ECAI 2023
K. Gal et al. (Eds.)
© 2023 The Authors.
This article is published online with Open Access by IOS Press and distributed under the terms
of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA230343

OER: Offline Experience Replay for Continual Offline Reinforcement Learning

Sibo Gai[ab], **Donglin Wang**[b;*] **and Li He**[b]

[a]Fudan University, Shanghai, China
[b]Westlake University, Zhejiang, China

Abstract. The capability of continuously learning new skills via a sequence of pre-collected offline datasets is desired for an agent. However, consecutively learning a sequence of offline tasks likely leads to the catastrophic forgetting issue under resource-limited scenarios. In this paper, we formulate a new setting, continual offline reinforcement learning (CORL), where an agent learns a sequence of offline reinforcement learning tasks and pursues good performance on all learned tasks with a small replay buffer without exploring any of the environments of all the sequential tasks. For consistently learning on all sequential tasks, an agent requires acquiring new knowledge and meanwhile preserving old knowledge in an offline manner. To this end, we introduced continual learning algorithms and experimentally found experience replay (ER) to be the most suitable algorithm for the CORL problem. However, we observe that introducing ER into CORL encounters a new distribution shift problem: the mismatch between the experiences in the replay buffer and trajectories from the learned policy. To address such an issue, we propose a new model-based experience selection (MBES) scheme to build the replay buffer, where a transition model is learned to approximate the state distribution. This model is used to bridge the distribution bias between the replay buffer and the learned model by filtering the data from offline data that most closely resembles the learned model for storage. Moreover, in order to enhance the ability on learning new tasks, we retrofit the experience replay method with a new dual behavior cloning (DBC) architecture to avoid the disturbance of behavior-cloning loss on the Q-learning process. In general, we call our algorithm offline experience replay (OER). Extensive experiments demonstrate that our OER method outperforms SOTA baselines in widely-used Mujoco environments.

1 Introduction

Similar to human beings, a general-purpose intelligence agent is expected to learn new tasks continually. Such sequential tasks can be either online tasks learned through exploration or offline tasks learned through offline datasets, where the latter is equally important but has not drawn sufficient attention so far. Learning sequential tasks in offline setting can greatly improve the learning efficiency and avoid the dangerous exploration process in practice. Moreover, online learning sequential tasks is not always feasible due to the temporal and spatial constraints of the environment and the agent's on-board consideration itself. Therefore, studying offline reinforcement learning in the continual setting is quite important and valuable for general-purpose intelligence. If having sufficient computational resources, it is easy to accomplish such a goal. However, for an agent with limited resources [1], continual learning methods are indispensable to deal with such offline datasets. Consequently, we propose a new setting named continual offline reinforcement learning (CORL) in this paper, which integrates offline RL and continual learning.

Offline RL learns from pre-collected datasets instead of directly interacting with the environment [11]. By leveraging pre-collected datasets, offline RL methods avoid costly interactions and thus enhance learning efficiency and safety. However, offline RL methods suffer from the over-estimation problem of the out-of-distribution (OOD) data, where the unseen data are erroneously estimated to be high values. This phenomenon stems from the distribution shift between the behavior policy and the learning policy. Various methods have been proposed to address the over-estimation problem [11, 21]. In this paper, we focus on dealing with a sequence of offline datasets, which requires continual learning techniques.

The major challenge of continual learning is how to alleviate the catastrophic forgetting [31] issue on previous tasks when learning new tasks. There are three types of continual learning methods, including regularization-based methods [19, 53], modular methods [8, 30], and rehearsal-based methods [27, 4]. Experience replay (ER) is a widely-used rehearsal-based method [42], which alternates between learning a new task and replaying samples of previous tasks. In this paper, we consider using ER as a base for our own problem. We will show that ER is the most suitable algorithm for the CORL problem in the experiment section following.

However, since ER is designed for online continual RL, directly applying ER in our CORL yields poor performance due to two kinds of distribution shifts. The first is the distribution shift between the behavior policy and the learning policy, and the second is the distribution shift between the selected replay buffer and the corresponding learned policy. Existing methods focus on addressing the first shift issue, and no related work considers the second, which only appears in our CORL setting. Thus, simply integrating ER with Offline RL fails to alleviate the catastrophic forgetting. To solve the new problem above, we propose a novel model-based experience selection (MBES) method to fill the replay buffer. The key idea is to take advantage of the dynamic model to search for and add the most valuable episodes in the offline dataset into the replay buffer.

After having a good replay buffer for previous tasks, the learned policy corresponding to a new task needs to clone previous tasks.

* Corresponding Author. Email: wangdonglin@westlake.edu.cn

[1] Running on a server and communicating with the agent are undesirable due to complex-system, real-time, and data privacy issues.

Please contact the corresponding author for any appendices or supplementary material mentioned in the paper.

Behavior cloning (BC) as an online ER method is widely used [54], which is incompatible with the actor-critic architecture in offline RL. Even though we can carefully tune the weight of BC loss, tuning the hyper-parameter is a cumbersome task in general and is difficult in the offline setting. Therefore, integrating online ER with offline RL often derives a non-convergent policy. The reason is that the actor-critic model is hard to train [39], and the rehearsal term in the loss function has a negative effect on the learning process. To effectively replay experiences, we propose a dual behavior cloning (DBC) architecture instead to resolve the optimization conflict, where one policy optimizes the performance of the new task by using actor-critic architecture, and the second optimizes from the continual perspective for both new and learned tasks.

In summary, this paper considers investigating a new setting CORL. A novel MBES is proposed to select valuable experiences and overcome the mismatch between the experiences in the replay buffer and trajectories from the learned policy. Then, a DBC architecture is proposed to deal with the optimization conflict problem. By taking MBES and DBC as two key ideas for CORL setting, we name our overall scheme as offline experience replay (OER). The main contributions of this paper can be summarized as follows:

- We present a new setting CORL and then propose a novel scheme OER for CORL setting.
- We propose a novel selection method MBES for offline replay buffer by utilizing a dynamic model to reduce the distribution shift between experiences from replay buffer and learned policy.
- On the other hand, we propose a novel DBC architecture to prevent the learning process from collapsing by separating the Q-learning on current task and BC processes on all previous tasks.
- We experimentally verify the performance of different modules and evaluate our method on continuous control tasks. Our method OER outperforms all SOTA baselines for all cases.

2 Related Works

Offline Reinforcement Learning Offline RL learns from a collected offline dataset and suffers from the issue of out-of-distribution (OOD). Some prior works propose to constrain the learned policy towards the behavior policy by adding KL-divergence[38, 36, 45, 55], MSE [6], or the regularization of the action selection [21]. Some articles suggest that if the collected data is sub-optimal, these approaches usually perform not well [29]. But other works point out that adding a supervised learning term to the policy improvement objective [10] will also receive high performance by reducing the exploration. Another effective way is to learn a conservative Q-function [22, 28, 20], which assigns a low Q-value for OOD states and then extracts the corresponding greedy policy. Moreover, other works propose to use an ensemble model to estimate Q-value [1] or consider the importance sampling [38]. Such methods have not previously considered the setting of sequential tasks and naive translating them into CORL setting is ineffective, while this paper focuses on sequential tasks and aims to solve the catastrophic forgetting problem during the process of learning a sequence of offline RL tasks.

On the other hand, recent works [5, 24] propose to train a dynamic model to predict the values of OOD samples in a supervised-learning way. Such model-based offline RL methods offer great potential for solving the OOD problem, even though the transition model is hardly accurate strictly. The model algorithm is thought to alleviate the OOD problem faced by offline RL and thus improve the robustness of the offline agent. Model-based offline RL methods have two major categories: one focuses on measuring the uncertainty of the learned dynamic model [52, 17], and the other considers the pessimistic estimation [51]. Different from most of these works using the dynamic model to generate OOD samples when training the agent, in this paper, we utilize the dynamic model to search for the most valuable episodes in the offline dataset for the ER method.

Continual Reinforcement Learning Offline methods may consider a single-task or multi-task scenario [50, 25, 26]. In contrast, continual learning attempts to learn new tasks after having learned old tasks and get the best possible results on all tasks. Generally, continual learning methods can be classified into three categories [37]: regularization-based approaches [19, 53] add a regularization term to prevent the parameters from far from the value learned from past tasks; modular approaches [8, 30] consider fixed partial parameters for a dedicated task; and rehearsal-based methods [27, 4] train an agent by merging the data of previously learned tasks with that of the current task. All three kinds of continual learning methods have been applied for RL tasks [15, 48, 32, 23]. Specifically, our work is based on the rehearsal method in an RL setting [42, 14]. Therefore, we will detail the works involved in this category following.

There are two essential questions to answer in rehearsal-based continual learning. The first is how to choose samples from the whole dataset to store in the replay buffer with a limited size [49]. The most representative samples [41, 43] or samples easy to forget [3] are usually selected in the replay buffer while random selection has also been used in some works [40, 2]. However, these algorithms are designed for image classification and are not applicable to RL. [14] focuses on replay buffer question sampling in online RL setting. The second is how to take advantage of the saved replay samples [49, 3]. In RL, the two most commonly used approaches are BC and perfect memory [46] in continual RL, where BC is more effective in relieving catastrophic forgetting. At present, all these methods are designed for online RL setting. Unlike prior works, we consider the offline RL setting in this paper, where catastrophic forgetting and overestimation must be overcome simultaneously.

To the best of our knowledge, this is *the first work* to solve offline RL problems in the continual learning setting.

3 Problem Formulation and Preliminary

Continual Offline Reinforcement Learning In this paper, we investigate CORL, which learns a sequence of RL tasks $\mathcal{T} = (T_1, \cdots, T_N)$. Each task T_n is described as a Markov Decision Process (MDP) represented by a tuple of $\{\mathcal{S}, \mathcal{A}, P_n, \rho_{0,n}, r_n, \gamma\}$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P_n : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability, $\rho_{0,n} : \mathcal{S}$ is the distribution of the initial state, $r_n : \mathcal{S} \times \mathcal{A} \to [-R_{\max}, R_{\max}]$ is the reward function, and $\gamma \in [0, 1)$ is the discounting factor. We assume that sequential tasks have different $P_n, \rho_{0,n}$ and r_n, but share the same $\mathcal{S}$, $\mathcal{A}$, and γ for simplicity. The return is defined as the sum of discounted future reward $R_{t,n} = \sum_{i=t}^{H} \gamma^{(i-t)} r_n(s_i, a_i)$, where H is the horizon.

We define a parametric Q-function $Q(s, a)$ and a parametric policy $\pi(a|s)$. Q-learning methods train a Q-function by iteratively applying the Bellman operator $\mathcal{B}^* Q(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(s'|s,a)} (\max_{a'} Q(s', a'))$. We also train a transition model for each task $\hat{P}_n(s'|s, a)$ by using maximum likelihood estimation $\min_{\hat{P}_n} \mathbb{E}_{(s,a,s') \sim \mathcal{D}} \left[\log \hat{P}(s'|s, a)\right]$. We use a multi-head architecture for the policy network π to avoid the same-state-different-task problem [16]. In detail, the policy network consists of a feature extractor θ_z for all tasks and multiple heads $\theta_n, n \in [1, N]$, where one

head is for each task. π_n is defined to represent the network with joint parameters $[\theta_z, \theta_n]$ and h_n is defined to represent the head with parameters θ_n. Our aim is to train sequential tasks all over $[T_1, \cdots, T_{N-1}]$ sequentially and get a high mean performance and a low forgetting of all the learned tasks without access to data from previous tasks except a small buffer.

In online RL setting, the experiences $e = (s, a, s', r)$ can be obtained through environment interaction. However, in offline RL setting, the policy $\pi_n(a|s)$ can only be learned from a static dataset $\mathcal{D}_n = \{e_n^i\}, e_n^i = (s_n^i, a_n^i, s_n'^i, r_n^i)$, which is assumed to be collected by an unknown behavior policy $\pi_n^\beta(a|s)$.

Experience Replay ER [42] is the most widely-used rehearsal-based continual learning method. In terms of task T_n, the objective of ER is to retain good performance on previous tasks $[T_1, \cdots, T_{n-1}]$, by using the corresponding replay buffers $[B_1, \cdots, B_{n-1}]$, which called perform memory. Moreover, two additional behavior cloning losses, including the actor cloning loss and the critic cloning loss, are commonly used for previous tasks as follows

$$L_{\text{actor\_cloning}} := \sum_{s,a \in B} \|\pi_n(s) - a\|_2^2, \quad (1)$$

$$L_{\text{critic\_cloning}} := \sum_{s,a,Q_{\text{replay}} \in B} \left\|Q_n(a,s) - Q_{\text{replay}}\right\|_2^2, \quad (2)$$

where B is the replay buffer and Q_{replay} means the Q value saved from previous tasks. These two losses are called BC [46].

Subsequent work [46] shows that for a soft actor-critic [13] architecture, the replay loss added on the actor network (Eq. 1) performs well, but the loss added on the critic network (Eq. 2) poorly effect. Therefore, we only consider the actor cloning loss (Eq. 1) in our work.

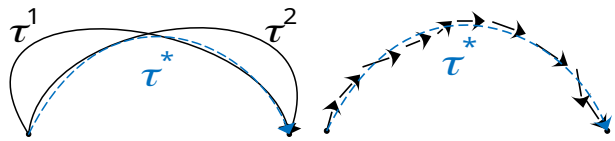
However, naively integrating the ER with offline RL results in a significant performance drop for CORL problems. To tackle this problem, we propose a novel method OER, which consists of two essential components as follows.

4 Offline Experience Replay (OER)

In this section, we first elaborate on how to select valuable experiences and build the replay buffer. Then, we describe a novel DBC architecture as our replay model. Finally, we summarize our algorithm and provide the Pseudocode.

4.1 Offline Experience Selection Scheme

Novel Distribution Shift Problem: Facing with sequential tasks, an agent learns the task T_n in order after learning T_{n-1}, and should prepare for the next learning. In terms of task T_n, how to select partial valuable data from offline dataset $\mathcal{D}_n$ to build a size-limited replay buffer B_n is critical. In online RL setting, rehearsal-based methods commonly utilize a ranking function $\mathcal{R}(s_i, a_i) = \left|r_i + \gamma \max_{a'} Q\left(s_i', a'\right) - Q(s_i, a_i)\right|$ to evaluate the value of trajectories. Here, $\mathcal{R}(s_i, a_i)$ with $(s_i, a_i) \in \mathcal{D}$ evaluates replay trajectories in terms of Q-value accuracy or total accumulative reward [14], where the trajectories are collected by the optimal policy π_n^* interacting with the environment. On the contrary, in offline RL setting, if we consider the similar method above, the data selection can only be made in the offline dataset $\mathcal{D}_n$ corresponding to the behavior policy π_n^β. Therefore, there exists a distribution shift between π_n^β and π_n^*, which inevitably affects the performance of the rehearsal. In our selection scheme, we attempt to identify and filter out those offline trajectories in $\mathcal{D}_n$ that are not likely to be generated by π_n^*. To clarify this point, we give an illustration in Fig.1, where those trajectories close to the offline optimal trajectory are selected and stored in the replay buffer B_n. The distribution of these trajectories in terms of both $\mathcal{S}$ and $\mathcal{A}$ differs from the offline dataset $\mathcal{D}_n$.



(a) offline learned trajectory (b) Offline buffer selection

Figure 1: Distribution shift between the experiences from replay buffer and trajectories from learned policy. (a) τ^1 and τ^2 represent two trajectories in offline dataset, and τ^* represents the trajectory generated by the optimal policy. The learned policy can generate better trajectories than original dataset. (b) The arrow represents the experiences in offline dataset near the optimal trajectory. Episodes close to the optimal trajectory are more valuably selected.

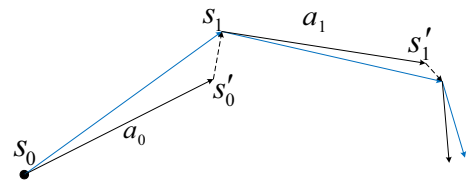


Figure 2: Illustration of our MBES method to fill in B_n for T_n. Blue lines represent the experiences in D_n, and black lines represent the generated experiences by optimal policy π_n^*. Starting from s_0 and $a_0 = \pi_n^*(s_0)$, the next state is $s_0' = \hat{P}(s_0, a_0)$. To avoid the accumulated error, we use the most similar state s_1 in the offline dataset $\mathcal{D}_n$ instead of the s_0' as the next state.

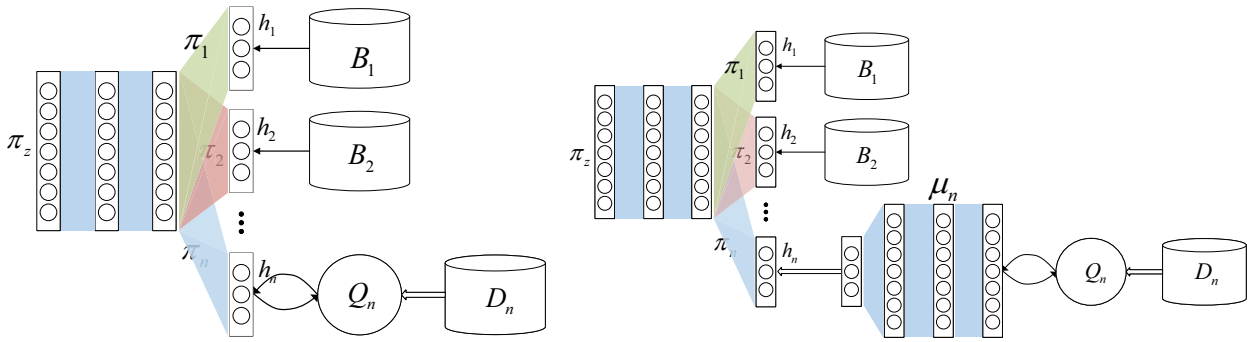
Model-based Experience Selection (MBES): In order to address the new distribution shift question above, we propose a novel MBES scheme. As shown in Fig.2, based on the offline trajectories in $\mathcal{D}_n$ collected from π_n^β, MBES aims to generate a new trajectory corresponding to π_n^*. Specifically, MBES considers both the learned optimal policy and a task-specific dynamic model $\hat{P}_n$, where the dynamic model $\hat{P}_n$ is obtained via supervised learning by using $\mathcal{D}_n$. Starting from the tth state s_t, we recursively sample an action by π_n^* and predict the next state $s_t' = \hat{P}_n\left(s_t, \pi_n^*(s_t)\right)$.

However, recursively sampling actions on predicted states causes a significant accumulation of compounding error after several steps, which hinders the selected trajectories from being revisited later. In order to remove the compounding error, after learning task n, starting from the tth state s_t, instead of directly taking the model output as the next state, a state in $\mathcal{D}_n$ most similar to the model prediction s_t' is selected as the next state s_{t+1} for the pseudo exploitation. Here, we use the L_2 metric to measure the similarity as follows

$$s_{t+1} = \underset{s \in \mathcal{D}_n}{\operatorname{argmax}} \operatorname{dist}\left(s, s_t'\right)_{s_t' \sim \hat{P}_n(s_t, \pi_n^*(s_t))}, \quad (3)$$

where dist means a distance between s and s_t'. [2] According to the

[2] Here we choose the L2 distance of the feature from the last hidden layer of the Q network in our experiments. However, we find that a simple L2 distance in the state space $\mathcal{S}$ also works well.



(a) offline learned T^* (b) Offline buffer selection

Figure 3: Network architecture of BC and our DBC for experience play, where the policy $\pi_n = [\pi_z, h_n]$ is for task n, $n = 1, \cdots, N$. (a) Existing multi-head architecture. Here, the newly-added head h_n for T_n is learned via an actor-critic algorithm with a Q-network Q_n, but other heads from h_1 to h_{n-1} corresponding to previous tasks are learned by cloning B_1 to B_{n-1}. (b) Our DBC architecture. We use an independent policy network μ_n to learn the current task T_n and a newly-added head h_n is used to clone μ_n.

model-based offline RL methods [5, 24], we further introduce the variance of $\hat{P}_n\left(s_t, \pi_n^*\left(s_t\right)\right)$ to determine whether the results of the dynamic model are reliable or not, where we use Eq.3 for selection only if the variance of the $\hat{P}_n$ is lower than the threshold; otherwise, keeping $\pi_n^\beta\left(s_t\right)$ instead. In our experiments, we specifically use $2\hat{P}_n\left(s_t, a_t\right)$ as the threshold, which has minimal impact.

For a start-up, we sample s_0 from $\rho_{0,n}$, and then iteratively carry out. In the end, we save the generated trajectory in B_n.

4.2 Dual Behavior Cloning (DBC)

Prior rehearsal-based approaches utilize a multi-head policy network π and a Q-network Q_n to learn task T_n, as shown in Fig.3 (a). During learning, the policy network π is to clone the experience data stored in replay buffers B_1 to B_{n-1} for all previous tasks T_1 to T_{n-1}. However, such an architecture suffers from an obvious performance drop when the number of tasks increases, which indicates that the policy π trained via BC has the incapability of mastering both previous knowledge from old tasks and new knowledge from the new task.

This performance drop is due to the existing inconsistency between the following two objectives: on the one hand, the multi-head policy π is optimized for all current and previous tasks T_1 to T_n for action prediction; on the other hand, the policy π is also used for updating the current Q-network Q_n. More specifically, we begin this analysis from Q-learning [34, 35] with the Bellman equation of $Q\left(s, a\right) = r\left(s, a\right) + \gamma * \max_{a'} Q\left(s', a'\right)$. In a continuous action space, as the maximization operator above cannot be realized, a parameterized actor network $\mu\left(s\right)$ is commonly used instead to take the optimal action corresponding to maximal Q value. μ and π can be considered equivalently in a single-task setting. However, for continual learning, π is constrained to clone all previous tasks from T_1 to T_{n-1} while μ depends only on the current task T_n. Consequently, it is difficult for the policy π to take action corresponding to the maximum future reward for s' in task T_n.

Based on such analysis, we propose a novel DBC scheme to solve the inconsistency mentioned above, and the schematic diagram of our proposed architecture is given in Fig. 3(b). In comparison to existing multi-head network architecture in Fig. 3(a) [42], we propose an extra policy network μ_n in order to learn the optimal state-action mapping for T_n. Specifically, when learning task T_n, we first obtain μ_n and Q_n by using an offline RL algorithm. Then, in the rehearsal phase, the continual learning policy π is required to clone the previous experiences from B_1 to B_{n-1} and meanwhile be close to μ_n. Thus, the corresponding loss L_π can be written as follows

$$\begin{aligned} L_\pi &= \mathbb{E}_{(s,a,s')\sim\mathcal{D}_n}\left(\pi_n(s,a) - \mu_n\left(s,a\right)\right)^2 \\ &+ \lambda_r \frac{1}{n}\sum_{j=1}^{n-1}\mathbb{E}_{(s,a)\sim\mathcal{B}_j}\left(\pi_j\left(s\right) - a\right)^2, \end{aligned} \tag{4}$$

where λ_r is a coefficient for the BC item. It is worth mentioning that the continual learning policy π attempts to clone the behavior of both μ_n and the previous experiences from B_1 to B_{n-1} simultaneously. Therefore, we name our scheme DBC.

4.3 Algorithm Summary

When considering a new task T_n, we first utilize DBC to learn the two policy networks π, μ_n, and the dynamic model $\hat{P}_n$ until convergence. Then, the learned policy π and the dynamic model $\hat{P}_n$ are used in MBES to select valuable data for B_n. To summarize, the overall process of OER, including DBC and MBES, is given in Algorithm 1.

Algorithm 1 Our proposed Method OER

Require: Number of tasks N; initiate the policy π.
1: **for** Tasks T_n in $[1, \cdots, N]$ **do**
2: Get offline dataset $\mathcal{D}_n$; Initiate the replay buffer $B_n = \emptyset$; Initiate new head h_n for π; Initiate μ_n, Q_n and $\hat{P}_n$.
3: **while** Not Convergence **do**
4: Update μ_n and Q_n via offline learning method.
5: Update π via Eqn. 4 to clone μ_n and B_0 to B_{n-1}.
6: Update the dynamic model $\hat{P}_n$.
7: **end while**
8: Sample initial state s_0 from $\rho_{0,n}$, and $s_t \leftarrow s_0$.
9: **while** B_n is not full **do**
10: **if** s_t is not terminal state **then**
11: Select s_{t+1} from $\mathcal{D}_n$ by π_n and $\hat{P}_n$ via Eqn. 3.
12: **else**
13: Sample s_{t+1} from $\rho_{0,n}$.
14: **end if**
15: Add s_{t+1} into B_n, and $s_t \leftarrow s_{t+1}$.
16: **end while**
17: **end for**
Ensure: π.

5 Implementation Details

We model the Q-function and policy network as a multi-layer perceptron (MLP) with two hidden layers of 128 neurons, each with ReLU non-linearity based on [44]. We use an ensemble dynamic model containing five individual models, which is also modeled as an MLP, the same as the Q-function and the policy network. Any offline RL method is compatible with our architecture. We choose TD3+BC [10] as the backbone algorithm because of its simple structure to demonstrate that our algorithm does not depend on a specific offline algorithm. Further, we use Adam [18] with a learning rate of 0.001 to update both the Q-function and the dynamic model and 0.003 to update both the policy network π and μ_n. Then, for each task, we train 30, 000 steps and switch to the next. We find that initializing μ_n with π_{n-1} and learning π and μ_n simultaneously work well from the experience. Also, learning π and μ_n together will reduce the scope of the gradient and avoid the jumping change to ensure stable learning and relieve catastrophic forgetting. The result is calculated via five repeated simulations with different numbers of seeds.

6 Experiments

Extensive experiments are conducted to demonstrate the effectiveness of our proposed scheme and test whether we can keep both stability and plasticity at the same time when learning sequential offline RL tasks. We evaluate the performance of MBES and DBC separately to test the performance of each approach.

6.1 Baselines and Datasets

Baselines *On one hand*, in order to evaluate the MBES, we consider six replay buffer selection approaches, where four from [14] are given as follows.

- Surprise [14]: store trajectories with maximum mean TD error: $\min_{\tau\in D_i} \mathbb{E}_{s,a,s_{t+1}\in\tau} \|\mathcal{B}^*Q(s,a) - Q(s,a)\|_2^2$.
- Reward [14]: store trajectories with maximum mean reward: $\max_{\tau\in D_i} R_{t,n}$.
- Random [14]: randomly select samples in dataset.
- Coverage [14]: store those samples to maximize coverage of the state space, or ensure the sparsity of selected states: $\min_{s\in D_i} |\mathcal{N}_i|\,; \mathcal{N}_i = \{s' \text{ s. t. dist}(s'-s) < d\}$.

Considering that these baselines are designed for online RL and it is not fair enough to use them only as the baselines, we have designed two algorithms for comparison that are applicable to offline RL, based on the idea of [14].

- Match: select samples in the offline dataset most consistent with the learned policy. Trajectories chosen in this way are most similar to the learned policy in the action space, but may not match in the state space: $\min_{\tau\in D_i} \mathbb{E}_{s,a\in\tau} \left\|a - \pi_i^*(s)\right\|_2^2$.
- Model: Given that we used a Model-Based approach to filter our data, we also used it as a criterion for whether the trajectories matched the transfer probabilities: $\min_{\tau\in D_i} \mathbb{E}_{s,a\in\tau} \left\|\hat{P}_i(s,a) - P_i(s,a)\right\|_2^2$. This metric is used to demonstrate that the introduction of the Model-Based approach alone does not improve the performance of the CORL algorithm.

On the other hand, in order to evaluate the DBC, we consider five widely-used continual learning methods, where three methods need to use replay buffers.

- BC [42]: a basic rehearsal-based continual method adding a behavior clone term in the loss function of the policy network.
- Gradient episodic memory (GEM) [27]: a method using an episodic memory of parameter gradients to limit the policy update.
- Averaged gradient episodic memory (AGEM) [4]: a method based on GEM that only uses a batch of gradients to limit the policy update.

In addition, the following two regularization-based methods are rehearsal-free so that they are independent of experience selection methods.

- Elastic weight consolidation (EWC) [19]: constrain the changes to critical parameters through the Fisher information matrix.
- Synaptic intelligence (SI) [53]: constrain the changes after each step of optimization.

We also show the performance on multi-task as a reference. The multi-task learning setting does not suffer from the catastrophic forgetting problem and can be seen as superior.

Offline Sequential Datasets We consider three sets of tasks from widely-used continuous control offline meta-RL library[3] as in [33]:

- Ant-2D Direction (Ant-Dir): train a simulated ant with 8 articulated joints to run in a 2D direction;
- Walker-2D Params (Walker-Par): train a simulated agent to move forward, where different tasks have different parameters. Specifically, different tasks require the agent to move at different speeds;
- Half-Cheetah Velocity (Cheetah-Vel): train a cheetah to run at a random velocity.

For each set of tasks, we randomly sample five tasks to form sequential tasks T_1 to T_5.

To consider different data quality as [9], we train a soft actor-critic to collect two benchmarks [12] for each task T_n, $n = 1, \cdots, 5$: 1) Medium (M) with trajectories from medium-quality policy, and 2) Medium-Random (M-R) including trajectories from both medium-quality policy and trajectories randomly sampled.

Metrics Following [7], we adopt the average performance (PER) and the backward transfer (BWT) as evaluation metrics,

$$\text{PER} = \frac{1}{N}\sum_{n=1}^{N} a_{N,n}, \ \text{BWT} = \frac{1}{N-1}\sum_{n=1}^{N-1} a_{n,n} - a_{N,n}, \qquad (5)$$

where $a_{i,j}$ means the final cumulative rewards of task j after learning task i. For PER, higher is better; for BWT, lower is better. These two metrics show the performance of learning new tasks while alleviating the catastrophic forgetting problem.

6.2 Overall Results

Evaluate MBES: Firstly, our method OER is compared with eleven baselines on two kinds of qualities M-R and M. Since our OER comprises MBES and DBC, six experience selection approaches are added with DBC for a fair comparison. The overall results are reported in Table 1, in terms of PER and BWT on three sequential tasks. From Table 1, we draw the following conclusions: 1)

[3] Most current continual RL methods perform not well on complex benchmarks such as [47], so it is not suitable for evaluating our method [33] and we do not consider it in this paper.

Benchmark	Methods	Ant-Dir		Walker-Par		Cheetah-Vel	
		PER	BWT	PER	BWT	PER	BWT
M-R	MultiTask	1387.54	-	1630.90	-	-112.49	-
	Coverage+DBC	677.72	727.40	231.17	1428.14	-404.33	293.42
	Match+DBC	776.85	432.74	120.96	1079.19	-121.30	26.50
	Supervise+DBC	903.12	175.17	196.07	1063.35	-233.05	82.60
	Reward+DBC	893.63	36.10	554.05	1087.96	-242.50	102.75
	Model+DBC	1156.43	65.66	194.31	1214.32	-141.20	57.90
	Random+DBC	845.82	126.56	614.71	979.90	-104.16	36.24
	OER	**1316.46**	**119.71**	**1270.62**	**550.18**	**-76.61**	**16.54**
M	MultiTask	1357.20	-	1751.68	-	-115.30	-
	Coverage+DBC	842.46	599.74	361.55	1342.87	-424.87	229.89
	Match+DBC	841.88	549.98	886.28	678.55	-196.83	88.87
	Supervise+DBC	1049.84	347.88	1020.57	666.15	-219.78	56.99
	Reward+DBC	1125.44	269.76	891.55	790.07	-222.83	40.15
	Model+DBC	1147.05	253.49	872.94	807.33	-184.83	24.22
	Random+DBC	1189.17	179.75	1102.89	616.52	-150.39	30.18
	OER	**1215.58**	**176.85**	**1192.59**	**518.20**	**-148.18**	**16.36**

Table 1: Performance of our OER and baselines to verify the effectiveness of MBES, where M-R and M are included. We can observe that our method OER has the highest PER and lowest BWT in all cases.

Benchmark	Methods	Ant-Dir		Walker-Par		Cheetah-Vel	
		PER	BWT	PER	BWT	PER	BWT
M-R	MBES+SI	747.95	643.89	124.04	1460.08	-437.49	316.84
	MBES+EWC	655.21	726.37	110.53	1589.62	-568.07	506.19
	MBES+GEM	748.90	643.06	114.07	1477.87	-445.39	389.10
	MBES+AGEM	722.41	687.97	62.27	1628.53	-546.15	419.61
	MBES+BC	407.94	874.80	46.75	860.17	-645.79	21.92
	OER	**1316.46**	**119.71**	**1270.62**	**550.18**	**-76.61**	**16.54**

Table 2: Performance of our OER and baselines to verify the effectiveness of DBC, where M-R is included. The result of M can be found in the Supplementary Material. We can observe that our method OER has the highest PER and lowest BWT in all cases.

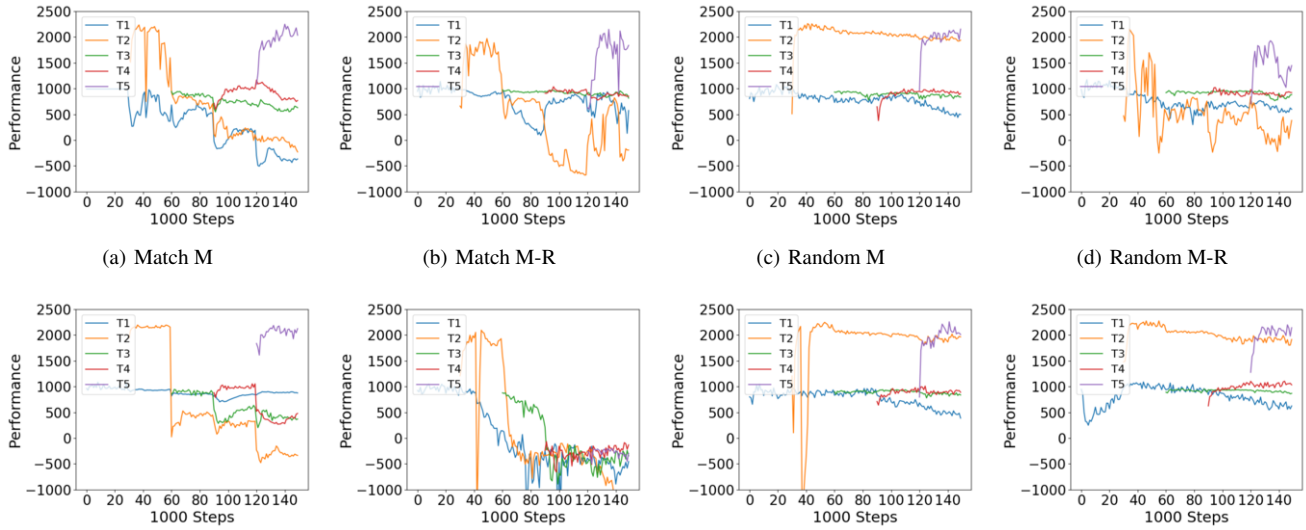
Our OER method outperforms all baselines for all cases, indicating the effectiveness of our MBES scheme; 2) Our OER method demonstrates greater superiority on M-R dataset than on M dataset, indicating that M-R dataset has a larger distribution shift than M dataset, and MBES addresses such shift; 3) Random+DBC performs better than the other five baselines because these five experience selection schemes are dedicated for online but not offline scenarios. Furthermore, Fig. 4 shows the learning process of Ant-Dir on five sequential tasks. From Fig.4(a) - 4(d), Fig.4(g) - 4(h) and Supplementary Material, we can observe that compared with baselines, our OER demonstrates less performance drop with the increment of tasks, indicating that OER can better solve the catastrophic forgetting.

Evaluate DBC: Secondly, our method OER is compared with five baselines. Similarly, continual learning approaches are added with MBES for a fair comparison, and the overall performance is reported in Table 2 and the Supplementary Material. From Table 2 and the Supplementary Material, we draw the following conclusions: 1) Our OER method outperforms all baselines, indicating the effectiveness of our DBC scheme; 2) Four continual learning methods perform not well due to the forgetting problem; 3) MBES+BC performs the worst due to the inconsistency of two policies π and μ_n in Section 4.2. From Fig.4(e) - 4(h) and Supplementary Material, we can observe that our OER can learn new tasks and remember old tasks well; other continual learning methods can only learn new tasks but forget old tasks, while BC cannot even learn new tasks.

6.3 *Parameter Analysis*

Size of Buffer B_n In rehearsal-based continual learning, the size of replay buffer is a key factor. In our CORL setting, the size of buffer B_n is selected as 1 K for all n, by considering both storage space and forgetting issues. With the increase of buffer size, we need more storage space but have less forgetting issue. In order to quantify such analysis, we consider a buffer size 10 K for OER and two baselines, and the results are listed in Table 3. From Table 3, we can observe that 1) With the increase of buffer size, OER and two baselines achieve better performance as expected. 2) Our DBC method is still much better than BC, indicating that solving the inconsistency is significant; 3) With larger storage space, the baseline Random performs similar as MBES, because in this case forgetting issue gets much smaller and the experience selection becomes not important.

Replay Coefficient λ_r Another key factor is the coefficient λ_r in Eq. 4, where λ_r is used to balance the anti-forgetting BC item and the new-policy constraint item. In our CORL setting, we select λ_r as 1, which is also the general choice in ER-based approaches [42, 14], and good performance has been achieved, as mentioned above. We analyze different values of λ_r and show the corresponding performance of our OER and baselines in Table 4, where λ_r is selected as 0.3, 1 and 3, respectively. From Table 4, we can observe that with a larger λ_r, the forgetting issue gradually reduces, but it gets looser that the learning policy π clones μ_n in Eq. 4, and vice versa. As a result, we achieve the best performance when $\lambda_r = 1$. This is why we use $\lambda_r = 1$ for all experiments in this paper.



(a) Match M (b) Match M-R (c) Random M (d) Random M-R

(e) EWC M (f) BC M (g) OER M (h) OER MR

Figure 4: Process of learning five sequential tasks, where our OER is compared with Match (M-R and M), Random (M-R and M), EWC (M) and BC (M). More results are given in Supplementary Material. Every 30000 steps on one task, we switch to the next task.

Method		Ant-Dir Medium		Walker-Par Medium		Cheetah-Vel Medium	
		PER	BWT	PER	BWT	PER	BWT
Reward	BC	-308.10	1049.32	124.54	1102.85	-706.31	343.23
	DBC	1310.45	78.89	1528.63	157.11	-207.76	39.82
Random	BC	-295.06	997.96	115.14	1199.63	-750.20	295.36
	DBC	**1374.54**	32.01	1565.50	**45.42**	-143.10	56.16
MBES	BC	-338.03	1164.01	588.26	958.38	-582.93	199.81
	DBC	1367.07	**6.32**	**1578.30**	78.72	**-141.17**	**21.18**

Table 3: Performance of baselines Reward, Random and our OER, where the replay buffer capacity is as 10,000 samples for BC and DBC.

7 Conclusion

In this work, we formulate a new CORL setting and present a new method OER, primarily including two key components: MBES and DBC. We point out a new distribution bias problem and training instability unique to the new CORL setting. Specifically, to solve a novel distribution shift problem in CORL, we propose a novel MBES scheme to select valuable experiences from the offline dataset to build the replay buffer. Moreover, in order to address the inconsistency issue between learning the new task and cloning old tasks, we propose a novel DBC scheme. Experiments and analysis show that OER outperforms SOTA baselines on various continuous control tasks.

Method	λ_r	Ant-Dir M		Walker-Par M	
		PER	BWT	PER	BWT
Supervise +DBC	0.3	980.25	537.92	1003.92	688.19
	1	1049.84	347.88	1020.57	666.15
	3	984.62	201.70	944.71	608.82
Random +DBC	0.3	1168.82	184.57	1052.13	647.71
	1	1189.17	179.75	1102.89	616.52
	3	952.46	168.81	1157.30	330.47
OER	0.3	940.38	431.13	1049.55	606.18
	1	1215.58	176.85	1192.59	518.20
	3	1128.66	170.32	1051.63	507.05

Table 4: Performance of our OER method with different coefficient λ_r. High λ_r indicates more on replaying previous tasks.

References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi, 'An optimistic perspective on offline reinforcement learning', in *ICML*, (2020).

[2] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara, 'Dark experience for general continual learning: A strong, simple baseline', in *NIPS*, (2020).

[3] Arslan Chaudhry, Albert Gordo, Puneet Kumar Dokania, Philip H. S. Torr, and David Lopez-Paz, 'Using hindsight to anchor past knowledge in continual learning', in *AAAI*, (2021).

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny, 'Efficient Lifelong Learning with A-GEM', in *ICLR*, (2019).

[5] Xiong-Hui Chen, Yang Yu, Qingyang Li, Fan Luo, Zhiwei Qin, Wenjie Shang, and Jieping Ye, 'Offline model-based adaptable policy learning', in *NeurIPS*, (2021).

[6] Robert Dadashi, Shideh Rezaeifar, Nino Vieillard, Léonard Hussenot, Olivier Pietquin, and Matthieu Geist, 'Offline Reinforcement Learning with Pseudometric Learning', in *ICML*, (July 2021).

[7] Mohammad Mahdi Derakhshani, Xiantong Zhen, Ling Shao, and Cees Snoek, 'Kernel Continual Learning', in *ICML*, volume 139, (July 2021).

[8] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra, 'PathNet: Evolution Channels Gradient Descent in Super Neural Networks', *CoRR*, **abs/1701.08734**, (2017). arXiv: 1701.08734.

[9] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine, 'D4RL: Datasets for Deep Data-Driven Reinforcement Learning', *CoRR*, **abs/2004.07219**, (2020). arXiv: 2004.07219.

[10] Scott Fujimoto and Shixiang (Shane) Gu, 'A Minimalist Approach to

Offline Reinforcement Learning', in *NIPS*. Curran Associates, Inc., (2021).
[11] Scott Fujimoto, David Meger, and Doina Precup, 'Off-policy deep reinforcement learning without exploration', in *ICML*, (2019).
[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine, 'Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor', in *ICML*, eds., Jennifer Dy and Andreas Krause, volume 80 of *Proceedings of Machine Learning Research*. PMLR, (July 2018).
[13] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, G. Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, P. Abbeel, and Sergey Levine, 'Soft actor-critic algorithms and applications', *ArXiv*, **abs/1812.05905**, (2018).
[14] David Isele and Akansel Cosgun, 'Selective Experience Replay for Lifelong Learning', *AAAI*, (April 2018).
[15] Christos Kaplanis, Murray Shanahan, and Claudia Clopath, 'Policy Consolidation for Continual Reinforcement Learning', in *ICML*, volume 97 of *Proceedings of Machine Learning Research*. PMLR, (2019).
[16] Samuel Kessler, Jack Parker-Holder, Philip J. Ball, Stefan Zohren, and Stephen J. Roberts, 'Same state, different task: Continual reinforcement learning without interference', in *AAAI*, (2021).
[17] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims, 'Morel : Model-based offline reinforcement learning', *ArXiv*, **abs/2005.05951**, (2020).
[18] Diederik P. Kingma and Jimmy Ba, 'Adam: A Method for Stochastic Optimization', in *ICLR*, (2015).
[19] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell, 'Overcoming catastrophic forgetting in neural networks', *Proceedings of the National Academy of Sciences*, **114**(13), (2017). _eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1611835114.
[20] Ilya Kostrikov, Ashvin Nair, and Sergey Levine, 'Offline reinforcement learning with implicit q-learning', in *ICLR*, (2021).
[21] Aviral Kumar, Justin Fu, G. Tucker, and Sergey Levine, 'Stabilizing off-policy q-learning via bootstrapping error reduction', in *NeurIPS*, (2019).
[22] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine, 'Conservative Q-Learning for Offline Reinforcement Learning', in *NIPS*, volume 33. Curran Associates, Inc., (2020).
[23] Erwan Lecarpentier, David Abel, Kavosh Asadi, Yuu Jinnai, Emmanuel Rachelson, and Michael L. Littman, 'Lipschitz Lifelong Reinforcement Learning', in *AAAI*, (2021).
[24] Byung-Jun Lee, Jongmin Lee, and Kee-Eung Kim, 'Representation Balancing Offline Model-based Reinforcement Learning', in *ICLR*, (February 2022).
[25] Jinxin Liu, Zhang Hongyin, and Donglin Wang, 'Dara: Dynamics-aware reward augmentation in offline reinforcement learning', in *International Conference on Learning Representations*, (2021).
[26] Jinxin Liu, Ziqi Zhang, Zhenyu Wei, Zifeng Zhuang, Yachen Kang, Sibo Gai, and Donglin Wang, 'Beyond ood state actions: Supported cross-domain offline reinforcement learning', *ArXiv*, **abs/2306.12755**, (2023).
[27] David Lopez-Paz and Marc' Aurelio Ranzato, 'Gradient Episodic Memory for Continual Learning', in *NIPS*, volume 30. Curran Associates, Inc., (2017).
[28] Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu, 'Mildly conservative q-learning for offline reinforcement learning', in *Advances in Neural Information Processing Systems*, eds., Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, (2022).
[29] Yecheng Ma, Dinesh Jayaraman, and Osbert Bastani, 'Conservative offline distributional reinforcement learning', *NIPS*, **34**, (2021).
[30] Arun Mallya and Svetlana Lazebnik, 'PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (June 2018).
[31] Michael McCloskey and Neal J. Cohen, 'Catastrophic interference in connectionist networks: The sequential learning problem', Psychology of Learning and Motivation, Academic Press, (1989).
[32] Jorge Mendez, Boyu Wang, and Eric Eaton, 'Lifelong Policy Gradient Learning of Factored Policies for Faster Training Without Forgetting', in *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., (2020).
[33] Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn, 'Offline meta-reinforcement learning with advantage weighting', in *ICML*, (2021).
[34] 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).
[35] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., 'Human-level control through deep reinforcement learning', *nature*, (2015).
[36] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine, 'Awac: Accelerating online reinforcement learning with offline datasets', *arXiv*, (2020).
[37] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter, 'Continual lifelong learning with neural networks: A review', *Neural Networks*, **113**, (2019).
[38] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine, 'Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning', *CoRR*, **abs/1910.00177**, (2019). arXiv: 1910.00177.
[39] David Pfau and Oriol Vinyals, 'Connecting generative adversarial networks and actor-critic methods', *CoRR*, (2016).
[40] Ameya Prabhu, Philip H. S. Torr, and Puneet Kumar Dokania, 'Gdumb: A simple approach that questions our progress in continual learning', in *ECCV*, (2020).
[41] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, G. Sperl, and Christoph H. Lampert, 'icarl: Incremental classifier and representation learning', *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2017).
[42] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne, 'Experience Replay for Continual Learning', in *NIPS*. Curran Associates, Inc., (2019).
[43] Gobinda Saha, Isha Garg, and Kaushik Roy, 'Gradient Projection Memory for Continual Learning', in *ICLR*. OpenReview.net, (2021).
[44] Takuma Seno and Michita Imai, 'd3rlpy: An Offline Deep Reinforcement Learning Library', *CoRR*, **abs/2111.03788**, (2021). arXiv: 2111.03788.
[45] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al., 'Critic regularized regression', *NIPS*, **33**, 7768–7778, (2020).
[46] Maciej Wolczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś, 'Disentangling transfer in continual reinforcement learning', in *NIPS*, (2022).
[47] Maciej Wołczyk, Michal Zajkac, Razvan Pascanu, Lukasz Kuci'nski, and Piotr Milo's, 'Continual world: A robotic benchmark for continual reinforcement learning', in *NeurIPS*, (2021).
[48] Annie Xie, James Harrison, and Chelsea Finn, 'Deep Reinforcement Learning amidst Continual Structured Non-Stationarity', in *ICML*, volume 139 of *Proceedings of Machine Learning Research*. PMLR, (July 2021).
[49] Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang, 'Online coreset selection for rehearsal-based continual learning', in *ICLR*, (2021).
[50] Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Sergey Levine, and Chelsea Finn, 'Conservative data sharing for multi-task offline reinforcement learning', *NIPS*, **34**, 11501–11516, (2021).
[51] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn, 'COMBO: Conservative Offline Model-Based Policy Optimization', in *NIPS*. Curran Associates, Inc., (2021).
[52] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma, 'MOPO: Model-based Offline Policy Optimization', in *NIPS*. Curran Associates, Inc., (2020).
[53] Friedemann Zenke, Ben Poole, and Surya Ganguli, 'Continual Learning Through Synaptic Intelligence', in *ICML*. PMLR, (2017).
[54] Siyuan Zhang and Nan Jiang, 'Towards hyperparameter-free policy selection for offline reinforcement learning', in *NIPS*. Curran Associates, Inc., (2021).
[55] Zifeng Zhuang, Kun Lei, Jinxin Liu, Donglin Wang, and Yilang Guo, 'Behavior proximal policy optimization', *ArXiv*, **abs/2302.11312**, (2023).