TACKLING CONTINUAL OFFLINE RL THROUGH SELECTIVE WEIGHTS ACTIVATION ON ALIGNED SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual offline reinforcement learning (CORL) has shown impressive ability in diffusion-based continual learning systems by modeling the joint distributions of trajectories. However, most research only focuses on limited continual task settings where the tasks have the same observation and action space, which deviates from the realistic demands of training agents in various environments. In view of this, we propose Vector-Quantized Continual Diffuser, named VQ-CD, to break the barrier of different spaces between various tasks. Specifically, our method contains two complementary sections, where the quantization spaces alignment provides a unified basis for the selective weights activation. In the quantized spaces alignment, we leverage vector quantization to align the different state and action spaces of various tasks, facilitating continual training in the same space. Then, we propose to leverage a unified diffusion model attached by the inverse dynamic model to master all tasks by selectively activating different weights according to the task-related sparse masks. Finally, we conduct extensive experiments on 15 continual learning (CL) tasks, including conventional CL task settings (identical state and action spaces) and general CL task settings (various state and action spaces). Compared with 17 baselines, our method reaches the SOTA performance.

1 INTRODUCTION

The endeavor of recovering high-performance policies from abundant offline samples gathered by various sources and continually mastering future tasks learning and previous knowledge maintaining gives birth to the issue of continual offline reinforcement learning (CORL) (Levine et al., 2020; Ada et al., 2024; Huang et al., 2024). Ever-growing scenarios or offline datasets pose challenges for most continual RL methods that are trained on static data and are prone to showing catastrophic forgetting of previous knowledge and ineffective learning of new tasks (Liu et al., 2024; Zhang et al., 2023c; Korycki & Krawczyk, 2021). Facing these challenges, three categories of methods, rehearsal-based (Huang et al., 2024; Peng et al., 2023; Chaudhry et al., 2018), regularization-based (Smith et al., 2023; Zhang et al., 2023b; 2022), and structure-based methods (Zhang et al., 2023a; Marouf et al., 2023; Borsos et al., 2020), are proposed to reduce forgetting and facilitate continual training.

However, most previous studies only focus on the continual learning (CL) setting with identical state and action spaces (Liu et al., 2024; Smith et al., 2023). It deviates from the fact that the ever-growing scenarios or offline datasets are likely to possess different state and action spaces with previous tasks for many reasons, such as the variation of demands and the number of sensors (Yang et al., 2023; Zhang et al., 2023a). Moreover, these datasets often come from multiple behavior policies, which pose the additional challenge of modeling the multimodal distribution of various tasks (Ada et al., 2024; Lee et al., 2024). Benefiting from diffusion models' powerful expressive capabilities and highly competitive performance, an increasing number of researchers are considering incorporating them to address the CORL problems (Ajay et al., 2022; Yue et al., 2024; Elsayed & Mahmood, 2024) from the perspective of sequential modeling. There have been several attempts to combine diffusion-based models with rehearsal-based and regularization-based techniques, which usually apply constraints to the continual model learning process with previous tasks' data or well-trained weights (Smith et al., 2023; Yue et al., 2024; Liu et al., 2024). However, constrained weight updating will limit the learning capability of new tasks and can not preserve the previously acquired knowledge perfectly (Yang et al., 2023). Although structure-based methods can eliminate forgetting and strengthen the learning capability by preserving well-trained weights of previous tasks and reserving disengaged weights for ongoing tasks, they are still limited in simple architecture and CL settings with identical state and action spaces (Zhang et al., 2023a; Wang et al., 2022b; Mallya & Lazebnik, 2018). Thus, in this paper, we seek to answer the question:

Can we merge the merits of diffusion models' powerful expression and structure-based parameters isolation to master CORL problems with any task sequence?

We answer this in the affirmative through the key insight of allocating harmonious weights for each continual learning task. Specifically, we propose Vector-Quantized Continual Diffuser called VQ-CD, which contains two complementary sections: the quantized spaces alignment (QSA) module and the selective weights activation diffuser (SWA) module. To expand our method to any task sequences under the continual learning setting, we adopt the QSA module to align the different state and action spaces. Concretely, we adopt vector quantization to map the task spaces to a unified space for training based on the contained codebook and recover it to the original task spaces for evaluation. In the SWA module, we first perform task mask generation for each task, where the task masks are applied to the one-dimensional convolution kernel of the U-net structure diffusion model. Then, we use the masked kernels to block the influence of unrelated weights during the training and inference. Finally, after the training process, we propose the weights assembling to aggregate the task-related weights together for simplicity and efficiency. In summary, our main contributions are fourfold:

- We propose the Vector-Quantized Continual Diffuser (VQ-CD) framework, which can not only be applied to conventional continual tasks but also be suitable for any continual tasks setting, which makes it observably different from the previous CL method.
- In the quantized spaces alignment (QSA) module of VQ-CD, we adopt ensemble vector quantized encoders based on the constrained codebook because it can be expanded expediently. During the inference, we apply task-related decoders to recover the various observation and action spaces.
- In the selective weights activation (SWA) diffuser module of VQ-CD, we first perform task-related task masks, which will then be used to the kernel weights of the diffuser. After training, we propose assembling weights to merge all learned knowledge.
- Finally, we conduct extensive experiments on 15 CL tasks, including conventional CL settings and any CL task sequence settings. The results show that our method surpasses or matches the SOTA performance compared with 17 representative baselines.

2 Related Work

Offline RL. Offline reinforcement learning is becoming an important direction in RL because it supports learning on large pre-collected datasets and avoids massive demand for expensive, risky interactions with the environments (Mnih et al., 2015; Nair et al., 2020; Cheng et al., 2022; Ball et al., 2023). Directly applying conventional RL methods in offline RL faces the challenge of distributional shift (Schaul et al., 2015; Levine et al., 2020; Xie et al., 2021; Yue et al., 2022) caused by the mismatch between the learned and data-collected policies, which will usually make the agent improperly estimate expectation return on out-of-distribution actions (Schaul et al., 2015; Kostrikov et al., 2021; Ada et al., 2024). To tackle this challenge, previous studies try to avoid the influences of out-of-distribution actions by adopting constrained policy optimization (Peng et al., 2019; Fujimoto et al., 2019; Nair et al., 2020; Kostrikov et al., 2021), behavior regularization (Nachum et al., 2019; Kumar et al., 2020; Ghosh et al., 2022), importance sampling (Jiang & Li, 2016; Hallak & Mannor, 2017; Zhang et al., 2020), uncertainty estimation (Agarwal et al., 2020; Wang et al., 2020a; Lee et al., 2022), and imitation learning (Wang et al., 2020b; Siegel et al., 2020; Chen et al., 2020).

Continual RL. Continual learning (CL) aims to solve the plasticity and stability trade-off under the task setting, where the agent can only learn to solve each task successively (Zhang et al., 2023c; Wang et al., 2023). CL can be classified into task-aware CL and task-free CL according to whether there are explicit task boundaries (Aljundi et al., 2019; Wang et al., 2023). In this paper, we mainly focus on task-aware CL. There are three main technical routes to facilitate forward transfer (plasticity) and mitigate catastrophic forgetting (stability). Rehearsal-based approaches (Shin et al., 2017; Mallya & Lazebnik, 2018; Wang et al., 2022b; Zhang et al., 2023a; Smith et al., 2023) store a portion of samples from previous tasks and use interleaving updates between new tasks' samples and previous tasks' samples. Simply storing samples increases the memory burden in many scenarios; thus, generative models such as diffusion models are introduced to mimic previous data distribution and generate synthetic replay for knowledge maintenance (Zhai et al., 2019; Qi et al., 2023; Gao & Liu, 2023). Regularization-based approaches (Kaplanis et al., 2019; Kessler et al., 2020; Zhang et al., 2022; 2023b) seek to find a proficiency compromise between previous and new tasks by leveraging constraint terms on the total loss function. Usually, additional terms of learning objectives will be adopted to penalize significant changes in the behaviors of models' outputs or the updating of models' parameters (Kirkpatrick et al., 2017; Kaplanis et al., 2019). In the structure-based approaches (Wang et al., 2022b; Kessler et al., 2022; Wang et al., 2022b; Zhang et al., 2023a; Smith et al., 2023; Konishi et al., 2023), researchers usually consider parameter isolation by using sub-networks or task-related neurons to prevent forgetting.

Diffusion RL. Recently, diffusion-based models have shown huge potential in RL under the perspective of sequential modeling (Sohl-Dickstein et al., 2015; Ho et al., 2020; Rombach et al., 2022; Janner et al., 2022; Ajay et al., 2022; Beeson & Montana, 2023). A typical use of diffusion models is to mimic the joint distribution of states and actions, and we usually use state-action value functions as the classifier or class-free guidance when generating decisions (Nichol & Dhariwal, 2021; Ho & Salimans, 2022; Pearce et al., 2023; Liu et al., 2023). Diffusion models, as representative generative models, can also be used as environmental dynamics to model and generate synthetic samples to improve sample efficiency or maintain previous knowledge in CL (Yamaguchi & Fukuda, 2023; Hepburn & Montana, 2024; Lu et al., 2024; Ding et al., 2024; Liu et al., 2024). It is noted that the diffusion model's powerful expression ability on multimodal distribution also makes it suitable for being used as policies to model the distribution of actions and as planners to perform long-horizon planning (Wang et al., 2022a; Kang et al., 2024; Chen et al., 2024). Besides, diffusion models can also be used as multi-task learning models to master several tasks simultaneously (He et al., 2024) or as multi-agent models to solve more complex RL scenarios (Zhu et al., 2023).

3 Preliminary

3.1 Continual Offline RL

We focus on the task-aware CL in the continual offline RL in this paper (Zhang et al., 2023c; Abel et al., 2023; Wang et al., 2023; Smith et al., 2023; Qing et al., 2024; Wang et al., 2024). Suppose that we have I successive tasks, and task j arises behind task i for any $i < j$. Each task $i, i \in [1 : I]$ is represented by a Markov Decision Process (MDP) $\mathcal{M}^i = \langle \mathcal{S}^i, \mathcal{A}^i, \mathcal{P}^i, \mathcal{R}^i, \gamma \rangle$, where we use supscript i to differentiate different tasks, I is the number of total tasks, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, respectively, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. Conventional CL tasks have the same state and action spaces for all tasks, i.e., $|\mathcal{S}^i| = |\mathcal{S}^j|, |\mathcal{A}^i| = |\mathcal{A}^j|, \forall\, i, j \in [1 : I]$. While for any tasks sequences, we have $|\mathcal{S}^i| \neq |\mathcal{S}^j|, |\mathcal{A}^i| \neq |\mathcal{A}^j|$. In the offline RL setting, we can only access pre-collected datasets $\{D^i\}_{i\in[1:I]}$ from each task $\mathcal{M}^i$. The goal of continual offline RL is to find an optimal policy that can maximize the discounted return $\sum_i^I \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r(s_t^i, a_t^i)]$ (Fujimoto & Gu, 2021; Yang et al., 2023; Sun et al., 2023) on all tasks.

3.2 Conditional Generative Behavior Modeling

In this paper, we adopt the diffusion-based model with the U-net backbone as the generative model to fit the joint distribution $q(\tau_s) = \int q(\tau_s^{0:K}) d\tau_s^{1:K}$ of state sequences τ_s and an inverse dynamics model $f_{inv,\psi}(s_t, s_{t+1})$ to produce actions a_t, where $k \in [1 : K]$ is the diffusion step, t is the RL time step, ψ is the parameters of inverse dynamics model, and we omit the identification of tasks for the sake of simplicity because the training is same for all tasks. Through specifying the pre-defined forward diffusion process $q(\tau_s^k|\tau_s^{k-1}) = \mathcal{N}(\tau_s^k; \sqrt{\alpha_k}\tau_s^{k-1}, \beta_k \boldsymbol{I})$ and the trainable reverse process $p_\theta(\tau_s^{k-1}|\tau_s^k) = \mathcal{N}(\tau_s^{k-1}; \mu_\theta(\tau_s^k, k), \Sigma^k)$ (Ho et al., 2020), we can train the diffusion model with the simplified loss function

$$\mathcal{L}(\theta) = \mathbb{E}_{k\sim U(1,2,\ldots,K),\epsilon\sim\mathcal{N}(0,\boldsymbol{I}),\tau_s^0\sim D, b\sim\mathcal{B}(\lambda)}[||\epsilon - \epsilon_\theta(\tau_s^k, k, b * \mathcal{C})||_2^2], \tag{1}$$

where $\tau_s^k = \sqrt{\bar{\alpha}_k}\tau_s^0 + \sqrt{1-\bar{\alpha}_k}\epsilon$, $\mu_\theta(\tau_s^k) = \frac{1}{\sqrt{\alpha_k}}(\tau_s^k - \frac{\beta_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon_\theta(\tau_s^k, k))$, $\Sigma^k = \frac{1-\bar{\alpha}_{k-1}}{1-\bar{\alpha}_k}\beta_k \boldsymbol{I}$, α_k is the approximate discretization pre-defined parameters (Chen et al., 2022; Lu et al., 2023), $\beta_k = 1 - \alpha_k$, $\bar{\alpha}_k = \prod_{\iota=1}^k \alpha_\iota$, U is the uniform distribution, ϵ is standard Gaussian noise, $\boldsymbol{I}$ is the identity matrix, $\tau_s^0 \sim D$ is the state sequences stored in the task replay buffer D, $\mathcal{B}$ is binomial

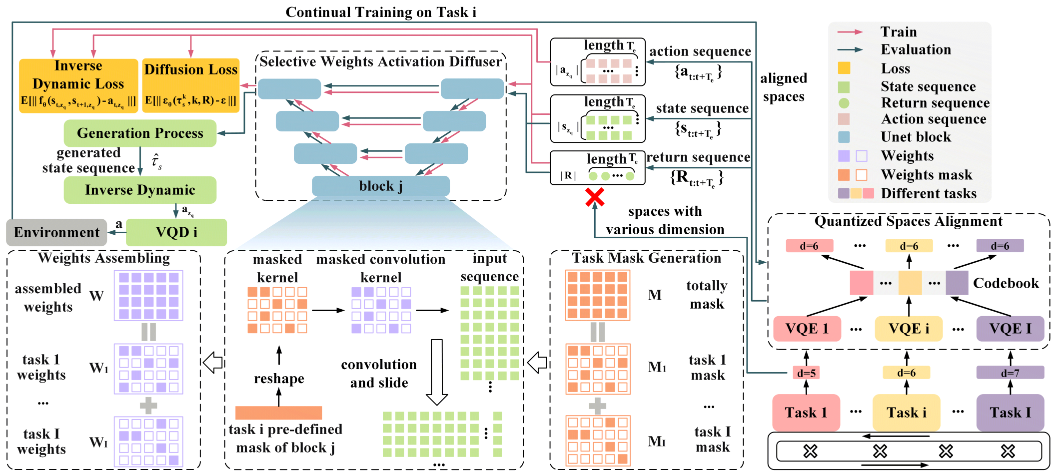

Figure 1: The framework of VQ-CD. It contains two sections: The Quantized Space Alignment (QSA) module and the Selective Weights Activation (SWA) module, where QSA enables our method to adapt for any continual learning task setting by transferring the different state and action spaces to the same spaces. SWA uses selective neural network weight activation to maintain the knowledge of previous tasks through task-related weight masks. After the training, we perform weights assembling to integrate the total weights and save the memory budget.

distribution, $\lambda = 0.25$ is the parameter of $\mathcal{B}$, $\mathcal{C}$ is condition, which is usually selected as discounted returns or value function in RL, and θ is the total parameters of model ϵ_θ. The following is

$$\hat{\tau}_s^{k-1} = \frac{1}{\sqrt{\alpha_k}}(\hat{\tau}_s^k - \frac{\beta_k}{\sqrt{1-\bar{\alpha}_k}}\hat{\epsilon}) + \sqrt{\frac{1-\bar{\alpha}_{k-1}}{1-\bar{\alpha}_k}\beta_k}\epsilon. \tag{2}$$

generation function, where we use $\hat{\tau}_s^k$ to denote the generated state sequences, $\hat{\epsilon} = \epsilon_\theta(\hat{\tau}_s^k, k, \emptyset) + \omega(\epsilon_\theta(\hat{\tau}_s^k, k, \mathcal{C}) - \epsilon_\theta(\hat{\tau}_s^k, k, \emptyset))$, ω is the guidance scale, $\emptyset$ means $b = 0$. We use inverse dynamics model $f_{inv,\psi}(\cdot)$ to produce actions, where the training loss is

$$\mathcal{L}(\psi) = \mathbb{E}_{(s_t, a_t, s_{t+1})\sim D}[||a_t - f_{inv,\psi}(s_t, s_{t+1})||_2^2]. \tag{3}$$

4 Method

Our method enables training on any CL task sequences through two sections (as shown in Figure 1): the selective weights activation diffuser (SWA) module and the quantized spaces alignment (QSA) module. Algorithm 1 shows how to generate the actions during inference. The detailed training process is shown in Algorithm 2 of Appendix A.1. In the following parts, we introduce these two modules in detail.

Algorithm 1: Evaluation Process

```
for For each environmental step t in task i do
    Receive the environmental state s_t^i
    Set the return condition R = 0.8
    s_{t,z_q} = f^i_{s,θ_q}(f^i_{VQE_s}(s_t^i; θ_e))
    Construct τ̂^K_{s_{z_q}} = [s_{t,z_q}, ŝ^K_{t+1,z_q}, ŝ^K_{t+2,z_q}, ...],
    where ŝ^K_{t',z_q} ~ N(0, I) for t' > t.
    for For k from K to 1 do
        Calculate ε̂ with ε_θ
        Obtain τ̂^{k-1}_{s_{z_q}} with Equation 2
        Replace the first state of τ̂^{k-1}_{s_{z_q}} with s_t
    end for
    Extract [s_{t,z_q}, ŝ_{t+1,z_q}] from τ̂^0_s
    Obtain a_{t,z_q} = f_{inv}(s_{t,z_q}, ŝ_{t+1,z_q})
    Interact with a_t^i = f^i_{VQD_a}(a_{t,z_q}; θ_d)
end for
```

4.1 Quantized Spaces Alignment

To make our method suitable for solving any CL task sequence setting, we propose aligning the different state and action spaces with the quantization technique. Specifically, we propose to solve the following quantized representation learning problem

$$\begin{aligned} \min_{\theta_e, \theta_d, \theta_q} \quad & \mathcal{L}_{QSA}(x; \theta_e, \theta_d, \theta_q), \\ \text{s.t.} \quad & ||z_q||_2^2 < \rho, \end{aligned} \tag{4}$$

where $\mathcal{L}_{QSA}(x) = \mathbb{E}\left[||x - f_{VQD}(z_q;\theta_d)||_2^2\right] + \mathbb{E}\left[||\mathbf{sg}(z_q) - z_e||_2^2\right] + \mathbb{E}\left[||\mathbf{sg}(z_e) - z_q||_2^2\right]$ is the total quantized loss, $\mathbf{sg}(\cdot)$ represents the stop gradient operation, θ_e and θ_d are the parameters of the vector quantized encoder (VQE) and vector quantized decoder (VQD), θ_q is the parameters of the codebook, ρ limits the range of codebook embeddings, x can represent the states or actions for each specific CL task, $z_q = f_{\theta_q}(z_e)$ is the quantized representation which is consisted of fixed number of fixed-length quantized vectors, and $z_e = f_{VQE}(x;\theta_e)$ is the output of the encoder. Here, we propose searching the constrained optimal solution of the above problem for the consideration of the diffusion model training within a limited value range, just like the limit normalization in CV (Ho et al., 2020; Dhariwal & Nichol, 2021) and RL (Ajay et al., 2022; Lu et al., 2023). There are many methods to force optimization under restricted constraints, such as converting the constraints to a penalty term (Boyd & Vandenberghe, 2004). In our method, for simplicity and convenience, we propose to directly clip the quantized vector z_q to meet the constraints after every codebook updating step. Moreover, to meet the potential demand for extra tasks beyond the predefined CL tasks, we design the codebook as easy to equip, where the quantized spaces of different tasks are separated so that we can expediently train new task-related encoders, decoders, and quantized vectors.

For tasks where the state and action spaces are different, we can use the well-trained QSA module to obtain the aligned state feature $s^i_{z_q} = f^i_{s,\theta_q}(f^i_{VQE_s}(s^i;\theta_e))$ and the action feature $a^i_{z_q} = f^i_{a,\theta_q}(f^i_{VQE_a}(a^i;\theta_e))$ for each task i. Thus, we can use $\tau^i_{s_{z_q}}$ and $\tau^i_{a_{z_q}}$ to represent the state and action feature sequences. Now, the action is produced through $a^i_t = f^i_{VQD_a}(f_{inv}(s_{z_q,t}, s_{z_q,t+1});\theta_d)$.

4.2 Selective Weights Activation

In this section, we introduce how to selectively activate different parameters of the diffusion model to reduce catastrophic forgetting and reserve disengaged weights for ongoing tasks.

Task Mask Generation. Suppose that the diffusion model contains L blocks, and the weights (i.e., parameters) of block l are denoted by $W_l, l \in \{1, ..., L\}$. There are two ways to disable the influence of the weights on the model outputs. One is masking the output neurons $O_l = f_l(\cdot; W_l)$ of each block, where $f_l(\cdot)$ is the neural network function of block l. This strategy is friendly to MLP-based neural networks for two reasons: 1) the matrix calculation, such as $W_l * x$, is relatively simple so that we can easily recognize the disabled weights; 2) we do not need to apply any special operation on the optimizer because the output masking will cut off gradient flow naturally. However, we can not arbitrarily apply the above masking strategy to more expressive network structures, such as convolution-based networks, because we can not easily distinguish the dependency between parameters and outputs. Thus, we search for another masking strategy: masking the parameters W_l with M_l, which permits us to control each parameter accurately.

Specifically, suppose that the total available mask positions of block l are M_l. In this paper, M_l is a ones matrix, and the entries with 0 mean that we will perform masking. Before training on task i, we first pre-define the specific mask $M_{i,l}$ of task i on block l by randomly sampling unmasked positions from the remaining available mask positions. Then, with the increase of the tasks, the remaining available mask positions decrease until $M_l = \sum_{i=1}^{I} M_{i,l}$.

Selective Weights Forward and Backward Propagation. After obtaining the mask $M_{i,l}$, we can perform forward propagation with masked weights

$$\begin{aligned} &\epsilon_\theta(\tau_\cdot^k, k, \mathcal{C}) = f_L(f_{L-1}(...(f_1(\cdot)))) \\ &O_{l+1} = f_{l+1}(O_l, k; M_{i,l+1} \circ W_{l+1}), O_0 = \tau^{i,k}_{s_{z_q}} / \tau^{i,k}_{a_{z_q}}, \end{aligned} \tag{5}$$

where ϵ_θ is the noise prediction model introduced in Equation 1, and $M_{i,l+1} \circ W_{l+1}$ represents the pairwise product. $\tau^{i,k}_{s_{z_q}}$ and $\tau^{i,k}_{a_{z_q}}$ denote the perturbed state or action sequences of task i at diffusion step k. Through forward designing, we can selectively activate different weights for different tasks through the mask $M_{i,l+1}$, thus preserving previously acquired knowledge and reserving disengaged weights for other tasks. Though we can expediently calculate the masked output O_{l+1} during forward propagation with weights or neurons masking, it poses a challenge to distinguishing the dependency from weights to loss and updating the corresponding weights during the backward propagation. In order to update the corresponding weights, we realize two methods. 1) Intuitively, we propose to update the neural network with the sparse optimizer rather than the dense optimizer Diederik (2014), where the position and values of the parameters are recorded to update the corresponding weights. However, in the implementation, we find that the physical time consumption of the sparse optimizer is intractable (Refer to Table 6 of Appendix B.5 for more details.), which encourages us to find a more straightforward and convenient method. 2) Thus, we propose extracting and assembling the corresponding weights at the end of the training rather than updating the corresponding weights during training. This choice brings two benefits: (1) It can significantly reduce the time consumption spent on training. (2) It is friendly to implementation on complex network structures.

Weights Assembling. Assembling weights after training permits us to save the total acquired knowledge and do not need extra memory budgets. Concretely, after training on task i, we will obtain the weights W_i, which can be extracted with the mask M_i from the total weights $W[i * \Omega]$, including all the diffusion model weights. We use W_i to denote the weights related to task i, Ω is the training step on each CL task, and $W[i * \Omega]$ represents the total weight checkpoint at training step $i * \Omega$. Then, at the end of the training, we can assemble weights $\{W_i | i \in I\}$ by simply adding these weights together because of the exclusiveness property, i.e., $W = \sum_{i=1}^{I} W_i = \sum_{i=1}^{I} M_i \circ W[i*\Omega]$.

5 Experiments

In this section, we will introduce environmental settings, evaluation metrics, and baselines in the following sections. Then, we will report and analyze the comparison results, ablation study, and parameter sensitivity analysis. Other implementation details are shown in Appendix A.2 and A.3.

5.1 Environmental Settings

Following previous studies (Zhang et al., 2023c; Yang et al., 2023), we select MuJoCo Ant-dir and Continual World (CW) to formulate traditional CL settings with the same state and action spaces. In Ant-dir, we select several tasks, such as 10-15-19-25 and 4-18-26-34-42-49, for training and evaluation. In CW, we adopt the task setting of CW10, which contains 10 robotic manipulation tasks for CL performance comparison. Additionally, we propose to leverage D4RL tasks (Fu et al., 2020) to construct the CL settings with diverse state and action spaces. We select the Hopper, Walker2d, and HalfCheetah as elements to construct CL tasks, where each environment among Hopper, Walker2d, and HalfCheetah contains 6 qualities (random, medium, expert, medium-expert, medium-replay, and full-replay) datasets.

5.2 Evaluation Metrics

Considering the various reward structures of different environments, we should adopt different performance comparison metrics. For Ant-dir, we adopt the average episodic return over all tasks as the performance comparison, i.e., the final performance $P = \text{mean}(\sum_i R_i)$ is calculated based on the task i's return R_i. In the CW environment, previous works (Wołczyk et al., 2021; Anand & Precup, 2023) usually adopt the success rate Ψ as the performance metric. Thus, we adopt the average success rate on all tasks as the final performance, i.e., $P = \text{mean}(\sum_i \Psi_i)$. For the D4RL environments, we use the normalized score Φ (Wang et al., 2022a; Huang et al., 2024) as the metric to calculate the performance $P = \text{mean}(\sum_i \Phi_i)$, where $\Phi_i = \frac{R_i - R_{random}}{R_{expert} - R_{random}} * 100$. Usually, we can use the interface of these environments to obtain the score expediently.

5.3 Baselines

We select various representative CL baselines, which can be classified into diffusion-based and non-diffusion-based methods. For example, the diffusion-based methods consist of CRIL (Gao et al., 2021), DGR (Shin et al., 2017), t-DGR (Yue et al., 2024), MTDIFF (He et al., 2023), CuGRO (Liu et al., 2024), CoD (Hu et al., 2024), and CoD variants. The non-diffusion-based methods include L2M (Schmied et al., 2024), EWC (Kirkpatrick et al., 2017), PackNet (Mallya & Lazebnik, 2018), Finetune, IL-rehearsal (Wan et al., 2024), and Multitask. From the perspective of mainstream CL classification standards, these baselines can also be sorted as rehearsal-based methods (CRIL, DGR, t-DGR, CoD, and IL-rehearsal), regularization-based methods (L2M, EWC, CuGRO, and Finetune), and structure-based methods (PackNet, Multitask, and MTDIFF).

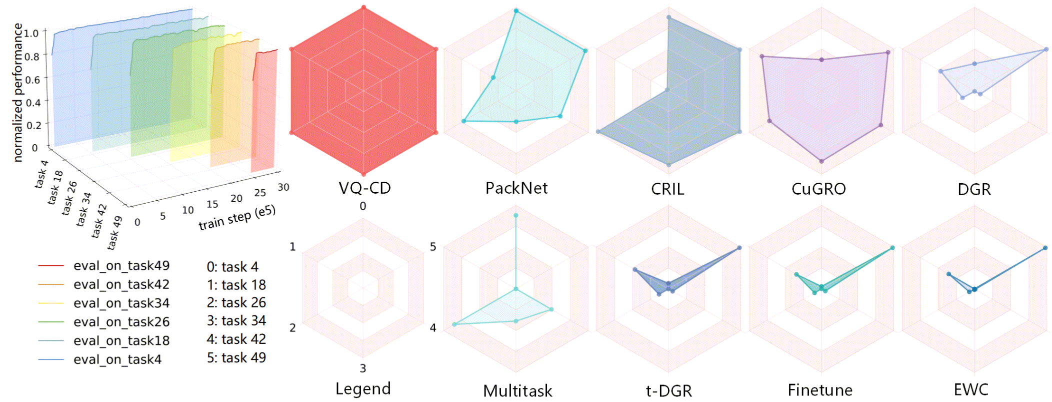

Figure 2: The comparison of VQ-CD and several baselines on the continual tasks setting (Ant-dir task 4-18-26-34-42-49). We train on each task for 500k steps. We report the normalized evaluation performance of VQ-CD in the top left corner, where the coordinates, e.g., task 4, represent evaluation on task 4 at different training tasks. To show the overall performance on all tasks, we show the normalized evaluation performance on the six tasks after finishing the training at the right part.

Table 1: The comparison of VQ-CD, diffusion-based baselines, and LoRA methods on Ant-dir tasks, where the continual task sequence is 10-15-19-25. The results are average on 30 evaluation rollouts with 30 random seeds.

Method	VQ-CD **(ours)**	CoD	Multitask CoD	IL-rehearsal	CoD-LoRA	Diffuser-w/o rehearsal	CoD-RCR	MTDIFF	DD-w/o rehearsal
Mean return	558.22$\pm$1.14	478.19$\pm$15.84	485.15$\pm$5.86	402.53$\pm$17.67	296.03$\pm$11.95	270.44$\pm$5.54	140.44$\pm$32.11	84.01$\pm$41.10	-11.15$\pm$45.27

5.4 Experimental Results

In this section, we mainly separate the experimental settings into two categories, the traditional CL settings with the same state and action spaces and the arbitrary CL settings with different state and action spaces, to show the effectiveness of our method. Besides, we also investigate the influence of the alignment techniques, such as auto-encoder, variational auto-encoder, vector-quantized variational auto-encoder (we adopt this in our method). More deeply, we investigate how to deal with the potential demand for additional tasks beyond the pre-defined task length by releasing nonsignificant masks or expanding more available weights (Refer to Appendix B.4 for more details.).

The traditional CL settings correspond to the first question we want to answer: *Can VQ-CD achieve superior performance compared with previous methods in the traditional CL tasks?*

We use Ant-dir and Continual World (Zhang et al., 2023c; Yang et al., 2023) to formulate the continual task sequence, where we select two types of task sequence in Ant-dir and "hammer-v2, push-wall-v2, faucet-close-v2, push-back-v2, stick-pull-v2, handle-press-side-v2, push-v2, shelf-place-v2, window-close-v2, peg-unplug-side-v2" to construct CW10 CL setting. For simplicity, we do not align the state and action spaces with quantized alignment techniques because the traditional CL setting naturally has the same spaces. The comparison results between our method and several diffusion-based baselines are shown in Table 1, where these baselines include rehearsal-based (CoD and IL-rehearsal), parameter-sharing (CoD-LoRA), multitask training (Multitask CoD and MTDIFF), and representative diffusion RL methods (Diffuser-w/o rehearsal, CoD-RCR, and DD-w/o rehearsal). Our method surpasses all baselines in the Ant-dir setting by a large margin in Figure 2, which directly shows the effectiveness of our method. As another experiment of CL setting with the same state and action spaces, we report the results in Figure 7. Compared with the upper bound performance of Multitask, our method reaches the same performance after the CL training. With the increasing of new tasks, our method continually masters new tasks and sustains the performance while the baselines show varying degrees of performance attenuation, which can be found in the fluctuation of the curves. Moreover, the final performance difference between one method and the

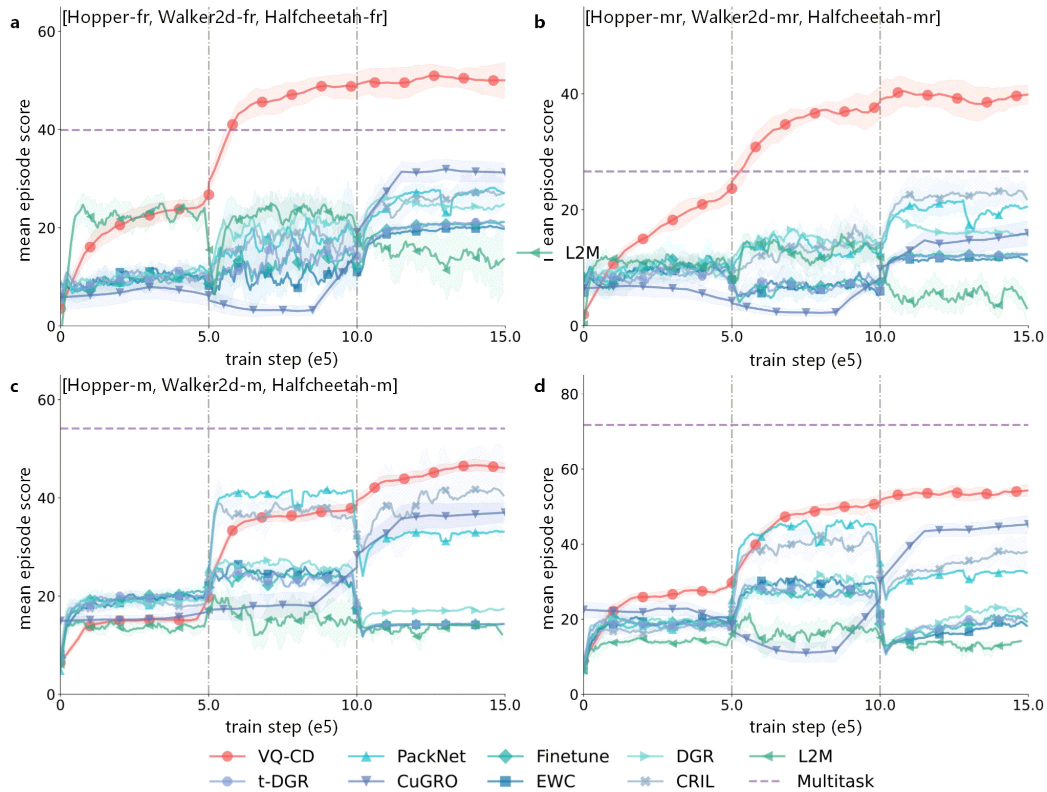

Figure 3: The comparison on the arbitrary CL settings. We select the D4RL tasks to formulate the CL task sequence. We leverage state and action padding to align the spaces. The experiments are conducted on various dataset qualities, where the results show that our method surpasses the baselines not only at the expert datasets but also at the non-expert datasets. The datasets characteristic "fr", "mr", "m", and "me" represent "full-replay", "medium-replay", "medium", and "medium-expert", respectively. "Hopper", "Walker2d", and "Halfcheetah" are the different environments.

Table 2: The feature difference between the aligned features produced by the space alignment module. We randomly sample thousands of aligned state and action features to calculate the difference.

Method	VQ-CD		AE-CD	
feature difference	state difference	action difference	state difference	action difference
[Hopper-fr,Walker2d-fr,Halfcheetah-fr]	8.83±1.98	4.54±0.74	51.31±26.91	14.06±2.09
[Hopper-mr,Walker2d-mr,Halfcheetah-mr]	9.03±1.97	4.45±0.74	48.12±21.94	15.39±3.71
[Hopper-m,Walker2d-m,Halfcheetah-m]	8.53±1.56	4.22±0.79	42.27±24.29	13.59±2.63
[Hopper-me,Walker2d-me,Halfcheetah-me]	8.93±2.00	4.05±0.56	57.91±36.94	13.93±3.20

Multitask method indicates the forgetting character, which can be reflected by the overall upward trend of these curves. More experiments of shuffling task orders can be found in Appendix B.2.

The arbitrary CL settings correspond to the second question we want to answer: *Can we use the proposed space alignment method to enable VQ-CD to adapt to incoming tasks with various spaces?*

To answer the above question, we select D4RL to formulate the CL task sequence because of the various state and action spaces, and the results are shown in Figure 3. Considering the dataset qualities of D4RL (Fu et al., 2020), we choose different dataset quality settings and report the mean episode score that is calculated with $\frac{R_i - R_{random}}{R_{expert} - R_{random}} * 100$. Generally, from the four sub-experiments (**a**, **b**, **c**, and **d**), we can see that our method (VQ-CD) surpasses these baselines in all CL settings. Especially in the CL settings (Figure 3 **a** and **b**), where the datasets contain low-quality trajectories, our method achieves a large performance margin even compared with the Multitask method. We can attribute the reason to the return-based action generation that helps our method distinguish different quality trajectories and generate high-reward actions during

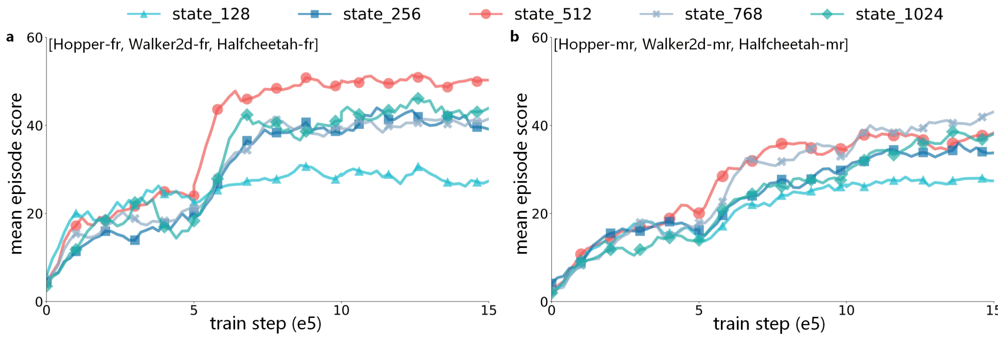

Figure 5: The effects of different codebook sizes about the states.

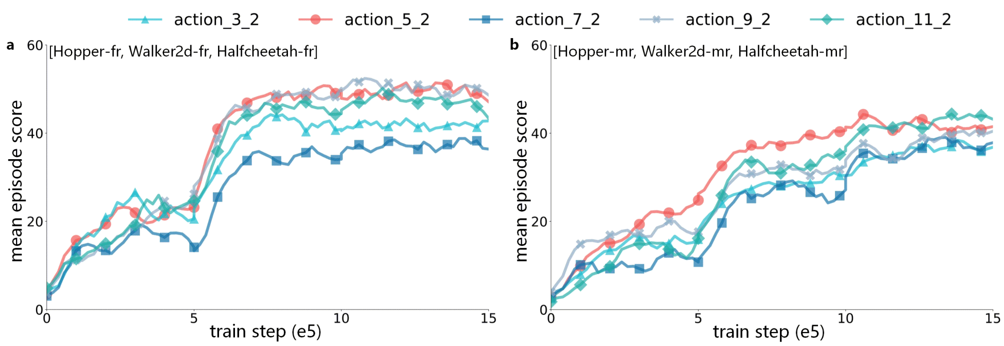

Figure 6: The effects of the number of latent vectors about the actions.

evaluation, as well as the selective weights activation that can reserve the previous knowledge and reduce forgetting. While other methods just possess the ability to continue learning and lack the ability to separate different qualities and actions, thus leading to poor performance. For trajectory qualities that are similar across the datasets (Figure 3 **c** and **d**), we can see lower improvement gains between our method and baselines. However, it should be noted that our method can still reach better performance than other baselines. Apart from the padding alignment, we also conduct experiments (Figure 11) on baselines that adopt our pre-trained QSA module to align state and action spaces in Appendix B.3.

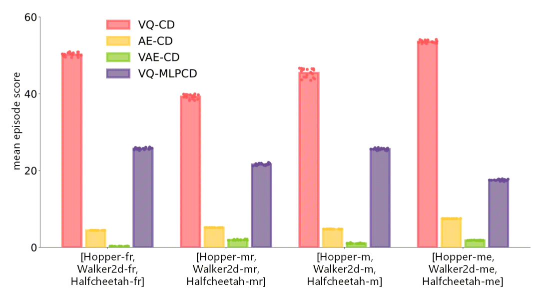

Figure 4: The ablation study of space alignment module and diffusion network structure. For each type of ablation study, we fix the other same and retrain the model on four D4RL CL settings.

5.5 Ablation Study

In this section, we want to investigate the influence of different modules of VQ-CD. Thus, the experiments contain two investigation directions: space alignment module ablation study and diffuser network structure ablation study. To show the importance of vector quantization, we change the space alignment module with auto-encoder (AE) and variational auto-encoder (VAE). Based on this modification, we retrain our method and report the results in Figure 4. The results show significant improvements in the D4RL CL settings, illustrating the importance and effectiveness of vector quantization in our method. Compared with AE-CD, VAE-CD performs poorer on all D4RL CL settings. The reason lies in that the implicit Gaussian constraint on each dimension may hurt the space alignment. Compared with the codebook in VQ-CD, AE-CD may cause a bigger difference between aligned features produced by AE (shown in Table 2), posing challenges for the diffusion model to model the distribution of the aligned features and leading to low performance. As for the diffuser network structure, we conduct the selective weights activation on the mlp-based and unet-based diffusion models. The latter structure is beneficial to making decisions with temporal information inside the trajectories, leading to higher performance evaluation.

5.6 Parameter Sensitivity Analysis

When performing on the aligned feature with diffusion models, the hyperparameters of state and action of the quantized spaces alignment module matter. Usually, the complexity of states is more significant than the actions, so the codebook size controls the performance of reconstruction. Thus, we investigate the effect of different codebook sizes and report the results in Figure 5. Obviously, a small codebook size limits performance, and a negative effect arises when it exceeds a certain value, such as 512. As for the actions, we believe the actions can be decomposed into several small latent vectors, and the number of latent vectors is crucial for reconstructing actions. Similarly, we also see the same trend in Figure 6, which inspires us that more latent vectors are not always better.

6 Discussion

The Interplay of VQ and CD. In this paper, we investigate broadening the application scenarios of the same state and action spaces to tasks of arbitrary state and action spaces by space alignment. Vector quantization is verified as one effective way to achieve space alignment compared with AE, VAE, and padding. Furthermore, we adopt the diffusion model to perform continual learning based on VQ due to its strong model expressiveness and competitive performance. The ablation study illustrates that integrating VQ and CD induces the proposed powerful method VQ-CD.

The Intuition of Constraint in QSA Module. In Equation 4, We add a constraint to encourage a more concentrated distribution of the quantized representation vectors as shown in Table 2, which benefits the diffusion model in learning the data distribution in a limited range (Ho et al., 2020; Ajay et al., 2022). However, this may not necessarily benefit other methods that do not focus on modeling distributions (Refer to Figure 11 in Appendix B.3.) because concentrated representations can make originally dissimilar state and action vectors from different tasks appear more similar, making them harder to distinguish and learn. We use the clip operation rather than convert the constraint to a penalty because our goal is to ensure that the magnitude of the quantized representation vectors does not exceed a certain value rather than minimize the norm of constraint.

Further Discussion of Experiments. In Figure 3 (a) and (b), we can see that VQ-CD surpasses Multitask. The reason is that the datasets contain trajectories collected from the entire training process, i.e., from a random policy to a well-trained policy. Our method leverages accumulated discounted returns to guide the generation of state sequences, encouraging the generation of higher-return state sequences. Consequently, the actions generated by the inverse dynamics model also yield higher returns. In contrast, Multitask does not currently incorporate returns, resulting in lower performance. In Figures 3 (c) and (d), the variance of trajectory returns in the dataset is smaller, allowing Multitask to achieve better learning outcomes.

7 Conclusion

In this paper, we propose Vector-Quantized Continual Diffuser, called VQ-CD, which opens the door to training on any CL task sequences. The advantage of this general ability to adapt to any CL task sequences stems from the two sections of our framework: the selective weights activation diffuser (SWA) module and the quantized spaces alignment (QSA) module. SWA preserves the previous knowledge by separating task-related parameters with task-related masking. QSA aligns the different state and action spaces so that we can perform training in the same aligned space. Finally, we show the superiority of our method by conducting extensive experiments, including conventional CL task settings (identical state and action spaces) and general CL task settings (various state and action spaces). The results illustrate that our method achieves the SOTA performance by comparing with 17 baselines on 15 continual learning task settings.

REFERENCES

David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. *arXiv preprint arXiv:2307.11046*, 2023.

Suzan Ece Ada, Erhan Oztop, and Emre Ugur. Diffusion policies for out-of-distribution generalization in offline reinforcement learning. *IEEE Robotics and Automation Letters*, 2024.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International conference on machine learning*, pp. 104–114. PMLR, 2020.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11254–11263, 2019.

Nishanth Anand and Doina Precup. Prediction and control in continual reinforcement learning. *arXiv preprint arXiv:2312.11669*, 2023.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023.

Alex Beeson and Giovanni Montana. Balancing policy constraint and ensemble size in uncertainty-based offline reinforcement learning. *arXiv preprint arXiv:2303.14716*, 2023.

Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *Advances in neural information processing systems*, 33:14879–14890, 2020.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. Simple hierarchical planning with diffusion. *arXiv preprint arXiv:2401.02644*, 2024.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022.

Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith Ross. Bail: Best-action imitation learning for batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18353–18363, 2020.

Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. Adversarially trained actor critic for offline reinforcement learning. In *International Conference on Machine Learning*, pp. 3852–3878. PMLR, 2022.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

P Kingma Diederik. Adam: A method for stochastic optimization. *(No Title)*, 2014.

Zihan Ding, Amy Zhang, Yuandong Tian, and Qinqing Zheng. Diffusion world model. *arXiv preprint arXiv:2402.03570*, 2024.

Mohamed Elsayed and A Rupam Mahmood. Addressing loss of plasticity and catastrophic forgetting in continual learning. *arXiv preprint arXiv:2404.00781*, 2024.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Chongkai Gao, Haichuan Gao, Shangqi Guo, Tianren Zhang, and Feng Chen. Cril: Continual robot imitation learning via generative and prediction model. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6747–5754. IEEE, 2021.

Rui Gao and Weiwei Liu. Ddgr: Continual learning with deep diffusion-based generative replay. In *International Conference on Machine Learning*, pp. 10744–10763. PMLR, 2023.

Dibya Ghosh, Anurag Ajay, Pulkit Agrawal, and Sergey Levine. Offline rl policies should be trained to be adaptive. In *International Conference on Machine Learning*, pp. 7513–7530. PMLR, 2022.

Assaf Hallak and Shie Mannor. Consistent on-line off-policy evaluation. In *International Conference on Machine Learning*, pp. 1372–1383. PMLR, 2017.

Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *arXiv preprint arXiv:2305.18459*, 2023.

Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *Advances in neural information processing systems*, 36, 2024.

Charles A Hepburn and Giovanni Montana. Model-based trajectory stitching for improved behavioural cloning and its applications. *Machine Learning*, 113(2):647–674, 2024.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Jifeng Hu, Li Shen, Sili Huang, Zhejian Yang, Hechang Chen, Lichao Sun, Yi Chang, and Dacheng Tao. Continual diffuser (cod): Mastering continual offline reinforcement learning with experience rehearsal. *arXiv preprint arXiv:2409.02512*, 2024.

Kaixin Huang, Li Shen, Chen Zhao, Chun Yuan, and Dacheng Tao. Solving continual offline reinforcement learning with decision transformer. *arXiv preprint arXiv:2401.08478*, 2024.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International conference on machine learning*, pp. 652–661. PMLR, 2016.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy consolidation for continual reinforcement learning. *arXiv preprint arXiv:1902.00255*, 2019.

Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J Roberts. Unclear: A straightforward method for continual reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J Roberts. Same state, different task: Continual reinforcement learning without interference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7143–7151, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Tatsuya Konishi, Mori Kurokawa, Chihiro Ono, Zixuan Ke, Gyuhak Kim, and Bing Liu. Parameter-level soft-masking for continual learning. In *International Conference on Machine Learning*, pp. 17492–17505. PMLR, 2023.

Lukasz Korycki and Bartosz Krawczyk. Class-incremental experience replay for continual learning under concept drift. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3649–3658, 2021.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Dongsu Lee, Chanin Eom, and Minhae Kwon. Ad4rl: Autonomous driving benchmarks for offline reinforcement learning with value-based dataset. *arXiv preprint arXiv:2404.02429*, 2024.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pp. 1702–1712. PMLR, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review. *and Perspectives on Open Problems*, 5, 2020.

Jinm ei Liu, Wenbin Li, Xiangyu Yue, Shilin Zhang, Chunlin Chen, and Zhi Wang. Continual offline reinforcement learning via diffusion-based dual generative replay. *arXiv preprint arXiv:2404.10662*, 2024.

Xihui Liu, Dong Huk Park, Samaneh Azadi, Gong Zhang, Arman Chopikyan, Yuxiao Hu, Humphrey Shi, Anna Rohrbach, and Trevor Darrell. More control for free! image synthesis with semantic diffusion guidance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 289–299, 2023.

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*, pp. 22825–22855. PMLR, 2023.

Cong Lu, Philip Ball, Yee Whye Teh, and Jack Parker-Holder. Synthetic experience replay. *Advances in Neural Information Processing Systems*, 36, 2024.

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*, pp. 7765–7773, 2018.

Imad Eddine Marouf, Subhankar Roy, Enzo Tartaglione, and Stéphane Lathuilière. Weighted ensemble models are strong continual learners. *arXiv preprint arXiv:2312.08977*, 2023.

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*, 518(7540):529–533, 2015.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, et al. Imitating human behaviour with diffusion models. *arXiv preprint arXiv:2301.10677*, 2023.

Liangzu Peng, Paris Giampouras, and René Vidal. The ideal continual learner: An agent that never forgets. In *International Conference on Machine Learning*, pp. 27585–27610. PMLR, 2023.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Daiqing Qi, Handong Zhao, and Sheng Li. Better generative replay for continual federated learning. *arXiv preprint arXiv:2302.13001*, 2023.

Yunpeng Qing, Jingyuan Cong, Kaixuan Chen, Yihe Zhou, Mingli Song, et al. Advantage-aware policy optimization for offline reinforcement learning. *arXiv preprint arXiv:2403.07262*, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Thomas Schmied, Markus Hofmarcher, Fabian Paischer, Razvan Pascanu, and Sepp Hochreiter. Learning to modulate pre-trained models in rl. *Advances in Neural Information Processing Systems*, 36, 2024.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

James Seale Smith, Yen-Chang Hsu, Lingyu Zhang, Ting Hua, Zsolt Kira, Yilin Shen, and Hongxia Jin. Continual diffusion: Continual customization of text-to-image diffusion with c-lora. *arXiv preprint arXiv:2304.06027*, 2023.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yanchao Sun, Shuang Ma, Ratnesh Madaan, Rogerio Bonatti, Furong Huang, and Ashish Kapoor. Smart: Self-supervised multi-task pretraining with control transformers. *arXiv preprint arXiv:2301.09816*, 2023.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Weikang Wan, Yifeng Zhu, Rutav Shah, and Yuke Zhu. Lotus: Continual imitation learning for robot manipulation through unsupervised skill discovery. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 537–544. IEEE, 2024.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020a.

Yuanfu Wang, Chao Yang, Ying Wen, Yu Liu, and Yu Qiao. Critic-guided decision transformer for offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 15706–15714, 2024.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022a.

Zhenyi Wang, Li Shen, Tiehang Duan, Qiuling Suo, Le Fang, Wei Liu, and Mingchen Gao. Distributionally robust memory evolution with generalized divergence for continual learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Zhi Wang, Chunlin Chen, and Daoyi Dong. A dirichlet process mixture of robust task models for scalable lifelong reinforcement learning. *IEEE Transactions on Cybernetics*, 2022b.

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. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020b.

Maciej Wołczyk, Michał Zajac, Razvan Pascanu, Łukasz Kucinski, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

Tengyang Xie, Nan Jiang, Huan Wang, Caiming Xiong, and Yu Bai. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. *Advances in neural information processing systems*, 34:27395–27407, 2021.

Shin'ya Yamaguchi and Takuma Fukuda. On the limitation of diffusion models for synthesizing training datasets. *arXiv preprint arXiv:2311.13090*, 2023.

Yijun Yang, Tianyi Zhou, Jing Jiang, Guodong Long, and Yuhui Shi. Continual task allocation in meta-policy network via sparse prompting. In *International Conference on Machine Learning*, pp. 39623–39638. PMLR, 2023.

William Yue, Bo Liu, and Peter Stone. t-dgr: A trajectory-based deep generative replay method for continual learning in decision making. *arXiv preprint arXiv:2401.02576*, 2024.

Yang Yue, Bingyi Kang, Xiao Ma, Zhongwen Xu, Gao Huang, and Shuicheng Yan. Boosting offline reinforcement learning via data rebalancing. *arXiv preprint arXiv:2210.09241*, 2022.

Mengyao Zhai, Lei Chen, Frederick Tung, Jiawei He, Megha Nawhal, and Greg Mori. Lifelong gan: Continual learning for conditional image generation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2759–2768, 2019.

Qizhe Zhang, Bocheng Zou, Ruichuan An, Jiaming Liu, and Shanghang Zhang. Split & merge: Unlocking the potential of visual adapters via sparse training. *arXiv preprint arXiv:2312.02923*, 2023a.

Ruiyi Zhang, Bo Dai, Lihong Li, and Dale Schuurmans. Gendice: Generalized offline estimation of stationary values. *arXiv preprint arXiv:2002.09072*, 2020.

Tiantian Zhang, Xueqian Wang, Bin Liang, and Bo Yuan. Catastrophic interference in reinforcement learning: A solution based on context division and knowledge distillation. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Tiantian Zhang, Zichuan Lin, Yuxing Wang, Deheng Ye, Qiang Fu, Wei Yang, Xueqian Wang, Bin Liang, Bo Yuan, and Xiu Li. Dynamics-adaptive continual reinforcement learning via progressive contextualization. *IEEE Transactions on Neural Networks and Learning Systems*, 2023b.

Tiantian Zhang, Kevin Zehua Shen, Zichuan Lin, Bo Yuan, Xueqian Wang, Xiu Li, and Deheng Ye. Replay-enhanced continual reinforcement learning. *arXiv preprint arXiv:2311.11557*, 2023c.

Zhengbang Zhu, Minghuan Liu, Liyuan Mao, Bingyi Kang, Minkai Xu, Yong Yu, Stefano Ermon, and Weinan Zhang. Madiff: Offline multi-agent learning with diffusion models. *arXiv preprint arXiv:2305.17330*, 2023.

Appendix of "Tackling Continual Offline RL through Selective Weights Activation on Aligned Spaces"

A Algorithm

A.1 Pseudocode of VQ-CD

Algorithm 2: Vector-Quantized Continual Diffuser (VQ-CD)

Input: Noise prediction model ϵ_θ, inverse dynamic model $f_{inv,\psi}$, state and action quantized model $f_q(\theta_e, \theta_d, \theta_q)$, tasks set $\mathcal{M}_i, i \in \{1, ..., I\}$, each task training step Ω, max diffusion step K, sequence length T_e, state dimension d_s, action dimension d_a, reply buffer $D_i, i \in \{1, ..., I\}$, noise schedule $\alpha_{0:K}$ and $\beta_{0:K}$

Output: ϵ_θ, $f_{inv,\psi}$, $\theta_e, \theta_d, \theta_q$

Initialization: θ, ψ, θ_e, θ_d, and θ_q

Separate the state-action trajectories of $D_i, i \in \{1, ..., I\}$ into state-action sequences with length T_e and calculate the discounted returns $R_t^i = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}$ for each step t

for *each task* i **do**
- // **Quantized Spaces Alignment (QSA) Pretraining**
- **for** *each train epoch* **do**
  - **for** *each train step* **do**
    - Sample states and actions from task i's buffer D_i
    - Calculate the quantization loss and reconstruction loss
    - Updating the parameters θ_e of $f^i_{VQE}(\cdot; \theta_e)$, θ_d of $f^i_{VQD}(\cdot; \theta_d)$, and θ_q of $f^i_{\theta_q}(\cdot)$ by solving the problem of Equation 4
  - **end**
- **end**
- Save the task i's well-trained $f^i_{VQE}(\cdot; \theta_e)$, $f^i_{VQD}(\cdot; \theta_d)$, and $f^i_{\theta_q}(\cdot)$
- // **Selective Weights Activation (SWA) Diffuser Training**
- Generate the task-related mask M_i for task i
- **for** *each train epoch* **do**
  - **for** *each train step* m **do**
    - Sample b sequences $\tau_i^0 = \{s^i_{t:t+T_e}, a^i_{t:t+T_e}, R^i_{t:t+T_e}\} \in \mathbb{R}^{T_e \times (d_s + d_a + 1)}$ from task i's buffer D_i
    - Obtain the quantized state and action feature $s^i_{z_q} = f^i_{s,\theta_q}(f^i_{VQE_s}(s^i; \theta_e))$ and $a^i_{z_q} = f^i_{a,\theta_q}(f^i_{VQE_a}(a^i; \theta_e))$ with the QSA module
    - Train the inverse dynamic model f_{inv} according to Equation 3
    - Formulate $s^i_{z_q}, a^i_{z_q}$ as sequences $\tau^{i,0}_{z_q} = \{s^i_{z_q, t:t+T_e}, a^i_{z_q, t:t+T_e}\}$
    - Sample the corresponding discounted returns $R^i_{t:t+T_e}$ from task i's buffer D_i
    - Sample diffusion time step $k \sim \text{Uniform}(K)$ and return coefficient $b \sim \mathcal{B}(\lambda)$
    - Sample Gaussian noise $\epsilon \sim \mathcal{N}(0, \boldsymbol{I}), \epsilon \in \mathbb{R}^{b \times T_e \times (d_{s_{z_q}} + d_{a_{z_q}})}$
    - Obtain $\tau^{i,k}_{s,z_q} = \sqrt{\bar{\alpha}_k} \tau^{i,0}_{s,z_q} + \sqrt{1 - \bar{\alpha}_k} \epsilon$
    - Perform the forward propagation with Equation 5
    - Train ϵ_θ according to Equation 1
  - **end**
- **end**
- Save task i's related models as $\epsilon_{i*\Omega,\theta}$

end

// **Weights Assembling**

Construct new models $\tilde{\epsilon}_\theta$ with the same structure as ϵ_θ

for *each task* i **do**
- Extract the task-related parameters W_i with mask information M_i from $\epsilon_{i*\Omega,\theta}$
- Fill the corresponding task-related parameters $W_i = M_i \circ W_i$ into $\tilde{\epsilon}_\theta$

end

Table 3: The hyperparameters of VQ-CD.

	Hyperparameter	Value
QSA section	network backbone	MLP
	hidden dimension of QSA module	256
	commitment cost coefficient	0.25
	codebook embedding limit ρ	3.0
	state codebook size per task	512
	number of state latent	10
	state latent dimension	2
	action codebook size per task	512
	number of action latent	5
	action latent dimension	2
	alignment type	VQ/AE/VAE
	VQ learning rate	[1e-4,1e-3]
SWA section	network backbone	Unet/MLP
	hidden dimension	256
	sequence length T_e	8
	diffusion learning rate	3e-4
	guidance value	0.95
	mask rate	$1/I$
	condition dropout λ	0.25
	max diffusion step K	200
	sampling speed-up stride	20
	condition guidance ω	1.2
	sampling type of diffusion	DDIM
Training	loss function	MSE
	batch size	32
	optimizer	Adam
	discount factor γ	0.99

The training of VQ-CD (Pseudocode is shown in Algorithm 2) contains three stages. 1) We first pre-train the QSA module for space alignment, as shown in lines 4-12, where we mainly want to solve the constrained problem of Equation 4. 2) Then, in lines 13-29, for each task i, we generate the task-related mask M_i followed by a standard diffusion model training process (Refer to Equation 1 and Equation 5 for the training loss) on the aligned state and action spaces. 3) Finally, we assemble the task-related weights W_i together with the mask information $\{M_i | i \in [1:I]\}$ according to $W = \sum M_i \circ W[i * \Omega]$, where Ω is the training steps for each CL task, and $W[i * \Omega]$ is the weights checkpoints of $\epsilon_{i*\Omega,\theta}$. It is noted that the pre-training of the QSA module and the training of the SWA module can be merged together, i.e., for each task i, we can first train the QSA module related to task i and then train the SWA module. The source code is available at here.

A.2 Hyperparameters

We classify the hyperparameters into three categories: QSA module-related, SWA module-related, and training-related hyperparameters. We use the learning rate schedule when pre-training the QSA module, so the VQ learning rate decreases from 1e-3 to 1e-4. In our experiments, the maximum diffusion steps are set as 200, and the default structure is Unet. Usually, it is time-consuming for the diffusion-based model to generate actions in RL. Thus, we consider the speed-up technique DDIM (Song et al., 2020) and realize it in our method to improve the generation efficiency during evaluation. For all models, we use the Adam (Kingma & Ba, 2014) optimizer to perform parameter updating.

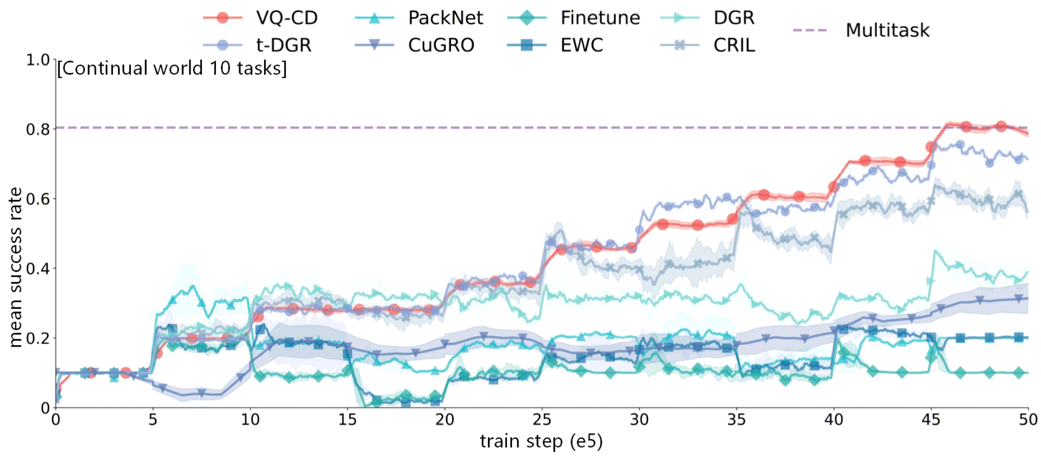

Figure 7: The experiments on the CW10 tasks, which contain various robotics control tasks. We train each method on each task for 5e5 steps and use the mean success rate on all tasks as the performance metric. Generally, we can see the superiority of our method from the above figure.

Table 4: The comparison of generation speed with different generation steps under the CL setting of Ant-dir task-4-18-26-34-42-49. In the main body of our manuscript, we use the 10 diffusion steps setting for all experiments.

Diffusion steps	200 (original)	100	50	25	20	10
sampling speed-up stride	1 (original)	2	4	8	10	20
Time consumption of per generation (s)	5.73±0.29	2.88±0.21	1.41±0.16	0.71±0.18	0.58±0.17	0.29±0.15
Speed-up ratio	1×	1.99×	4.06×	8.07×	9.88×	19.76×

Table 5: The GPU memory consumption.

domain	CL task setting	GPU memory consumption (GB)
D4RL	[Hopper-fr,Walker2d-fr,Halfcheetah-fr]	4.583
	[Hopper-mr,Walker2d-mr,Halfcheetah-mr]	4.583
	[Hopper-m,Walker2d-m,Halfcheetah-m]	4.583
	[Hopper-me,Walker2d-me,Halfcheetah-me]	4.583
Ant-dir	task-10-15-19-25	4.711
	task-4-18-26-34-42-49	5.955
CW	CW10	5.897

A.3 Computation

We conduct the experiments on NVIDIA GeForce RTX 3090 GPUs and NVIDIA A10 GPUs, and the CPU type is Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz. Each run of the experiments spanned about 24-72 hours, depending on the algorithm and the length of task sequences.

A.4 Generation Speed-up Technique

The time and memory consumption of diffusion models is attributed to the mechanism of diffusion generation process that requires multiple computation rounds to generate data Ho et al. (2020). Fortunately, previous studies provide useful speed-up strategies to accelerate the generation process (Nichol & Dhariwal, 2021; Song et al., 2020). In this paper, we adopt DDIM as the default

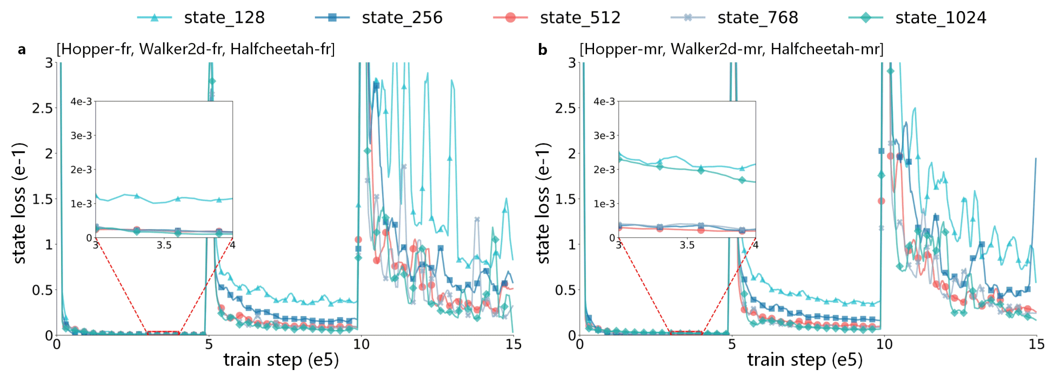

Figure 8: The QSA module loss under different codebook sizes about states. We explore five codebook size settings: 128, 256, 512, 768, and 1024. The red line represents the experimental codebook size setting for states.

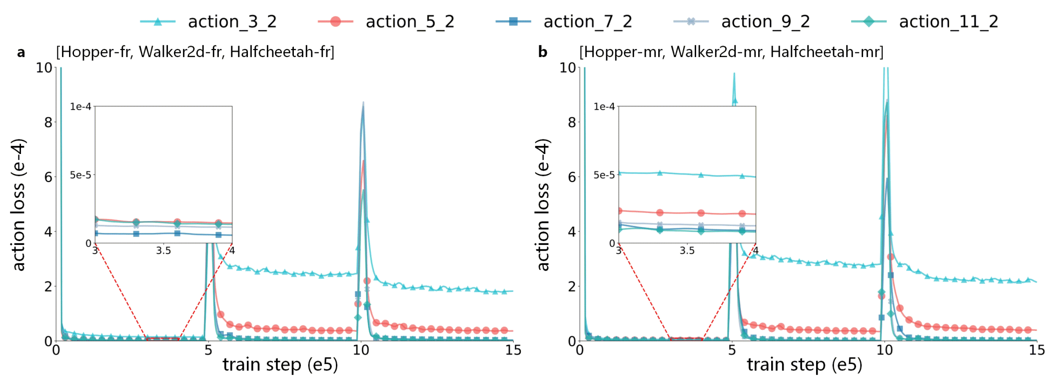

Figure 9: The QSA module loss under different latent numbers about actions. The setting includes 3, 5, 7, 9, and 11, which correspond to the aligned action space sizes 6, 10, 14, 18, and 22. The red line represents the experimental latent numbers setting for actions.

generation speed-up technique and reduce the reverse diffusion generation step to 10 compared to the original 200 generation steps. In Table 4, we use the CL setting of Ant-dir task-4-18-26-34-42-49 as an example to compare the time consumption of different generation steps. Compared with the original 200 diffusion steps, we can see that incorporating DDIM will significantly (**19.76×**) improve the efficiency of generation. In the experiments, we find that 10 diffusion steps setting performs well on performance and generation efficiency. Thus, we set the default sampling speed-up stride to 20, and the diffusion step is 200/20=10 steps.

A.5 Memory Consumption

In Table 5, we report the GPU memory consumption during the training process. We mainly consider the experiments on the D4RL, Ant-dir, and CW CL tasks. We can change the first block of the diffusion model to make our model suitable for a longer CL task sequence. For example, we expand the dimension length from 512 to 1024 when switching the CL training task from 'task-10-15-19-25' to 'task-4-18-26-34-42-49'.

A.6 Baselines Implementation

All the comparison methods used in this paper utilize their official codebases. Specifically,

- For L2M, we use the official source code: https://github.com/ml-jku/L2M

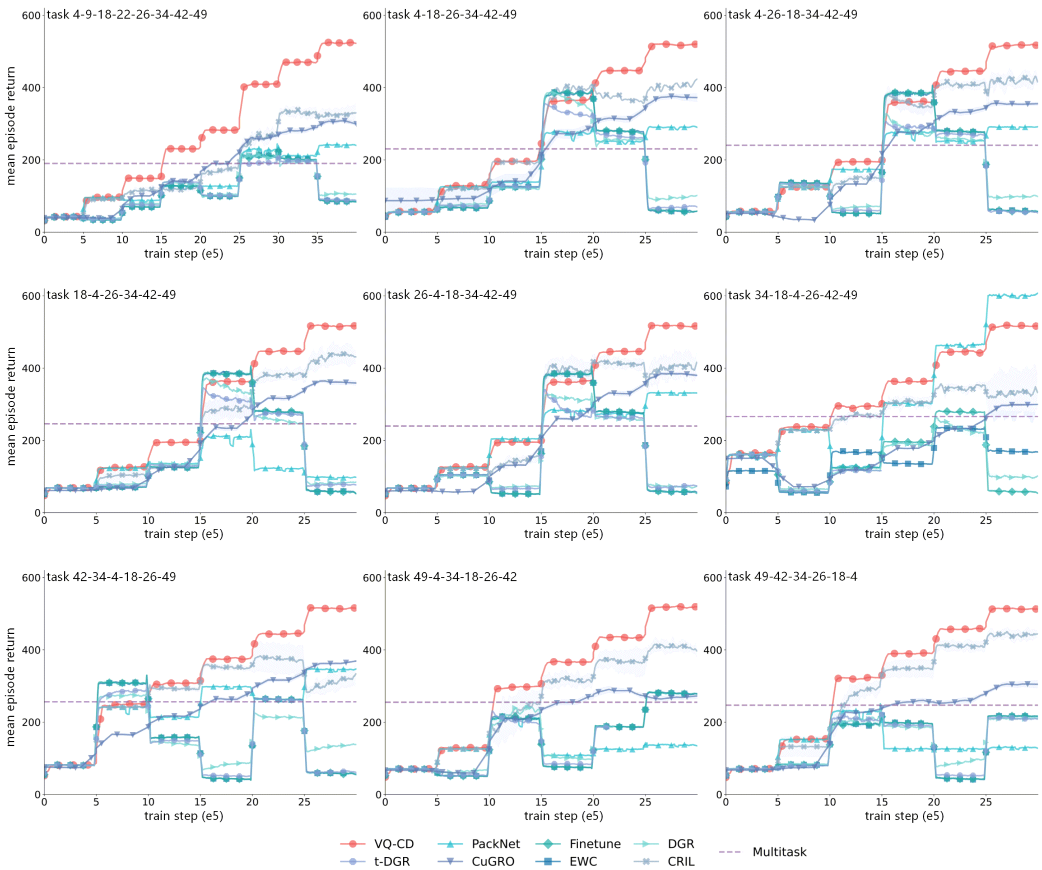

Figure 10: The experiments of Ant-dir with shuffled task order. We investigate the influence of shuffled task order in the Ant-dir environment, where the experiments include inserting new tasks into the predefined task order '4-18-26-34-42-49' and disrupting the tasks order.

- For CuGRO, we use the official source code: https://github.com/NJU-RL/CuGRO
- For CoD, we use the official source code: https://github.com/JF-Hu/Continual_Diffuser
- For MTDIFF, we use the official source code: https://openreview.net/forum?id=fAdMly4ki5

A.7 Network Details

In the diffusion model (SWA module), we utilize a UNet network structure, incorporating residual connections at both the input and output of each block. Additionally, residual connections are applied between the down-sampling and up-sampling blocks, meaning that the output of the down-sampling block serves as the input to the up-sampling block. The convolution kernels in the UNet are one-dimensional, with their shapes corresponding to the shape of the mask matrix.

In the QSA module, there are no shared parameters. The primary purpose of the QSA module is to align the state and action spaces across different environments. Consequently, for different tasks, the internal components of the QSA module, vector quantized encoder (VQE), vector quantized decoder (VQD), and codebook are task-specific, and none of their parameters are shared. Thanks to the alignment provided by the QSA module, the inverse dynamics model in the SWA module can be shared. This is because the state and action spaces of different environments are mapped into an alignment space with the same value range.

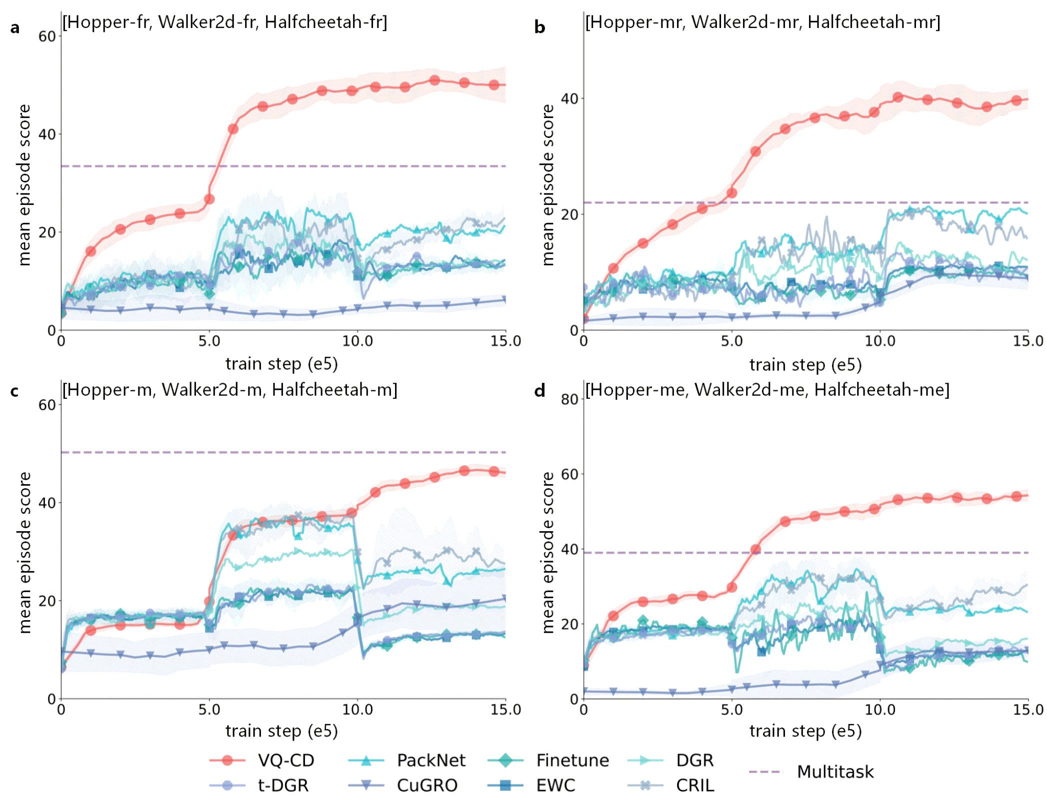

Figure 11: The comparison on the arbitrary CL settings. We select the D4RL tasks to formulate the CL task sequence. In order to align the state and action spaces, we use the pre-trained QSA module (the same as our method) to provide aligned spaces during training. The experiments are conducted on various dataset qualities, where the results show that our method surpasses the baselines not only at the expert datasets but also at the non-expert datasets, which illustrates the wide task applicability of our method. The datasets characteristic "fr", "mr", "m", and "me" represent "full-replay", "medium-replay", "medium", and "medium-expert", respectively. "Hopper", "Walker2d", and "Halfcheetah" are the different environments.

B Additional Experiments

B.1 QSA Module Loss Analysis

Under the same hyperparameter settings in Section 5.6, we report the loss of the QSA Module to investigate the effects of codebook size and latent number. For the states, we investigate the influence of codebook size, where we set codebook size as 128, 256, 512, 768, 1024, and select D4RL CL setting [Hopper-fr, Walker2d-fr, Halfcheetah-fr] and [Hopper-mr, Walker2d-mr, Halfcheetah-mr] as the example. The results are shown in Figure 8, where we train the QSA module on each task for 5e5 steps. We can see that for states, a codebook size of 512 is good enough for aligning the different tasks' state spaces. A larger codebook size, such as 768 and 1024 in Figure 8 **a** and **b**, will not bring significant loss improvements. Smaller codebook sizes can not provide sufficient latent vectors to map the state spaces to a uniform space.

For the action, we select the latent number to explore the QSA action loss and report the results in Figure 9. We can see the same trend that has been seen in QSA state loss (Figure 8). Though the lower loss value of the more latent number indicates that we should use more action latent vectors, we find that the gap between action latent number settings 5 and 7 is small when we increase computation resources. Besides, we also see inconspicuous performance gains in the final performance in Figure 6, which urges us to use 5 as the default action latent number setting. For the action latent vector dimension, we directly use 2 as the default setting.

Table 6: The comparison of time consumption per update between sparse and dense (normal) optimizers. We compare these two types of optimizers on the CL settings and find that when we first use the normal optimizer, such as Adam, to train the model and then use weights assembling to obtain the final model, the total physical time consumption is significantly smaller than sparse optimizer (e.g., sparse Adam).

domain	CL task setting	time consumption per update (s)	
		dense optimizer	sparse optimizer
D4RL	[Hopper-fr,Walker2d-fr,Halfcheetah-fr]	0.089±0.219	0.198±0.224
	[Hopper-mr,Walker2d-mr,Halfcheetah-mr]	0.096±0.223	0.197±0.223
	[Hopper-m,Walker2d-m,Halfcheetah-m]	0.089±0.211	0.195±0.224
	[Hopper-me,Walker2d-me,Halfcheetah-me]	0.090±0.223	0.206±0.225
Ant-dir	task-10-15-19-25	0.062±0.064	0.239±0.282
	task-4-18-26-34-42-49	0.064±0.061	0.214±0.270
CW	CW10	0.061±0.065	0.218±0.286

B.2 Experiments of Task Order Shuffling

To investigate the influence of task order in CORL, we choose Ant-dir as the testbed and change the task order for new CL training. We change the task order by inserting new tasks into the predefined task order '4-18-26-34-42-49' and disrupting the task order. We can see from the results shown in Figure 10 that our method achieves the best performance in almost all CL task orders. The task order will affect the final performance of other baselines. For example, CRIL performs better in the task orders 'task 18-4-26-34-42-49' and 'task 49-42-34-26-18-4' than in other task order experiments. Another example is PackNet, which achieves the best performance only in the task order 'task 34-18-4-26-42-49'. Different from the baselines, whose performance fluctuates with the changing of task orders, our method (VQ-CD) shows stable training performance no matter what task orders are defined.

B.3 Experiments of Baselines Equipped QSA

In Section 5.4, we report the comparison of our method and baselines in the arbitrary CL settings, where in the D4RL CL settings, we adopt state and action padding to align the state and action spaces. Apart from the state and action padding, we can also use the pre-trained QSA module to align the different state and action spaces. In Figure 11, we report the results of baselines equipped with QSA. When introducing the QSA, the model is actually trained on the feature space rather than the original state and action spaces, which makes it hard to learn for these baselines proposed from the traditional CL setting. From the results, we can also see that our method still achieves the best performance compared with these baselines. Considering the results of Figure 4 (VQ-MLPCD) and Figure 11 (VQ baselines), we can see the importance of complementary sections: QSA and SWA.

B.4 Supporting Tasks Training Beyond the Pre-defined Task Sequence

After training on pre-defined task sequences, we may hope the model has the capacity to support training on potential tasks, which means that we need more weights or weight masks. Releasing weight masks that are used to learn previous tasks is a straightforward choice when the total weights are fixed. We conduct the experiments of mask pruning on Ant-dir 'task 4-18-26-34-42-49' and report the performance and weight mask prune rate when pruning weight masks according to certain absolute value thresholds in Figure 12. The results illustrate that we can indeed release some weight masks under the constraint of preserving 90% or more performance compared with the unpruned model. On the other hand, we can also see that this mask pruning method can only provide finite capacity for tasks beyond the pre-defined task sequence. We postpone the systematic investigation of mask pruning to future works.

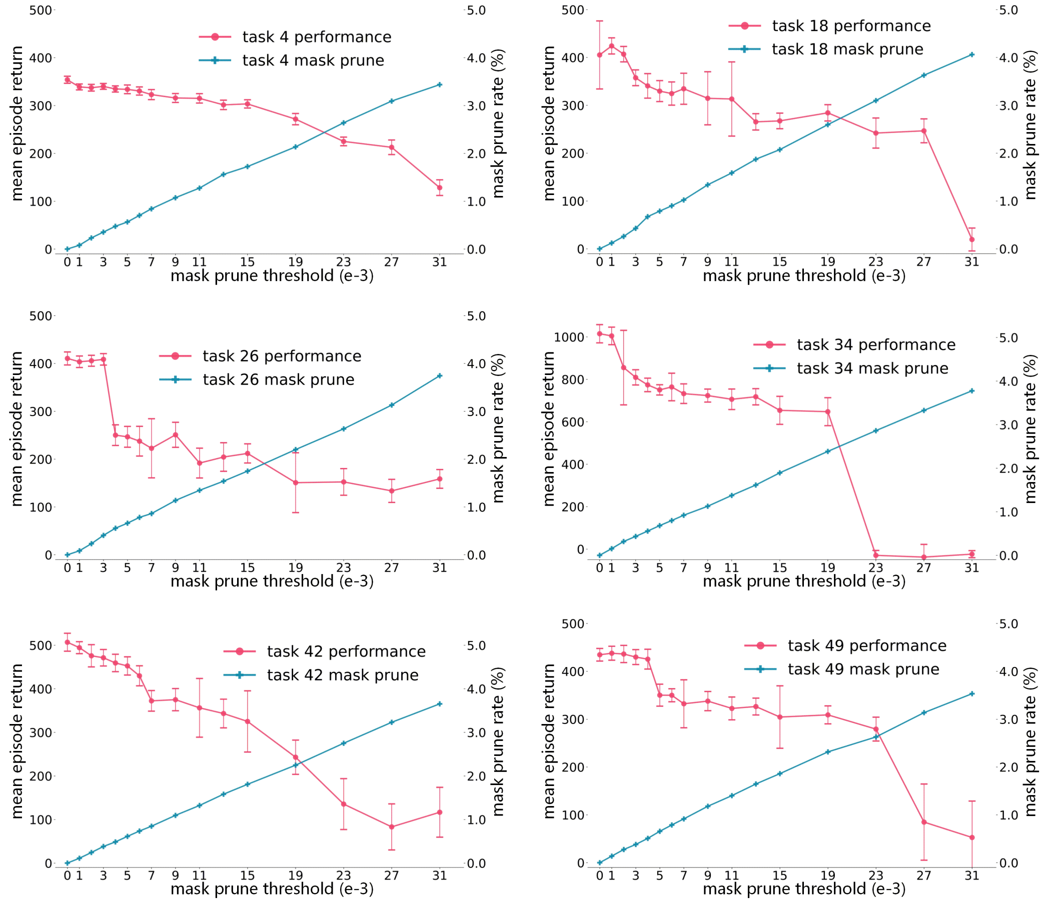

Figure 12: The mask pruning experiments of Ant-dir 'task 4-18-26-34-42-49'. We investigate the task pruning according to the absolute weight values, i.e., we release the weights to train on potential new tasks according to the mask prune threshold.

B.5 Time Consumption of Different Optimizers

In the CL settings of our experiments, we compare two types of optimizers and find that when we first use the normal optimizer, such as Adam, to train the model and then use weights assembling to obtain the final model, the total physical time consumption is significantly smaller than sparse optimizer (e.g., sparse Adam). Thus, we propose the weights assembling to obtain the final well-trained model after the training rather than suffering huge time burden of sparse optimizer during the training.

B.6 Mask Visualization

We select [Hopper-m, Walker2d-m, Halfcheetah-m] to visualize the weights mask of our method in Figure 13. To make it easy to show the mapping relation between masks and the weights, we draw the network structure and mask matrices, where we only report the first 100 channels of the mask matrices.

B.7 Alignment Space Visualization

In order to further demonstrate the effectiveness of our method. We conduct the visualization experiments of aligned state feature and report the visualization results in Figure 14. From the experimental results, we can see that the state features learned by the AE method are not well mapped to separate regions but are instead mapped to multiple areas. In contrast, the features obtained by our method

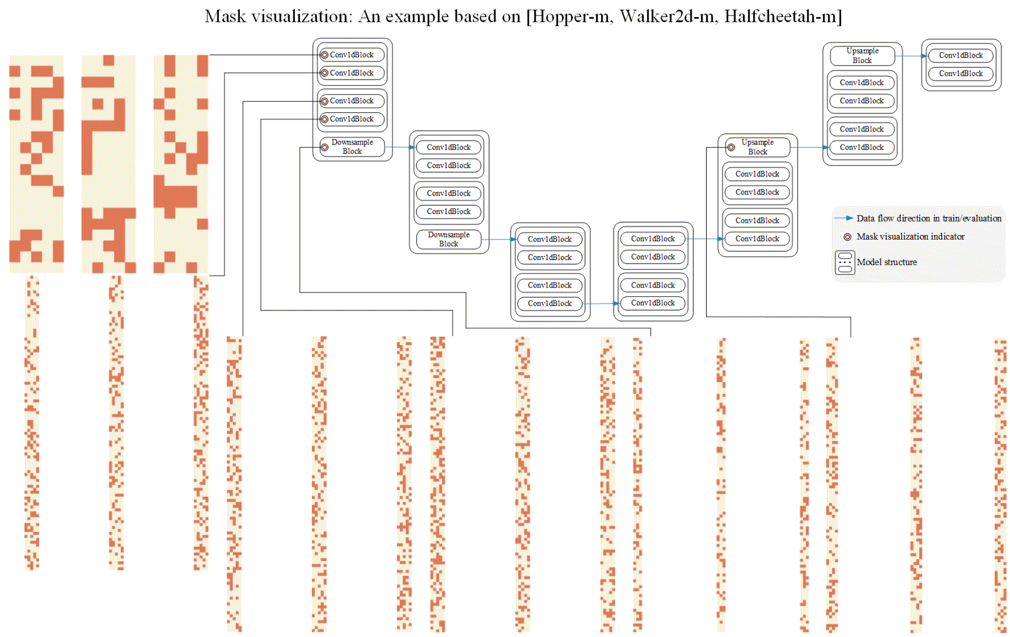

Figure 13: Mask matrices visualization. we select [Hopper-m, Walker2d-m, Halfcheetah-m] as an example to report the mask results. For each mask matrix, we only draw the first 100 channels of the weights mask matrix if the mask matrix is too large.

are better partitioned into individual regions, which is more conducive for the model to capture the data distribution.

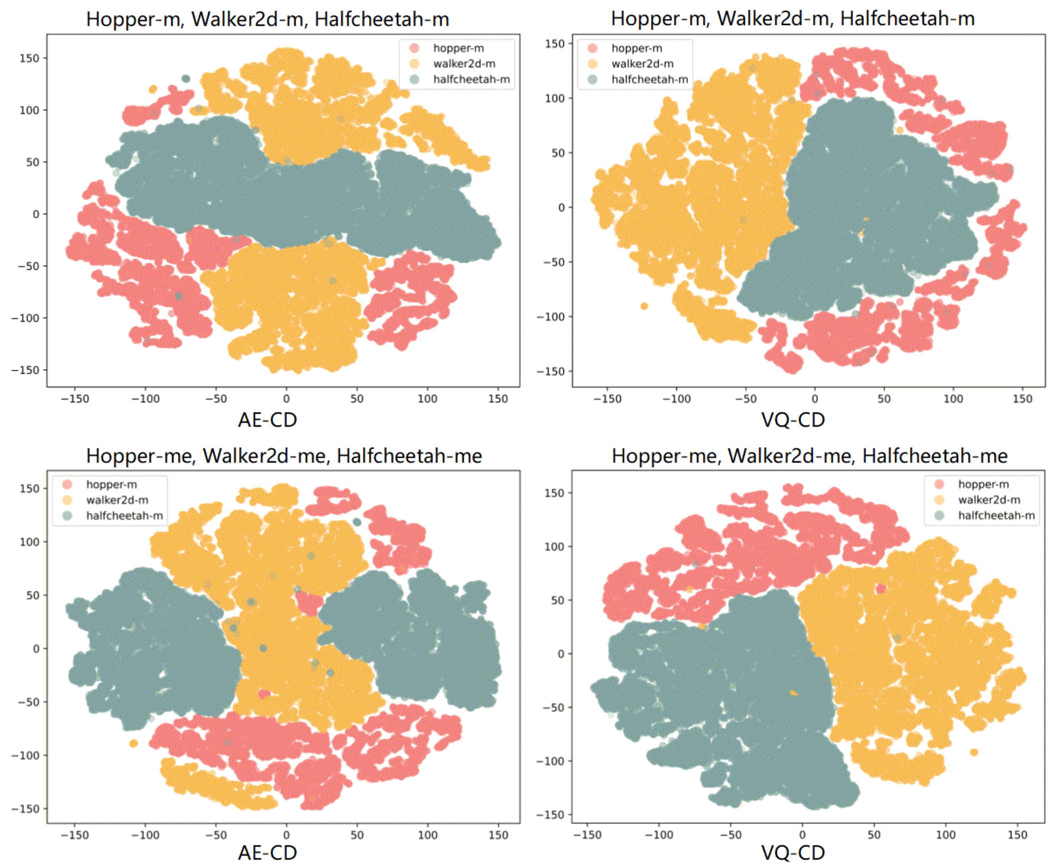

Figure 14: Visualization of aligned state feature. We use the QSA module to process the different state spaces and align them in the same space. Then we use t-SNE (Van der Maaten & Hinton, 2008) to visualize aligned state features.

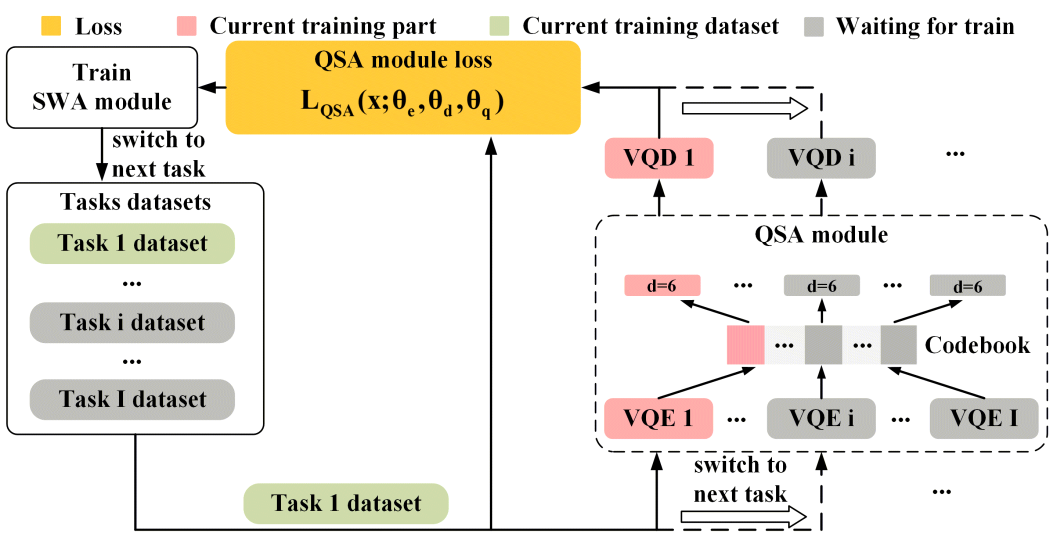

Figure 15: A graphical depiction of QSA training. From the figure, it is intuitively clear that the training of the QSA module can fully adhere to the CL training setup.