modeling the internal hierarchy of the environment. Specifically, we first employ clustering methods in the policy sampling process to generate an action candidate set. Then we introduce a clustering buffer for modeling the internal hierarchy, which consists of on-policy data, off-policy data, and expert data to evaluate actions from the clusters in the action candidate set in the exploration stage. In this way, our approach is able to take advantage of the supervision information in the expert demonstration data. Experiments on six different continuous locomotion environments demonstrate superior reinforcement learning performance and faster convergence of selective sampling. In particular, on the LGSVL task, our method can reduce the number of convergence steps by 46.7% and the convergence time by 28.5%. Furthermore, our code is open-source for reproducibility. The code is available at https://github.com/Shihwin/SelectiveSampling.

Key words: Reinforcement learning; Sample efficiency; Sampling process; Clustering methods; Autonomous driving

https://doi.org/10.1631/FITEE.2300084 **CLC number:** TP181

1 Introduction

Reinforcement learning (RL) algorithms have recently attained significant performance in various domains, such as autonomous vehicles (Li et al., 2023), traffic signal control (Dai et al., 2022), active object detection (Liu et al., 2022), and StarCraft II (Vinyals et al., 2019). By using the reward function, which defines what an agent should do, RL algorithms determine how to perform an action through trial-and-error learning under supervision, maximizing the cumulative rewards. Then, agents interact with the environment using the learned policy from the RL algorithms.

However, the effectiveness of RL is heavily constrained by sample efficiency. The reason is that during the RL training process, the agents constantly collect experience by interacting with the environment according to the latest learned policy, and then use experience in updating the policy (Sutton and Barto, 1998) to maximize the expected future returns. Thus, millions of samples are often required to train a satisfactory policy to attain remarkable performance improvement and robust behavior, which

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 62176059), the Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01), Zhangjiang Lab, and the Shanghai Center for Brain Science and Brain-inspired Technology

ORCID: Shihmin WANG, https://orcid.org/0000-0002-7288-8323; Jian PU, https://orcid.org/0000-0002-0892-1213

results in an expensive environment interaction process.

One way to alleviate the need for samples is to solve the credit assignment problem to improve sample efficiency. As one of the most successful on-policy algorithms, proximal policy optimization (PPO) (Schulman et al., 2017) uses a truncated version of the generalized advantage estimator (GAE) (Schulman et al., 2016), which employs importance sampling to reduce sample complexity and updates a policy with new samples acquired from the environment interaction per gradient step. Considering that the sample complexity of an effective policy trained with a PPO algorithm increases with increasing task complexity, soft actor–critic (SAC) (Haarnoja et al., 2018), one of the remarkable off-policy algorithms, solves this issue by importing a replay buffer that stores all transition samples.

Nevertheless, these improvements are still too expensive to use in some costly domains, such as robotic manipulation and autonomous driving. Alternatively, model-based RL (Moerland et al., 2023) teaches the dynamics model of the environment from interaction experience to produce generated samples. That is, expert demonstration is regarded as effective supervision added during training, which is called learning from demonstration (LfD) (Ravichandar et al., 2020). The aforementioned studies aim to solve the credit assignment problem by formulating a Q-value to represent the expected future return, and then use it as a critic to guide the direction of policy optimization. However, a robust and accurate dynamics model is difficult to learn.

Another problem for sample efficiency is the exploration–exploitation dilemma. Given an amount of exploration, the ideal goal of bounding the expected return is unrealistic in real-world applications because the exploration degree is hard to quantify. Therefore, various heuristic strategies have been proposed to select an appropriate action to balance the degrees between exploration and exploitation throughout training. Several commonly used strategies, including optimistic exploration (Cheung et al., 2020), posterior sampling (Wang and Li, 2020), and information gain (Houthooft et al., 2016), are used to select the following action to better maintain the delicate balance. However, these methods neglect the importance of expert demonstration in exploration.

In this paper, we propose a novel and efficient sampling approach, selective sampling, by evaluating the next action to interact with the environment. Specifically, we model the internal hierarchy of environments with a clustering buffer where the expert demonstration is added with on-policy and off-policy data. Selective sampling is performed based on cluster values to improve sample efficiency. Our experiments indicate that compared with other algorithms, our proposed method can help deep RL algorithms achieve better performance on continuous locomotion tasks. Furthermore, from the application point of view, our method can achieve performance that surpasses those of other methods in the simulation environment of autonomous driving. The main contributions of this paper include the following: (1) introduction of the principle of internal hierarchy to evaluate samples with a clustering buffer during the sampling process; (2) provision of a two-stage perspective of the sampling process and integration of expert demonstration with a clustering buffer in the exploration stage.

2 Related works

In past research, many RL algorithms try to use expert data to achieve good results, which is called human–machine augmented intelligence (HAI) (Xue et al., 2022; Zhang et al., 2023a; Zhou et al., 2023). The use of expert data in past methods can be divided into expert feedback and expert data. Among them, the learning method from expert data is teaching–learning. The core motivation of teaching–learning is to use expert data to guide the two critical processes in RL, “credit distribution” and “exploration and exploitation,” in the process of interaction between the agent and the environment, so that the value network and the policy network can iterate each other under the mitigation of policy evaluation and policy improvement, until the optimum is achieved.

2.1 Guidance for value networks

In prior approaches, expert data have guided the value network. The primary motivation for using expert data to guide the value network is the hope that the value function can reasonably distribute credit to the expert’s state–action pair in the policy evaluation process. For example, Hester et al. (2018) used expert data to pre-train the

Q-value network in a deep Q-network (DQN) by introducing expert data into the replay buffer, which is called deep Q-learning from demonstration (DQfD). Vecerik et al. (2017) applied similar ideas to deep deterministic policy gradient (DDPG) and proposed DDPG from demonstration (DDPGfD). Soft actor–critic from demonstration (SACfD) (Haarnoja et al., 2018) used a similar guidance approach for Q-value networks in SAC.

However, the above methods require a large sample size, which hinders their further development. To solve the problem of high sample complexity, subsequent research naturally uses the supervision information contained in expert data more directly.

2.2 Guidance for policy networks

Expert data can guide the value network and, naturally, the policy network. The use of expert data to guide the policy network is intended to provide guidance in the early stage of policy exploration. This is achieved by using a pre-training method to give the policy a good initial point in the state–action pair space, making it closer to the expert policy. Then, the parameters of the pre-trained policy model are used as the initialization parameters of the policy model in the RL agent. For example, the SAC-based behavior cloning approach (Nair et al., 2018), the Tencent King of Glory game AI JueWu (Ye et al., 2020), the Go AI AlphaGo (Silver et al., 2016), and the StarCraft II game AI AlphaStar (Vinyals et al., 2019) all belong to this category.

2.3 Guidance for replay buffers

Expert data can directly guide the replay buffers. This method usually modifies the sampling mechanism of the interactive experience pool. Instead of random sampling, it uses a playback mechanism to preferentially sample the expert data, for example, prioritized experience replay (PER) (Schaul et al., 2016), hindsight experience replay (HER) (Andrychowicz et al., 2017), and attentive experience replay (AER) (Sun et al., 2020).

2.4 Guidance for the reward function

The expert data can guide the reward function. This is done by modeling the reward function and extracting the reward information from the expert trajectory data. This method of extracting knowledge from the heuristic information of expert trajectories is called inverse reinforcement learning (IRL). After the generative adversarial network (GAN) was proposed (Goodfellow et al., 2020), adversarial training techniques came to the attention of RL researchers. Ho and Ermon (2016) proposed generative adversarial imitation learning (GAIL), a generative adversarial network based imitation learning algorithm. GAIL generates the predicted actions of the network based on the generator and determines whether the actions are derived from expert data distribution or the network predictions based on the discriminator, which largely extracts the supervisory information contained in the expert data. GAIL achieves the best results at the time in many continuous control tasks. However, GAIL has the disadvantage of being sensitive to environmental noise. Fu et al. (2017) proposed adversarial inverse reinforcement learning (AIRL), which is more robust to the dynamic characteristics of the environment.

3 Proposed method

3.1 Preliminaries

RL usually formulates sequential decision-making problems as discounted finite horizon Markov decision processes, defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma, \mathcal{T}_0)$, where $s \in \mathcal{S}$ represents a state from a set of finite states, $a \in \mathcal{A}$ denotes an action in a set of finite actions, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the transition probability distribution, $\mathcal{T}_0 : \mathcal{S} \to \mathbb{R}$ is the initial state distribution, and $\gamma \in (0, 1)$ is a discount factor. Let π represent the current policy $\pi : \mathcal{S} \to \mathcal{A}$ and τ represent a trajectory with length L illustrated as $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ..., s_L)$; then the trajectory distribution of the current policy is described as

$$\pi(\tau) = \mathcal{T}_0 \prod_{t=0}^{L-1} \pi(a_t|s_t) \mathcal{T}(s_{t+1} \mid s_t, a_t). \tag{1}$$

The trajectory return is defined as $r(\tau) = \sum_{t=0}^{L-1} \gamma^t r(s_t, a_t)$, where t is a time step. To assign credit, the Q-value function used to evaluate the state and action is formulated as

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}} \left[Q(s_{t+1}, a_{t+1}) \right]. \tag{2}$$

Thus, the objective of the expected policy return is

$$\mathcal{J}(\pi) = \mathbb{E}_{\tau\sim\pi(\tau)}\left[r(\tau)\right] = \mathbb{E}_{s_0\sim\mathcal{T}_0, a_0\sim\pi(\cdot|s_0)}\left[Q(s_0,a_0)\right]. \tag{3}$$

In SAC (Haarnoja et al., 2018), the function approximation of the Q-value with parameter θ is updated toward the target Q-value by using transitions sampled uniformly from the replay buffer $(s_t, a_t, r_t, s_{t+1}) \in \mathcal{D}$ to minimize the Bellman residual $\mathcal{B}(Q)$:

$$\mathbb{E}_{\mathcal{D}}\left[Q_\theta(s_t,a_t) - r_t - \gamma\mathbb{E}_{a\sim\pi_\phi(\cdot|s_{t+1})}[Q_{\theta^-}(s_{t+1},a)]\right]^2, \tag{4}$$

where θ^- and ϕ are the parameters of the target Q-network and current policy, respectively. As in expression (4), $a \sim \pi_\phi(\cdot \mid s_{t+1})$ is the formalism of the sampling process of the policy. We can explicitly model the policy network's output with a Gaussian distribution. In the implementation, the input to the policy network is s_{t+1}, and the output is a Gaussian distribution with mean function μ and variance function σ, which together form the action distribution. Normally, the reparameterization trick (Kingma and Welling, 2014) is employed to sample a continuous action from this Gaussian distribution, shown as

$$a = \mu(s_{t+1}) + \epsilon\sigma(s_{t+1}),\ \epsilon \sim \mathcal{N}(0, \mathcal{I}). \tag{5}$$

For the delicate balance of exploration and exploitation, the well-known optimistic exploration method, upper confidence bound (UCB) (Bellemare et al., 2016), argues that new states are potentially good, so we should encourage the agent to choose an action leading to unknown states, formalized as

$$a = \arg\max \mu(s_{t+1}) + \sqrt{\frac{2\ln L}{N(a)}}, \tag{6}$$

where $N(a)$ is a count-based function to record the visiting frequency for action a.

3.2 Overview of the algorithm structure

In this subsection, we introduce our proposed approach, selective sampling, in detail. Our goal is to model the internal hierarchy of the environment by clustering methods on the replay buffer and introduce expert data for guidance. Furthermore, selective sampling is integrated with SAC because it achieves better performance on sample complexity than on-policy algorithms. For better understanding, we illustrate two stages of the sampling process, target-Q stage and exploration stage, within deep RL in Fig. 1.

In the exploration stage, we first concatenate the on-policy data, off-policy data, and expert data, and subsequently perform clustering operations on these data. The process divides many high-dimensional data, which spread throughout the state–action pair space, into several clear hierarchical clusters. At the same time, the clustering process could obtain a classifier that can classify any of the state–action pairs in the space. From this perspective, this process models the internal

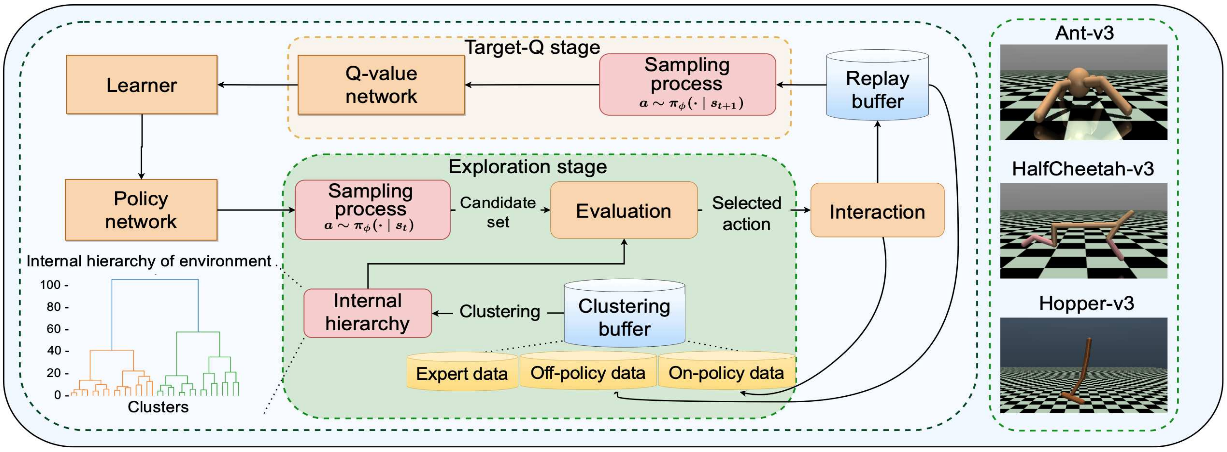


Fig. 1 Structure of selective sampling. Integrated with soft actor–critic (SAC), which determines the target-Q stage, learner, and policy network, selective sampling with a clustering buffer consists of on-policy data, off-policy data, and expert data that model the internal hierarchy of three environments. It takes effect mainly in the sampling process of the exploration stage to evaluate the candidate set for the selected action

hierarchy of the environment. After the above modeling, we use the model as an evaluation criterion of state–action pairs from the candidate set to select an appropriate action. The candidate set is composed of sampling data at the current state denoted as $\mathcal{D}_{\mathrm{s}} = \{(s_t, a_1), (s_t, a_2), ..., (s_t, a_m)\}$, where sampling data are acquired by using the reparameterization trick in Eq. (5) from the sampling process of policy $a_i \sim \pi_\phi\ (\cdot \mid s_t)$ isotropically, as noise ϵ is sampled from the standard Gaussian distribution.

It is noticeable that most existing methods mentioned above (Houthooft et al., 2016; Cheung et al., 2020; Wang and Li, 2020) lack explicit expert guidance, and the selection of samples is based on the principle of optimism, posterior, and information, rather than the internal environmental hierarchy, which is more valuable to the subsequent action. Unlike the evaluation process of the target-Q stage, where the Q-value function is used for evaluation, we model the expected return of the state–action pair of the current policy toward maximizing cumulative rewards. Then, the Q-function approximator is updated by experience algorithms such as HER (Andrychowicz et al., 2017), PER (Schaul et al., 2016), and AER (Sun et al., 2020). One advantage of this is that some rarely visited but valuable states in the replay buffer do not affect the overall performance of the current policy. However, the target-Q stage requires an enormous replay buffer containing millions of samples for evaluation to guide the current policy. With sample transition denoted as (s_t, a_t, r_t, s_{t+1}), the target Q-value is formalized as follows:

$$y_t = r_t + \gamma \min_{i=1,2} Q_{\theta_i^-}\left(s_{t+1}, \pi_\phi(\cdot \mid s_{t+1})\right). \quad (7)$$

3.3 Algorithm details

As illustrated in Algorithm 1, a clustering buffer is added with expert demonstration to use the advantages of the Q-value function and alleviate sample complexity. We employ clustering methods to model the internal hierarchy of the environment as an evaluation criterion for the state–action pair in the candidate set. Let $\mathcal{D}_{\mathrm{Q}}$ represent the replay buffer that updates the Q-value function in SAC. Typically, $\mathcal{D}_{\mathrm{Q}}$ contains all experience and is sampled uniformly.

Algorithm 1 Selective sampling

```
Require: on-policy data $\mathcal{D}_{\mathrm{on}}$, off-policy data $\mathcal{D}_{\mathrm{off}}$, expert data $\mathcal{D}_{\mathrm{E}}$
Ensure: learned policy $\pi_\phi(a \mid s)$
Initialize network parameters $\theta$, $\theta^-$, $\phi$
for each environment step do
    $\mathcal{D}_{\mathrm{s}} \leftarrow a \sim \pi_\phi(\cdot \mid s_t)$
    $\mathcal{D}_{\mathrm{c}} \leftarrow (\mathcal{D}_{\mathrm{on}}, \mathcal{D}_{\mathrm{off}}, \mathcal{D}_{\mathrm{E}})$
    $N_{\mathrm{c}} \leftarrow \mathrm{AgglomerativeClustering}(\mathcal{D}_{\mathrm{c}})$
    $\mathcal{C} \leftarrow K\text{-means}(\mathcal{D}_{\mathrm{c}}, N_{\mathrm{c}}, \mathcal{D}_{\mathrm{s}})$
    for each $c$ in $\mathcal{C}$ do
        $V_\eta(c) \leftarrow \frac{1}{n_h} \sum_{k=i}^{i+n_h-1} \eta(s_k, a_k)$
    end for
    $a_{\mathrm{selected}} \leftarrow \mathrm{Random}(\mathrm{Softmax}_{c \in \mathcal{C}} V_\eta(c))$
    $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_{\mathrm{selected}})$
    $\mathcal{D}_{\mathrm{Q}} \leftarrow \mathcal{D}_{\mathrm{Q}} \cup \{s_t, a_{\mathrm{selected}}, r_t, s_{t+1}\}$
    for each gradient step do
        $\theta \leftarrow \theta - \lambda_{\mathrm{Q}} \nabla_\theta \mathcal{B}_\theta(Q)$
        $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi \mathcal{J}_\phi(\pi)$
        $\theta^- \leftarrow \lambda_{\mathrm{target}} \theta + (1 - \lambda_{\mathrm{target}}) \theta^-$
    end for
end for
```

With the explicit reward function, it would be easy to evaluate each sample based on the confidence $C(a) = r(s_t, a)$ of action in the collected candidate set, followed by selecting the action with the highest confidence. However, in most situations, there is no explicit reward signal for different actions in the current state without interacting with the environment, and this type of situation can be regarded as the problem of implicit reward. Therefore, one feasible way to solve this issue is to consider clustering methods as an ideal similarity measure to quantify how good an action is in the candidate set. We call it the implicit value evaluation operation. To quantify the goodness, we introduce a clustering buffer $\mathcal{D}_{\mathrm{c}}$, consisting of state–action pairs from on-policy data, off-policy data, and expert data, denoted as $\mathcal{D}_{\mathrm{on}}$, $\mathcal{D}_{\mathrm{off}}$, and $\mathcal{D}_{\mathrm{E}}$, respectively.

These three kinds of data have their own roles to play in $\mathcal{D}_{\mathrm{c}}$. First, an appropriate proportion of expert data allows the policy network to be guided when sampling from the action candidate set, accelerating convergence and improving network performance. Second, the on-policy data are helpful in improving the accuracy of value evaluation because they ensure unbiasedness for the current policy, and compensate for possible biases introduced during training. Using the on-policy data can help the policy converge to the optimal space faster while using the expert data for directed exploration. Third, the off-policy data can improve sample efficiency. The cooperation of the above three kinds of data enables selective sampling to achieve good results.

Clustering methods are used to discover the internal structure of the data itself. Compared with neural-network-based supervised classification methods, they have no labels, are simple to implement, and are easy to use. Among the latest clustering methods, the more representative ones are deep clustering methods that combine representation learning with unsupervised learning fine-tuning, for example, deep embedded clustering (DEC) (Xie et al., 2016). Huang et al. (2023) proposed ProPos. This work combines the advantages of contrastive and non-contrastive representation learning to deal with the class collision issue and the collapse of clustering in current deep clustering methods. Niu et al. (2022) proposed SPICE, a semantic pseudo-labeling-based image-clustering framework that divides the clustering network into a feature model for measuring instance-level similarity and a clustering head for identifying cluster-level discrepancy. Also, some approaches try to eliminate the multi-stage module and use a single network for deep clustering. For example, Niu et al. (2020) proposed GATCluster, a self-supervised Gaussian attention network that eliminates the extraction of intermediate features followed by traditional clustering algorithms and instead outputs semantic clustering labels directly without further post-processing.

Selective sampling improves the sampling efficiency of RL by (1) selectively sampling the actions used for exploration and (2) introducing expert demonstrations. To achieve this, we introduce clustering methods to model the replay buffer consisting of three kinds of data as an easy-to-use action value assessment criterion. For implementation difficulty and model reuse feasibility, we use agglomerative clustering (Murtagh and Legendre, 2014) and K-means instead of the deep clustering approach. Agglomerative clustering is a clustering method that constructs data into a hierarchical structure by gradually merging clusters. Compared with other clustering algorithms, which assign data to a fixed number of clusters, agglomerative clustering creates a hierarchy between clusters. Also, it does not require a prior determination of the number of clusters, and the results can be visualized and analyzed by a tree diagram, as shown in Fig. 1. In the beginning, we regard each state–action pair in the clustering buffer as a cluster c_i to model the internal hierarchy of the environment by minimizing the change in variance, where $\|\cdot\|_1$ denotes the L_1 norm and $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm:

$$\delta(c_1, c_2) = \frac{\|c_1\|_1 \|c_2\|_1}{\|c_1\|_1 + \|c_2\|_1} \|c_1 - c_2\|_{\mathrm{F}}. \tag{8}$$

As a result, we acquire the number of clusters N_{c} on the clustering buffer. Then we employ the K-means algorithm to classify the clustering buffer in the cluster set as $\mathcal{C}$. During the evaluation of each cluster, we use the assessment function η of the state–action pair to evaluate the cluster. Let $h \in \mathcal{C}$ represent one cluster of the cluster set containing n_h transitions: $(s_i, a_i, r_i, s_{i+1}), \cdots, (s_{i+n_h-1}, a_{i+n_h-1}, r_{i+n_h-1}, s_{i+n_h})$. Then the cluster value is

$$V_\eta(c) = \frac{1}{n_h} \sum_{k=i}^{i+n_h-1} \eta(s_k, a_k). \tag{9}$$

Here, the assessment function η, as a reward, provides confidence in state–action pairs such as the reward function, Q-value function, or target-Q value.

Similarly, we employ the K-means algorithm on cluster $\mathcal{D}_{\mathrm{s}}$ and perform the implicit value evaluation operation for each of these clusters. Specifically, we use the model to classify $(s_i, a_i) \in \mathcal{D}_{\mathrm{s}}$, and evaluate the value of the corresponding (s_i, a_i) based on the mean reward of the cluster in $\mathcal{D}_{\mathrm{c}}$. After that, we use the value of each (s_i, a_i) to evaluate the value of the clusters in $\mathcal{D}_{\mathrm{s}}$. Finally, we use softmax selection to use the value of each cluster in $\mathcal{D}_{\mathrm{s}}$ as the selection probability. This soft selection allows the clustering buffer to be more fully used for those samples whose value function is underestimated, but actually has a higher value, described as

$$a_{\mathrm{selected}} = \mathrm{Random}\left(\mathrm{Softmax}_{c \in \mathcal{C}} V_\eta(c)\right). \tag{10}$$

The selected action contains some similarities to the expert policy and then affects the subsequent state–action distribution of the policy. Consequently, the replay buffer experience used to recover the Q-value function would be more promising.

4 Experiments

4.1 Mujoco

Experiments were performed on five challenging continuous control tasks of the benchmark suite

Mujoco (Brockman et al., 2016) to evaluate the performance of selective sampling. Compared with the original SAC algorithm, under different environments and configurations, we focused mainly on the sample complexity and convergence performance in five Mujoco environments, including Hopper-v3, Ant-v3, HalfCheetah-v3, Reacher-v2, and Walker2d-v3. In this subsection, we will first introduce our experimental environments and implementation. After that, we will present the results and discussion.

4.1.1 Environments and implementation

The goal of the Reacher-v2 task is to move the robot's end effector close to a target spawned at a random position. Moreover, the four other locomotion tasks aim to move forward as quickly as possible. In these environments, the state represents position, velocity, and acceleration, while the action space encapsulates joint forces or torques to control the system. The dimensions of these states and actions are listed in Table 1.

Table 1 Environmental dimensions

Task	Dimension	
	State	Action
Hopper-v3	11	3
HalfCheetah-v3	17	6
Ant-3	111	8
Reacher-v2	11	2
Walker2d-v3	17	6

Each experiment with a specified configuration was initialized with 10 random seeds to guarantee stability and to average the test episode reward during training for evaluation. Beginning with random exploration, the learning process was launched after 50 000 steps. Then, every four steps, we proceeded with a gradient update for the Q-network and policy network with policy delay, where the Adam optimizer was initialized with a 3×10^{-4} learning rate, and a batch size of 512 was used to perform the update. Meanwhile, to ensure the stability of the clustering buffer, the clustering buffer was updated every 1000 steps. In the Hopper-v3, Ant-v3, and HalfCheetah-v3 environments, we conducted ablation experiments by adjusting the ratio of on-policy, off-policy, and expert data that composed the clustering buffer. Specifically, the on-policy degree was 2048, which measures how many of the latest interaction samples lie in the on-policy buffer. With the size of the clustering buffer fixed at 4096, the 112-configuration in our methods means that 1024 on-policy state–action pairs were sampled from the on-policy buffer (size of 2048), 1024 off-policy state–action pairs were sampled from the replay buffer, and 2048 expert state–action pairs were sampled from the expert buffer. In addition, data of the expert buffer were generated by running the expert policy trained with TRPO (Schulman et al., 2015) in place of human demonstration in reality. Similarly, we performed a 211-configuration of selective sampling in our experiments.

4.1.2 Results and discussion

Fig. 2 shows that our method learned rapidly at first and achieved better performance and higher sample efficiency. In addition, as shown in Fig. 2c, the ablation study demonstrates the importance of introducing expert data into the clustering buffer because the model with added expert data converged to better performance. In particular, when only off-policy clustering was used as an evaluation criterion, the ablation study struggled like the SAC algorithm. The above results suggest that our method can exploit valuable supervision in the form of an internal hierarchy from the clustering buffer in the exploration stage.

Based on the results of the ablation experiments with the 112-configuration and the 211-configuration in different environments, we found that in the HalfCheetah-v3 environment, the 112-configuration worked better than the 211-configuration, but in the Ant-v3 and Hopper-v3 environments, the results were reversed. This phenomenon occurs because in different environments, the agent's skeletal structure is different, resulting in different proportional derivative (PD) control of the agent's motion. Specifically, in the Hopper-v3 environment, the agent's skeletal structure is the simplest. In the Ant-v3 environment, the agent is a quadrupedal structure, which allows the agent to remain stable at the beginning and also makes it easier for the agent to move forward to obtain the reward. In contrast, in the HalfCheetah-v3 environment, the agent is a bipedal structure, and its front and rear foot structures are different. At the same time, its rear foot has the highest degree of freedom among all the agents above, making it the most difficult to walk stably. As we discussed

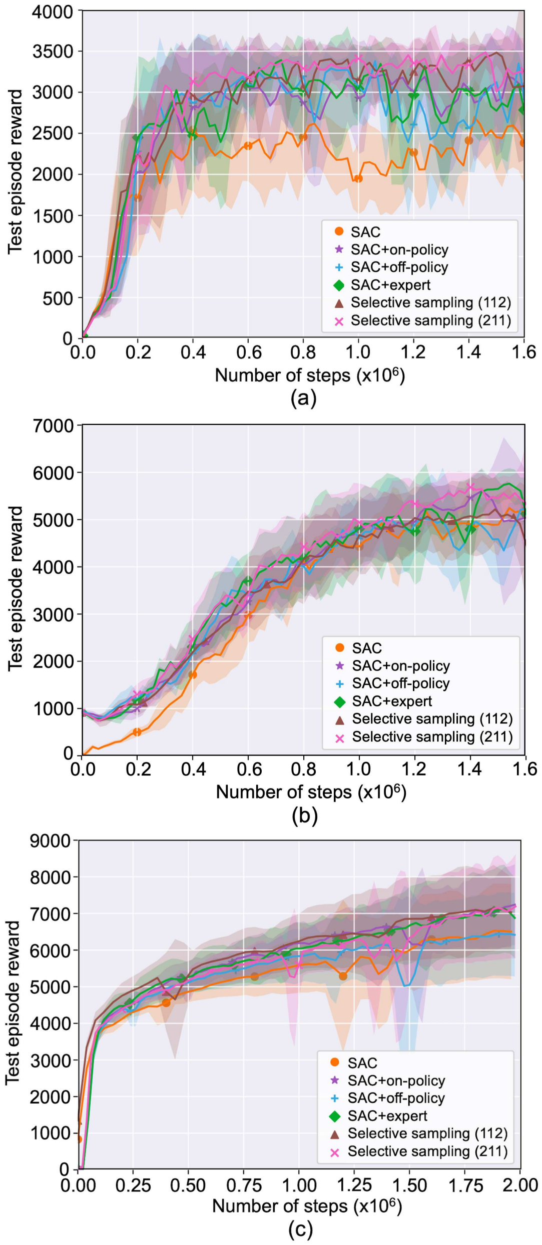


Fig. 2 Training curves in three environments of continuous control tasks: (a) Hopper-v3; (b) Ant-v3; (c) HalfCheetah-v3. For these plots, the x axis denotes the number of interaction steps, which reflects the sample complexity. The solid curves correspond to the mean and the shaded region to the range of one standard deviation from the mean episode reward over 10 random seeds. We average the value taken from each point itself with the value taken from the next point as the final value for the current point. The SAC+on-policy setting refers to constructing the clustering buffer using only the on-policy data, while the SAC+off-policy setting and the SAC+expert setting are defined similarly. The numbers in selective sampling (112, 211) indicate the proportions of on-policy, off-policy, and expert data in the clustering buffer, separately

earlier, expert data allow the policy network to be guided when selecting actions from the action candidate set. When the structure of the agent makes it more difficult to obtain rewards, the reliance on expert guidance is also higher to some extent. This explains why the conclusion as to which configuration will achieve the best performance will be different in different environments.

As shown in Figs. 2a and 2b, in the Hopper-v3 and Ant-v3 environments, the 211-configuration outperformed the other settings. Similarly, in the ablation study of the HalfCheetah-v3 environment, as shown in Fig. 2c, the 112-configuration outperformed the on-policy and expert data setup. Meanwhile, the 211-configuration, although slightly worsened at the beginning of the training, was able to reach the same level as the first three at the end of the training. In other words, the setup containing all the three kinds of data has the potential to outperform all other setups while staying caught up in the slightly worse cases. Therefore, we believe that a setup with three kinds of data is more general. It illustrates that better performance requires the interplay of all three kinds of data, as discussed above. Note that the 112-configuration and the 211-configuration had significantly better performance than the baseline in the Hopper-v3 and HalfCheetah-v3 environments.

To give more insight into how each of the on-policy data, off-policy data, and expert data influences the performance of selective sampling, we supplemented ablation experiments without one of the three kinds of data in the HalfCheetah-v3 environment, as shown in Fig. 3a.

We can see that the performance of the 112-configuration was still optimal, and that the model performance decreased after removing any of the three kinds of data. At the early stage of training, the 112-configuration, the SAC+on-policy+expert setting, and the SAC+off-policy+expert setting converged similarly faster than the other settings between 0 and 2×10^5 steps, because they had the same amount of expert data. This indicates that more expert data can guide the model to converge faster in the early stage. However, the convergence speeds of the SAC+on-policy+expert setting and the SAC+off-policy+expert setting decreased in the subsequent training phase. This illustrates that in the absence of one kind of data, the advantage brought by expert data in the early stage of training

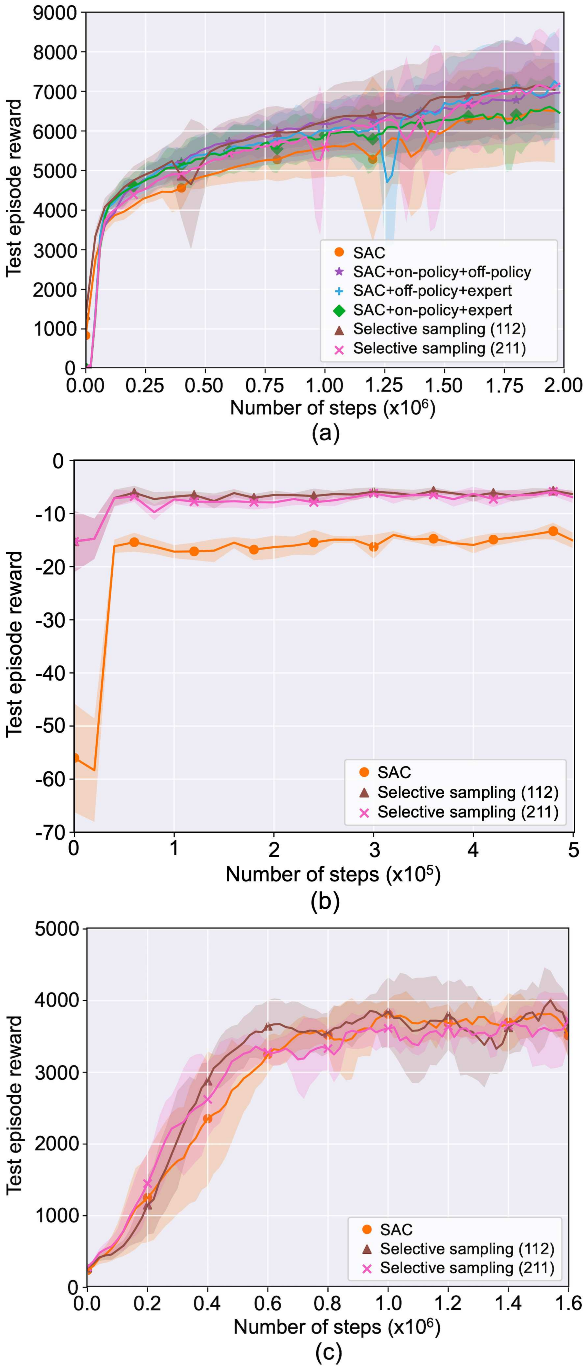


Fig. 3 Training curves in three environments of continuous control tasks: (a) HalfCheetah-v3; (b) Reacher-v2; (c) Walker2d-v3. For these plots, the x axis denotes the number of interaction steps, which reflects the sample complexity. The solid curves correspond to the mean and the shaded region to the range of one standard deviation from the mean episode reward over 10 random seeds. The SAC+on-policy+off-policy setting refers to constructing the clustering buffer using the on-policy and off-policy data. The SAC+off-policy+expert setting and the SAC+on-policy+expert setting are defined similarly

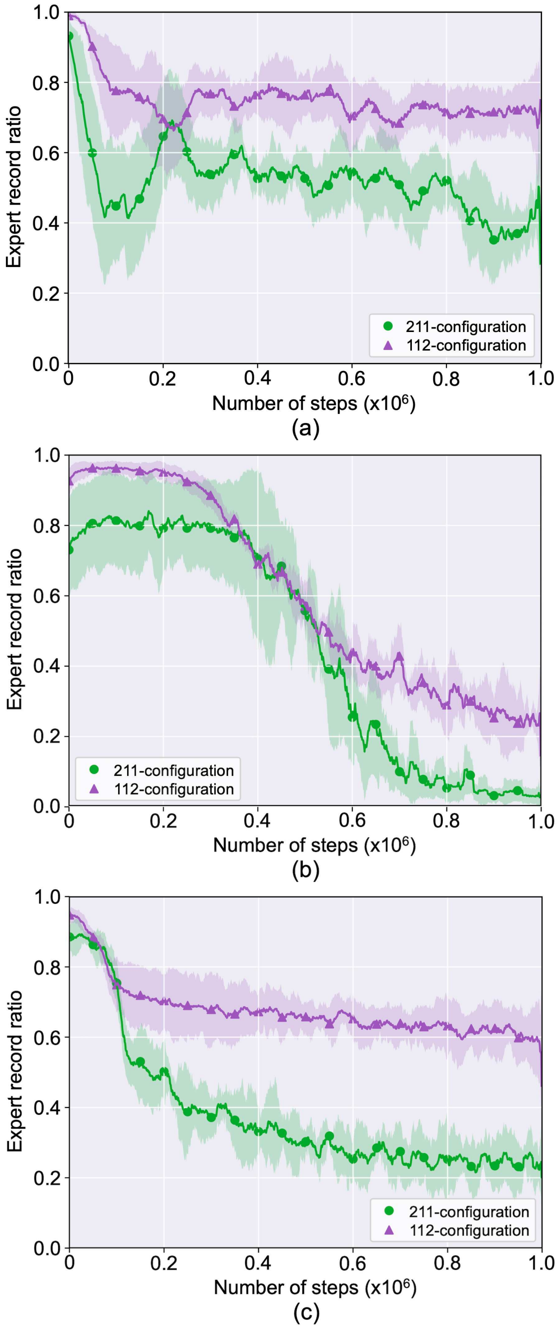
may not be maintained. In addition, although the SAC+on-policy+off-policy setting achieved similar performance to the 112-configuration in the middle part of the training, it converged slower than the 112-configuration in the early stage and had worse performance in the final stage than both the 112-configuration and 211-configuration. This experiment again illustrates that better performance requires interplay among all three kinds of data.

To further demonstrate the effectiveness of selective sampling, we conducted experiments with the 112-configuration and 211-configuration in two additional Mujoco environments, Reacher-v2 and Walker2d-v3.

As shown in Fig. 3b, in the Reacher-v2 environment, both the 112-configuration and 211-configuration converged faster initially and eventually to a better level. The lower difficulty of this task led to similar results for both configurations. The reason for the different starting heights is that the first point was drawn at the 20 000$^{\text{th}}$ step in our implementation. As shown in Fig. 3c, in the Walker2d-v3 environment, both the 112-configuration and 211-configuration achieved faster initial convergence compared to the baseline. The 112-configuration had the best performance finally. The above results showed that selective sampling can improve model performance in these two environments, making the previous conclusions more convincing.

To show more details on the clustering buffer, we provide the trend of the ratio of expert data in the clustering buffer $\mathcal{D}_{\mathrm{c}}$ over time in the three Mujoco environments, as shown in Fig. 4.

The percentage of expert data was high at the beginning in all the three environments. It is because the rewards of expert data were much larger than those of the two other kinds of data in the early stage, and the positions of expert data were closer to each other in the feature space. As a result, when we selected the top five clusters with the highest mean reward, we will naturally select several clusters mainly composed of expert data. In addition, in the 112-configuration, the percentage of expert data in the clustering buffer was twice as large as that in the 211-configuration. We can also see that this advantage can be maintained until the convergence stage in all the three environments, indicating that our settings for the percentage of expert data in the clustering buffer were consistently effective. This advantage was more beneficial for a task like HalfCheetah-v3, which is significantly dependent on expert demonstration. This explains why it performed better in the 112-configuration.



(a)

(b)

(c)

Fig. 4 The trend of the ratio of expert data in the clustering buffer D_c over time in the three environments of Mujoco, using the 112-configuration and 211-configuration: (a) Hopper-v3; (b) Ant-v3; (c) HalfCheetah-v3. The numbers in the 112- and 211-configuration indicate the proportions of on-policy, off-policy, and expert data in the clustering buffer, separately. For these plots, the x axis denotes the number of interaction steps. We select the top five clusters with the highest mean reward in the clustering buffer D_c. The solid curves correspond to the mean and the shaded region to the range of one standard deviation from the mean value over five clusters

To summarize, the results demonstrated the importance of introducing expert data into the clustering buffer, and our method can exploit valuable supervision from it. The amount of expert data is crucial, but it also needs to be specifically analyzed with the specific environment. When using selective sampling, our suggestion is to use all the three kinds of data, which can be more general. In particular, in some environments where there is a high dependency on expert demonstration, we can consider increasing the proportion of expert data. There are four main linkage criteria in agglomerative clustering. The ward setting minimizes the variance of the clusters being merged. It has been used in previous experiments. The average setting uses the average distances of each observation of the two sets. The complete setting uses the maximum distance between all observations of the two sets, and the single setting uses the minimum distance. To explore which linkage criterion is the best, we conducted supplementary ablation experiments in the HalfCheetah-v3 environment, with one training session for each linkage criterion setting and 1×10^6 steps for each training session. Table 2 shows the test episode reward that the model can obtain as training proceeds using different linkage criteria.

Table 2 Ablation study of linkage criteria

Stage	Number of steps ($\times 10^6$)	Test episode reward			
		Ward	Average	Complete	Single
Early	0.1	**3197**	909	1003	938.8
	0.2	**3968**	3167	1206	1295
	0.3	4539	**4603**	2810	3367
Middle	0.5	5188	**5658**	4363	4216
	0.6	5651	**5874**	4401	4509
Final	1.0	6622	**7645**	4947	5252

The bold number indicates which linkage criterion receives the greatest test episode reward at a particular moment

The ward setting had the highest initial convergence speed, significantly ahead of the other settings. The average setting was the second, and the single and complete settings the lowest. However, the average setting can catch up to a level similar to the ward setting, at around 3×10^5 steps, and performed better than the ward setting after 5×10^5 steps.

In selective sampling, a different linkage criterion causes a different number of clusters for the K-means method, which affects the final structure of the clustering buffer. The following conclusions can be drawn: among the four linkage criteria, the ward setting achieves the best balance between the initial speed and the later performance. The average setting sacrifices part of the initial speed but can obtain better performance in the later period. The performances of single and complete settings are similar to that of the original SAC algorithm. According to the application requirements, the implementation of selective sampling can be considered in the choice between the ward setting and the average setting.

4.2 LGSVL

Autonomous driving is an essential application of RL, often involving a significant amount of data (Zhang et al., 2023b), which makes sample efficiency critical. To further evaluate the performance of selective sampling, we performed full experiments on an end-to-end task of autonomous driving on the LGSVL (SVL Simulator by LG) simulation platform (Rong et al., 2020). LGSVL is an open-source simulation platform developed by LG for autonomous driving testing. Its implementation of the Python application programming interface (API) enables us to easily adjust to maps, vehicles, sunlight, and weather to obtain large-scale, diverse, and customized autonomous driving testing scenarios. In our experiments, we used the Python API to set up the experimental task, providing the environment abstraction needed for selective sampling. The version of LGSVL used here was svlsimulator-linux64-2021.2.2.

In this subsection, we will first introduce the specific task details and the experimental environments. As an important part of the task setup, we will introduce the reward function settings. After that, we will present the experimental implementation. At the end of this subsection, we will show the results and discuss the results of the ablation experiments.

4.2.1 Task details

The current implementation of autonomous driving systems is divided into two main technical routes: modular and end-to-end autonomous driving systems. The modular autonomous driving system is artificially divided into three major modules: perception, planning, and control. In contrast, the end-to-end autonomous driving system uses a single model as a direct solver from sensor input to controller output. Considering that the focus of this paper is on RL with high sampling efficiency rather than the coordinated cooperation of different algorithms among different modules, we set an end-to-end autonomous driving task.

We set up the experiments in the “Shalun” map provided by the LGSVL simulation platform, a digital twin map of a realistic closed test site for autonomous vehicles built by Taiwan CAR Lab in Guiren District, Tainan City, China. As shown in Fig. 5a, this autonomous vehicle test site was designed specifically for Asian road conditions and contains rich map elements: straight lanes, curved lanes, intersections, traffic signals, circular lanes, tunnels, railroad level crossings, etc. The section of road shown in Fig. 5b was selected as the task scope for our experiments. There was a composite road, which consisted of a straight road and a curved road.

Here we call the vehicle equipped with our autonomous driving system the Ego Car. The specific task set in this subsection is to train an autonomous driving system to be able to compute the specific control signals that the Ego Car should take at the current moment to perform safe, legal, start-to-finish driving, using the signals from the front camera mounted on the Ego Car. Making the whole task more difficult, the initial starting point of the Ego Car (yellow “Start” in Fig. 5b) was placed in the right lane of a straight section of that road, while the endpoint (red “End” in Fig. 5b) was placed in the left lane of a curved section of that road. In other words, the automated driving system needed to drive the Ego Car along the lane in a straight line at the beginning, change to the left lane at the appropriate time, and drive along the curve of the lane near the endpoint. No collision with obstacles and no illegal driving behavior were permitted. Finally, the Ego Car arrived at the endpoint successfully. Overall, this composite autonomous driving task contained

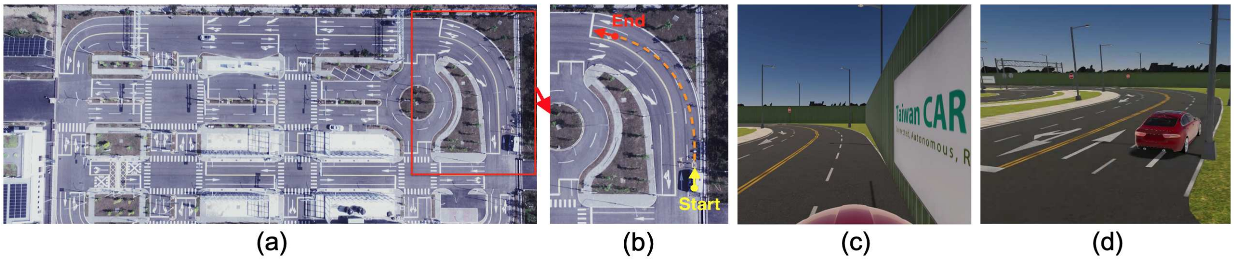


Fig. 5 Details in the LGSVL experiment: (a) full-size bird-eye view of Shalun; (b) experimental range we choose to perform our experiments, where the yellow point represents the starting point of the Ego Car and the red point represents the end point of the Ego Car; (c) front camera view of the Ego Car; (d) overhead view of the Ego Car. References to color refer to the online version of this figure

several sub-tasks with a certain degree of difficulty and training significance.

4.2.2 Environments

In the experiments, we still used the Open AI Gym (Brockman et al., 2016) RL environment abstraction format, including state, action, and reward functions.

1. State

We used the image from the front view camera mounted on the front of the Ego Car as the state representation of the environment. Specifically, at any moment, it is a 1920×1080 image in JPEG format, divided into three RGB channels, with one pixel under each channel taking values in the range $[0, 255]$. For ease of computation and to compress the state dimension, we compressed this original image to a $128 \times 128 \times 3$ matrix representation.

2. Action

We defined the output actions of the policy network in the reinforcement learning model as a 1×2 vector, where each dimension corresponds to the steering wheel and the pedal operations of the Ego Car in the LGSVL simulation platform. First, we implemented the steering operation with the first dimension, which takes values in the range of $[-1, 1]$. The value -1 represents the maximum degree of steering wheel leftward rotation applied, the value 0 represents the steering wheel maintaining straight-forward operation, and the value 1 represents the maximum degree of steering wheel rightward rotation applied. The rest of the values are applied to the corresponding degree of steering wheel leftward or rightward rotation according to the absolute value of the variable. Second, we implemented the throttle pedal and the brake pedal using the second dimension, which still takes values in the range $[-1, 1]$. If this dimension takes values in $(0, 1]$, the brake pedal application degree will be set to 0, and the throttle pedal application degree is set according to the variable. Specifically, 1 represents the maximum application degree of the throttle pedal, and a value closer to 0 represents a smaller application degree. Similarly, if this dimension is in $[-1, 0)$, the degree of application of the throttle pedal will be set to 0, and the degree of application of the brake pedal is set according to the absolute value of the magnitude. Specifically, -1 represents the maximum degree of application of the brake pedal, and a value closer to 0 represents a smaller degree of application.

3. Reward function

The reward function is usually set from some prior human knowledge for safe driving, such as we cannot collide with other things, we cannot violate traffic rules, and we need to drive to the specified target. An appropriate reward function setting can significantly improve the convergence speed of the RL network and the final model performance. The reward function setup in the experiments was divided into four parts: risk-driving behavior penalties, bad-driving behavior penalties, moving forward rewards, and goal rewards. The specific implementation is summarized in Table 3.

We gave a one-time penalty of -30 and terminated the current epoch when we detected collisions with other elements of the map (located on the right side of the Ego Car) and when driving across double yellow lines into the opposite lane (located on the left side of the Ego Car). In addition, we calculated the vertical projection distance (in meters) between the current Ego Car position and the road centerline at each step, multiplied it by a factor of -1.4, and

Table 3 Reward function settings

Category	Operation	Value	Ratio
Collision	End epoch, one-time penalty	−30	1.0
Retrograde	End epoch, one-time penalty	−30	1.0
Centerline departure	Give by step	Euclidean distance	−1.4
Moving forward	Give by step	Euclidean distance	2.4
Nearing goal (≤5 m)	Give by step (≤10 steps)	Reward assignment	1.0 (decrease), −0.5 (expand)
Goal (≤2 m)	End epoch, one-time reward	Reward assignment	1.0

counted it in the reward function. We also calculated the distance (in meters) traveled by the Ego Car for this step, multiplied it by 2.4, and counted it in the reward function. In the goal detection part, we added a well-designed reward calculation method, which was divided into four parts.

First, we designed the reward credit assignment within 5 m near the endpoint, described as

$$r_t^{\text{goal}} = \frac{400}{\|\boldsymbol{X}_t^{\text{goal}} - \boldsymbol{X}_t^{\text{ego}}\|_{\text{F}} + 1}, \tag{11}$$

where $\boldsymbol{X}^{\text{goal}}$ represents the coordinates of the endpoint and $\boldsymbol{X}^{\text{ego}}$ represents the coordinates of the Ego Car. The left part of the denominator calculates the Euclidean distance between the two coordinates at this step. The visualization of the reward is shown in Fig. 6, where the reward obtained by Ego increased faster when Ego got closer to the endpoint.

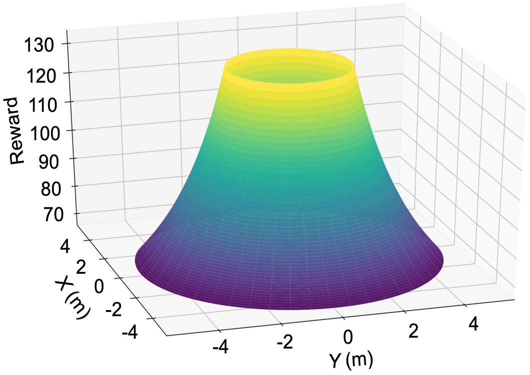


Fig. 6 Reward credit assignment

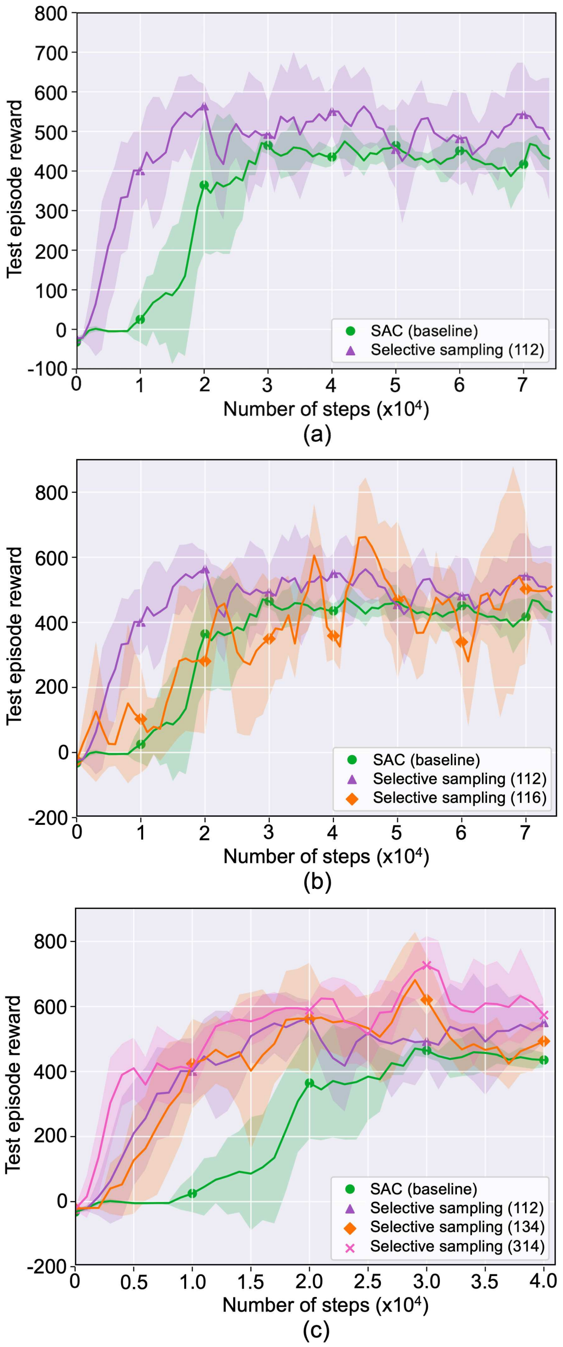
Second we judged whether the distance was decreasing or expanding, multiplied the corresponding factor of 1.0 or −0.5, and counted it in the reward function. If not, the Ego Car will receive a huge cumulative reward by passing through an area ≥2 m and ≤5 m from the endpoint, which is not what we expect. Third, we calculated the number of steps to obtain the credit assignment reward, and terminated the current epoch after reaching the limit number of 10. If not, the Ego Car will stop 2 m away from the endpoint to obtain a huge cumulative reward, which is not what we expect. Fourth, we judged that the goal was reached when it reached 2 m near the endpoint and terminated the current epoch.

4.2.3 Implementation

In this part, each experiment with a specified configuration was initialized with 10 random seeds. The initial random exploration was 2048 steps. The Adam optimizer was initialized with a 3×10^{-4} learning rate, and a batch size of 256 was used to perform gradient update for the Q-network and policy network. The clustering buffer was updated every 200 steps. We also conducted ablation experiments by adjusting the ratio of on-policy, off-policy, and expert data. Specifically, the on-policy degree was 1200, which measures how many of the latest interaction samples were in the on-policy buffer. With the size of the clustering buffer fixed at 2048, the 112-configuration in our methods means that 512 on-policy state–action pairs were sampled from the on-policy buffer, 512 off-policy state–action pairs sampled from the replay buffer, and 1024 expert state–action pairs sampled from the expert buffer. Similarly, we performed the 116-configuration, 134-configuration, and 314-configuration of selective sampling in our experiments. In addition, expert buffer data were generated by running the expert policy trained with SAC in place of human demonstration in reality.

4.2.4 Results and discussion

Fig. 7a shows that our approach provided better guidance signals for the policy network, allowing the model to converge faster and achieve better performance and higher sample efficiency than the



(a)

(b)

(c)

Fig. 7 Training curves of three different configurations: (a) baseline and 112; (b) baseline, 112, and 116; (c) baseline, 112, 134, and 314. For these plots, the x axis denotes the number of interaction steps, which reflects the sample complexity. The solid curves correspond to the mean, and the shaded region corresponds to the range of one standard deviation from the mean episode reward over 10 random seeds. The numbers in selective sampling indicate the proportions of on-policy, off-policy, and expert data in the clustering buffer, separately

baseline. A higher convergence value means that the Ego Car will have a lower speed before reaching the goal to obtain the nearing goal reward for more steps, which is exactly what we expect.

In addition, in the 116-configuration experiments as shown in Fig. 7b, we found that settings too biased toward expert data introduced greater volatility. This is because too many expert data crowd out the sampling of on-policy and off-policy data during training, and the large difference between the current policy and expert policy can cause instability.

According to Fig. 7c, the 314-configuration outperformed the 134-configuration. This is because the autonomous driving task is much more complex in terms of the environment state compared to the previous experiments. Therefore, it relies more on the on-policy data, which can eliminate the bias.

The number of steps required to reach the same level, which means the test episode reward was ≥ 450, was reduced from 3×10^4 to 1.6×10^4. It was a 46.7% reduction. Considering the additional computation caused by our method, we calculated the actual time cost of both approaches. In terms of training time, the time required to reach the same level was reduced by 28.5% (from an average of 10 768.25 s to 7701.4 s). This result shows that our approach is still practical for application scenarios that require high algorithm generalizability, such as autonomous driving.

5 Conclusions

We propose to separate on-policy, off-policy, and expert demonstration into a clustering buffer and then use the selective sampling model and the internal environment hierarchy as an evaluation criterion. Our experimental results demonstrate that sample complexity is improved for faster convergence and better prediction performance on continuous locomotion tasks, which makes deploying the RL algorithm more practical for the new environment. For future studies, we will investigate a more suitable structure for retaining the internal hierarchy of clustering buffers and develop a more advanced and computationally efficient method to select the following action.

Contributors

Shihmin WANG, Binqi ZHAO, Zhengfeng ZHANG, and Junping ZHANG designed the research. Shihmin WANG and

Binqi ZHAO processed the data. Shihmin WANG drafted the paper. Junping ZHANG and Jian PU helped organize the paper. Shihmin WANG, Junping ZHANG, and Jian PU revised and finalized the paper.

Compliance with ethics guidelines

Junping ZHANG and Jian PU are an editorial board member and a corresponding expert of *Frontiers of Information Technology & Electronic Engineering*, respectively, and they were not involved with the peer review process of this paper. All authors declare that they have no conflict of interest.

Data availability

The code is available at https://github.com/Shihwin/SelectiveSampling. The other data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Andrychowicz M, Wolski F, Ray A, et al., 2017. Hindsight experience replay. Proc 31st Int Conf on Neural Information Processing Systems, p.5055-5065. https://doi.org/10.5555/3295222.3295258

Bellemare MG, Srinivasan S, Ostrovski G, et al., 2016. Unifying count-based exploration and intrinsic motivation. Proc 30th Int Conf on Neural Information Processing Systems, p.1479-1487. https://doi.org/10.5555/3157096.3157262

Brockman G, Cheung V, Pettersson L, et al., 2016. OpenAI Gym. https://arxiv.org/abs/1606.01540

Cheung WC, Simchi-Levi D, Zhu RH, 2020. Reinforcement learning for non-stationary Markov decision processes: the blessing of (more) optimism. Proc 37th Int Conf on Machine Learning, Article 172. https://doi.org/10.5555/3524938.3525110

Dai XY, Zhao C, Wang X, et al., 2022. Image-based traffic signal control via world models. *Front Inform Technol Electron Eng*, 23(12):1795-1813. https://doi.org/10.1631/FITEE.2200323

Fu J, Luo K, Levine S, 2017. Learning robust rewards with adversarial inverse reinforcement learning. https://arxiv.org/abs/1710.11248

Goodfellow I, Pouget-Abadie J, Mirza M, et al., 2020. Generative adversarial networks. *Commun ACM*, 63(11):139-144. https://doi.org/10.1145/3422622

Haarnoja T, Zhou A, Abbeel P, et al., 2018. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. Proc 35th Int Conf on Machine Learning, p.1861-1870.

Hester T, Vecerik M, Pietquin O, et al., 2018. Deep Q-learning from demonstrations. Proc AAAI Conf on Artificial Intelligence, p.394. https://doi.org/10.1609/aaai.v32i1.11757

Ho J, Ermon S, 2016. Generative adversarial imitation learning. Proc 30th Int Conf on Neural Information Processing Systems, p.4572-4580. https://doi.org/10.5555/3157382.3157608

Houthooft R, Chen X, Duan Y, et al., 2016. VIME: variational information maximizing exploration. Proc 30th Int Conf on Neural Information Processing Systems, p.1117-1125. https://doi.org/10.5555/3157096.3157221

Huang ZZ, Chen J, Zhang JP, et al., 2023. Learning representation for clustering via prototype scattering and positive sampling. *IEEE Trans Patt Anal Mach Intell*, 45(6):7509-7524. https://doi.org/10.1109/TPAMI.2022.3216454

Kingma DP, Welling M, 2014. Auto-encoding variational Bayes. Proc 2nd Int Conf on Learning Representations.

Li HQ, Huang J, Cao Z, et al., 2023. Stochastic pedestrian avoidance for autonomous vehicles using hybrid reinforcement learning. *Front Inform Technol Electron Eng*, 24(1):131-140. https://doi.org/10.1631/FITEE.2200128

Liu SP, Tian GH, Cui YC, et al., 2022. A deep Q-learning network based active object detection model with a novel training algorithm for service robots. *Front Inform Technol Electron Eng*, 23(11):1673-1683. https://doi.org/10.1631/FITEE.2200109

Moerland TM, Broekens J, Plaat A, et al., 2023. Model-based reinforcement learning: a survey. *Found Trends Mach Learn*, 16(1):1-118. https://doi.org/10.1561/2200000086

Murtagh F, Legendre P, 2014. Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion. *J Classif*, 31(3):274-295. https://doi.org/10.1007/s00357-014-9161-z

Nair A, McGrew B, Andrychowicz M, et al., 2018. Overcoming exploration in reinforcement learning with demonstrations. Proc IEEE Int Conf on Robotics and Automation, p.6292-6299. https://doi.org/10.1109/ICRA.2018.8463162

Niu C, Zhang J, Wang G, et al., 2020. GATCluster: self-supervised Gaussian-attention network for image clustering. Proc 16th European Conf on Computer Vision, p.735-751. https://doi.org/10.1007/978-3-030-58595-2_44

Niu C, Shan HM, Wang G, 2022. SPICE: semantic pseudo-labeling for image clustering. *IEEE Trans Image Process*, 31:7264-7278. https://doi.org/10.1109/TIP.2022.3221290

Ravichandar H, Polydoros AS, Chernova S, et al., 2020. Recent advances in robot learning from demonstration. *Ann Rev Contr Robot Auton Syst*, 3:297-330. https://doi.org/10.1146/annurev-control-100819-063206

Rong GD, Shin BH, Tabatabaee H, et al., 2020. LGSVL Simulator: a high fidelity simulator for autonomous driving. Proc 23rd Int Conf on Intelligent Transportation Systems, p.1-6. https://doi.org/10.1109/ITSC45102.2020.9294422

Schaul T, Quan J, Antonoglou I, et al., 2016. Prioritized experience replay. Proc 4th Int Conf on Learning Representations.

Schulman J, Levine S, Moritz P, et al., 2015. Trust region policy optimization. Proc 32nd Int Conf on Machine Learning, p.1889-1897. https://doi.org/10.5555/3045118.3045319

Schulman J, Moritz P, Levine S, et al., 2016. High-dimensional continuous control using generalized advantage estimation. Proc 4th Int Conf on Learning Representations.

Schulman J, Wolski F, Dhariwal P, et al., 2017. Proximal policy optimization algorithms. https://arxiv.org/abs/1707.06347

Silver D, Huang A, Maddison CJ, et al., 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484-489. https://doi.org/10.1038/nature16961

Sun PQ, Zhou WG, Li HQ, 2020. Attentive experience replay. Proc AAAI Conf on Artificial Intelligence, p.5900-5907. https://doi.org/10.1609/aaai.v34i04.6049

Sutton RS, Barto AG, 1998. Reinforcement Learning: an Introduction. MIT Press, Cambridge, UK.

Vecerik M, Hester T, Scholz J, et al., 2017. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. https://arxiv.org/abs/1707.08817

Vinyals O, Babuschkin I, Czarnecki WM, et al., 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350-354. https://doi.org/10.1038/s41586-019-1724-z

Wang SC, Li B, 2020. Implicit posterior sampling reinforcement learning for continuous control. Proc 27th Int Conf on Neural Information Processing, p.452-460. https://doi.org/10.1007/978-3-030-63833-7_38

Xie JY, Girshick R, Farhadi A, 2016. Unsupervised deep embedding for clustering analysis. Proc 33rd Int Conf on Machine Learning, p.478-487. https://doi.org/10.5555/3045390.3045442

Xue JR, Hu B, Li LX, et al., 2022. Human-machine augmented intelligence: research and applications. *Front Inform Technol Electron Eng*, 23(8):1139-1141. https://doi.org/10.1631/FITEE.2250000

Ye DH, Chen GB, Zhang W, et al., 2020. Towards playing full MOBA games with deep reinforcement learning. Proc 34th Int Conf on Neural Information Processing Systems, Article 53.

Zhang JP, Pu J, Xue JR, et al., 2023a. HiVeGPT: human-machine-augmented intelligent vehicles with generative pre-trained transformer. *IEEE Trans Intell Veh*, 8(3):2027-2033. https://doi.org/10.1109/TIV.2023.3256982

Zhang JP, Pu J, Chen J, et al., 2023b. DSiV: data science for intelligent vehicles. *IEEE Trans Intell Veh*, 8(4):2628-2634. https://doi.org/10.1109/TIV.2023.3264601

Zhou J, Ke P, Qiu XP, et al., 2023. ChatGPT: potential, prospects, and limitations. *Front Inform Technol Electron Eng*, early access. https://doi.org/10.1631/FITEE.2300089