IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Transferring Policy of Offline Reinforcement Learning from Hybrid Dataset to Real World via Progressive Neural Network

Pengyu Zhao[1], Zheng Fang[1], Tongxu Ai[1], Eric Nichols[2], Randy Gomez[2], Bo He[1], Guangliang Li[1]

***Abstract*—Offline reinforcement learning (Offline RL) provides a compelling solution for applying RL in high-risk or resource-constrained real-world domains such as healthcare, autonomous driving, and robotic manipulation, where online exploration can be unsafe or impractical. However, Offline RL faces critical challenges arising from limited data coverage and potential distributional mismatch between the pre-training dataset and real-world environment. In this paper, we propose to allow an agent to learn from a hybrid dataset: high-quality real-world data and high-diversity simulation data, and assume that the dynamics of the simulation and the real world do not match, but the state space is the same. To address the policy extrapolation error and potentially catastrophic failures because of out-of-distribution actions and sim-to-real gap, we use progressive neural networks (PNNs) to transfer the offline policy to the real world. Results in two robotic manipulation tasks with a six-degree-of-freedom Ned robotic arm show that, the hybrid dataset facilitates faster offline learning and better adaptation to real-world tasks during online learning. In addition, further analysis shows that transferring the offline policy via PNN can not only effectively retain the policy learned from the hybrid dataset and bridge the gap between simulation and reality data, but also allow the agent to explore in a more diverse distribution of samples during online learning.**

***Index Terms*—Offline Reinforcement Learning; Transfer Learning; Progressive Neural Network**

I. INTRODUCTION

DEEP reinforcement learning (DRL) has shown remarkable potential in a wide range of tasks, from video games to complex robotic control [1], [2], [3]. However, directly applying DRL to real-world robot control poses significant challenges due to the long time and high cost of collecting samples to learn a good policy, as well as the safety risks associated with physical exploration. Therefore, DRL has been mostly limited to simulated domains or benchmark environments in which tasks are well-defined and data collection is safe and inexpensive.

To address the above issues, offline RL was proposed to enable policy training from static, pre-collected datasets, eliminating the need for real-time environmental interaction [4]. This paradigm is particularly promising in high-risk or resource-constrained domains such as healthcare, autonomous driving, and robotic manipulation, where online exploration can be unsafe or impractical. However, collecting data in real environments for offline learning is expensive and risky [5], and often results in datasets with limited behavioral diversity. Therefore, despite the advantages in safety and efficiency, offline RL faces critical challenges due to limited data diversity [6], [7] and distributional coverage [5], [7], a key limitation caused by mismatch between offline pre-training and real-world deployment. Although offline pre-training followed by real-world fine-tuning has been proposed to mitigate these issues [8], [9], this distributional shift between pre-training dataset and real-world environment increases the risk of "policy shift" where the offline policy selects actions outside the behavior policy's support, leading to inaccurate value estimation and degraded performance after directly transferring to the real world for fine-tuning.

On the other hand, simulation environments provide cost-effective, safe, and controllable platforms for data generation with high diversity, facilitating robust policy learning without real-world risks [10]. However, transferring policies from simulation to the real world is non-trivial due to differences in observation and dynamics – such as sensor noise, actuator inaccuracies and environmental variation – commonly referred to as "reality gap" [11], [10]. To bridge this gap, many sim-to-real transfer methods have been proposed, including domain adaptation [12], inverse dynamics models [13], domain randomization [14], [15], and progressive neural networks (PNNs) [16]. Domain randomization relies on continuously perturbing environment parameters, domain adaptation requires large amounts of real data for distribution alignment or adversarial training, which is impractical for our offline learning setting. In contrast, PNNs have shown strong potential for forward transfer across tasks by dynamically expanding network structures without requiring explicit alignment between source and target domains. By preserving previously acquired knowledge and isolating learning for new tasks, PNNs offer a powerful mechanism to adapt to new and even different environments [17], which is important for accommodating dissimilar inputs between simulations and real-world sensors in sim-to-real policy transfer.

Since the success of offline learning heavily depends on the quality and diversity of the dataset: high-quality datasets containing near-optimal behavior can support effective policy learning and high-diversity datasets can improve the generalization of the learned policy, in this paper, we propose to allow an agent to learn from a hybrid dataset: high-quality real-world data and high-diversity simulation data. However,

Manuscript received: July 15, 2025; Revised: October 19, 2025; Accepted: December 6, 2025. This paper was recommended for publication by Editor Aleksandra Faust upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by Young Taishan Scholars Program (Grant No. tsqn202408072) and Qingdao Natural Science Foundation (Grant No. 23-2-1-153-zyyd-jch). (Pengyu Zhao and Zheng Fang contributed equally to this work.) (Corresponding author: Guangliang Li.)

[1] Pengyu Zhao, Zheng Fang, Tongxu Ai, Bo He and Guangliang Li are with the College of Electronic Engineering, Ocean University of China, Qingdao 266100, China (e-mail: guangliangli@ouc.edu.cn). [2] Eric Nichols and Randy Gomez are with the Honda Research Institute Japan Co., Ltd, Wako, Japan (e-mail: {e.nichols,r.gomez}@jp.honda-ri.com).

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

there is usually a gap between simulation and reality, and we assume that the dynamics of the simulation and the real world do not match, but the state space is the same. To address the policy extrapolation error and potentially catastrophic failures because of out-of-distribution (OOD) actions and sim-to-real gap, we propose to use PNNs to transfer the learned offline policy to the real world. We hypothesize and expect that offline policy transfer via PNNs can achieve more efficient and faster learning in the real-world task.

We evaluated and tested our proposed method in two robotic manipulation tasks: Reach and Push, both with a six-degree-of-freedom Ned robotic arm. Our main contributions and findings are as follows:

- The proposed hybrid dataset consisting of high-diversity simulation data and high-quality real-world demo data allows for faster offline policy learning;
- Our proposed approach can mitigate the distributional shift observed in the offline-to-online learning setup;
- Using PNN can effectively mitigate the sim-to-real gap and allow the agent to explore a more diverse distribution of samples during further online learning, which results in the faster online learning speed and higher performance in the real-world task;
- Our method benefits from simulation data collected during the early exploratory stage of agent training in simulation rather than the late stage one.

II. RELATED WORK

Offline RL provides a compelling solution for applying DRL in high-risk or resource-constrained domains where online exploration can be unsafe or impractical. Assuming there is no further interaction with the environment and to mitigate the extrapolation errors resulting from the distributional shift, policy constraint mechanisms have been introduced to improve training stability and policy generalization. For example, Fujimoto et al. [18] proposed Batch-Constrained Q-Learning (BCQ) using a variational autoencoder (VAE)-based generative model to ensure that the policy-generated actions remain close to the behavior policy, thus avoiding unreliable extrapolation. Kumar et al. [19] further proposed bootstrapping error accumulation reduction (BEAR) using Maximum Mean Discrepancy (MMD) to constrain the policy's action distribution to be consistent with the behavior policy. Conservative Q-Learning (CQL) [20] improves stability by maximizing Q-values on training data actions while minimizing Q-values on out-of-distribution (OOD) actions. Building upon CQL, Cal-QL [21] introduces a calibration mechanism that prevents the learned Q-values from falling below those of a reference policy, avoiding early-stage performance drops during online fine-tuning. Similarly, Implicit Q-Learning (IQL) [22] skips explicit policy updates and instead directly extracts the policy through advantage-weighted regression. Qiao et al. [23] proposed a Search-based Uncertainty estimation method for Model-based Offline RL (SUMO) that measures cross-entropy between synthetic and real samples. Q-value Regularized Transformer (QT) [24] addresses suboptimal trajectory stitching in conditional sequence modeling by integrating Q-value regularization into Transformers, balancing behavior cloning and policy improvement to select high-return actions. Ada et al. [25] proposed State Reconstruction for Diffusion Policies (SRDP) that incorporates state reconstruction feature learning in diffusion policies to address the problem of out-of-distribution (OOD) generalization.

Hybrid offline-to-online RL approaches aim to leverage large amounts of pre-collected data while allowing for additional online interaction to fine-tune offline policy. For instance, Song et al. [26] proposed a preferential experience replay mechanism to synergistically use offline data and online interaction data in training to improve the sample efficiency and policy stability. Advantage-Weighted Actor-Critic (AWAC) [8] combines sample efficient dynamic programming with maximum likelihood policy updates, enabling rapid learning of skills with a combination of prior offline demonstration data and online real-world robotic experience. Niu et al. [27] proposed a dynamics-aware hybrid offline-and-online RL framework, which combines learning from limited real data in offline RL and unrestricted exploration through imperfect simulators in online RL. Xie et al. [28] proposed policy finetuning (PF) which can simultaneously maximizes expected return and constrains policy deviation from the offline behavior policy through a regularization term to ensure a smooth transition from offline pretraining to online finetuning. To facilitate a smooth transfer of the offline policy to real-world deployment, Mao et al. [29] proposed a model-based hybrid offline-to-online reinforcement learning method – MOORe – which learns a transition model from offline data and then refines it with limited online samples. Zhang et al. [30] proposed noise-perturbed Q-value updates for better Q-value estimation during offline pre-training and increased update frequency for faster online fine-tuning. Chaudhary et al. [31] proposed Meta Offline-Online Reinforcement Learning (MOORL) that uses a meta-policy to seamlessly adapt across offline and online trajectories for robust initialization with offline data and efficient exploration via online interactions. Ren et al. [32] proposed hybrid inverse RL that learns on a mixture of online and expert data to improve reward learning through offline inference and online fine-tuning. RO2O [33] introduces a robust offline-to-online algorithm incorporating Q-ensembles and adversarial samples to ensure stable online adaptation without altering the learning objective. Similar to our work, Zhang et al. [34] presents a policy expansion strategy that retains the offline-learned policy while introducing a learnable policy for online exploration, enabling adaptive composition of behaviors.

Different from the above work which uses a mixture of offline and online data at the same time, high-quality offline data plus online data, or direct interactions with simulators, we propose to allow the agent learning from a hybrid dataset composed of random simulation data, expert simulation data and real-world demo data, and investigate how the diversity of offline data affects the offline learning and online fine-tuning. In addition, we employ PNNs to transfer the offline policy to real-world robotic tasks to bridge the reality gap and resolve the distribution shift problem arising during the offline-to-online learning process.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

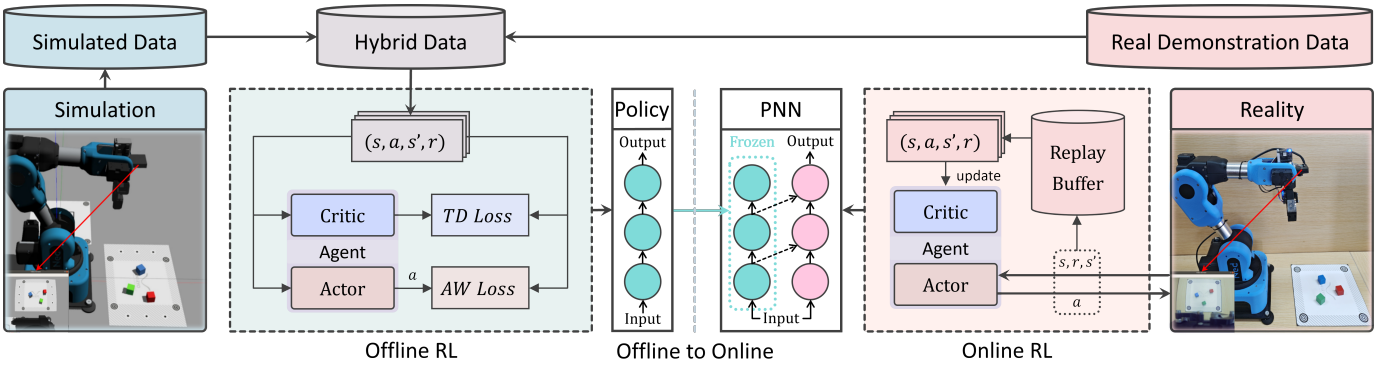


Fig. 1: An illustration of mechanism for our proposed hybrid offline-policy transfer via progressive neural network method.

III. BACKGROUND

A. Offline Reinforcement Learning

A reinforcement learning (RL) problem is often modeled as a Markov Decision Process (MDP), represented by the tuple $M = \{S, A, T, R, \gamma\}$. Here, S represents the state space and A is the action space. The transition probability distribution $T(s_{t+1} \mid s_t, a_t)$ describes how an agent transitions from one state to another by performing an action. The reward function $R(s_{t+1} \mid s_t, a_t)$ provides feedback to the agent for its actions, indicating progress towards achieving the task objective. The discount factor $\gamma \in (0, 1]$ determines the agent's preference for immediate versus future rewards. The goal of an RL agent is to discover an optimal policy π^* that maximizes the expected cumulative rewards.

Offline RL optimizes policies using pre-collected datasets without further environment interaction [4], [5], enabling safe and sample-efficient learning in high-stakes domains such as robotics and healthcare. However, limited data coverage and distributional mismatch pose challenges: real-world datasets are often biased and narrow, leading to inaccurate value estimates and out-of-distribution actions, which may result in catastrophic failures.

In the offline setting, the agent is given a fixed dataset $\mathcal{D} = \{(s, a, r, s')\}$ collected via an unknown or suboptimal behavior policy [4], [5]. The objective is to learn a policy $\pi(a|s)$ that maximizes the expected return solely from this static dataset. Typically, transitions (s, a, r, s') are sampled from $\mathcal{D}$, and the Q-function is updated by minimizing the Bellman error:

$$\mathcal{L}_Q = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q(s,a) - \left(r + \gamma\, \mathbb{E}_{a'\sim\pi}\left[Q(s',a')\right]\right)\right)^2\right]. \tag{1}$$

The success of the offline learning process heavily depends on the quality and diversity of the dataset. High-quality datasets containing near-optimal behavior can support effective policy learning, while low-quality or narrowly distributed datasets may hinder generalization, especially in unexplored high-value regions of the state-action space. A central challenge in offline RL is the policy extrapolation error, where the learned policy may select actions from underrepresented or unseen regions of the dataset. When these actions are inaccurately assigned high values, the resulting overestimation bias can significantly degrade policy performance.

B. Progressive Neural Network

PNNs allocate an independent neural network column for each task and achieve feature transfer through lateral connections, enabling continuous learning. The initial structure of a PNN consists of a single-column network with L layers, and is used to train the source task. The activation function of each layer is represented as $h_i \in \mathbb{R}^{n_i}$, where n_i is the number of neurons in the i-th layer.

When extended to the second column for the target (or new) task, the parameters of the first column, $\Theta^{(1)}$, are frozen, while the parameters of the new column, $\Theta^{(2)}$, are initialized either with the values of $\Theta^{(1)}$ or random values. At this point, the activation function $h_i^{(2)}$ of the i-th layer in the second column receives inputs from both the output of the $(i-1)$-th layer in the second column, $h_{i-1}^{(2)}$, and the output of the $(i-1)$-th layer in the first column, $h_{i-1}^{(1)}$.

This mechanism can be generalized to K columns, with the expression as follows:

$$h_i^{(k)} = f\left(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)}\right), \tag{2}$$

where $W_i^{(k)}$ is the weight matrix of the i-th layer in the k-th column, $U_i^{(k:j)}$ is the lateral connection parameter between the i-th layers of the k-th column and the j-th column, and $f(x) = \max(0, x)$ is the non-linear activation function used in the hidden layers.

IV. METHODOLOGY

This paper proposes an offline policy transfer method via progressive neural networks (PNNs) to achieve more efficient and faster learning in the real world, as shown in Fig. 1. Since the success of offline learning heavily depends on the quality and diversity of the dataset, the proposed method first allows an offline RL agent to learn from hybrid datasets: high-quality real-world data and high-diversity simulation data. However, there is usually a gap between simulation and reality, and we assume that the dynamics of simulation and real world do not match, but the state space is the same. To address the policy extrapolation error and potentially catastrophic failures because of out-of-distribution (OOD) actions and sim-to-real gap, then the learned offline policy will be transferred to the real-world task via PNNs for fine-tuning via online RL.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

A. Offline Policy Learning from Hybrid Dataset

In our approach, we first train an agent to learn an offline policy from hybrid datasets: high-quality real-world data and high-diversity simulation data. The high-quality real-world data is expert demonstrations by operating the physical robot in the real world. High-diversity simulation data can be generated by running a random or online learning policy in the simulation environment. Since the real and simulated datasets share the same observation and action space specifications, their differences stem solely from dynamic discrepancies between the respective environments. This compatibility enables the two sources of data to be directly merged for joint training without requiring additional preprocessing or alignment. The AWAC algorithm is used for offline reinforcement agent policy learning. AWAC combines the advantage function weight with policy update, which not only makes efficient use of offline data, but also maintains the flexibility of online training adjustments, making it easier to transfer to the real environment for continual learning [8].

Specifically, we first collect real-world data by running an expert policy π_β and simulation data by running a policy to form a hybrid dataset $\mathcal{D}_{hybrid}$. The agent then learns an offline policy via AWAC from the fixed hybrid dataset $\mathcal{D}_{hybrid}$. The AWAC agent adopts an actor-critic structure and alternates between updating the Q-function and the policy network. During each iteration, a mini-batch of transitions (s, a, r, s') is sampled from the offline dataset $\mathcal{D}_{hybrid}$. The Q-function is updated by minimizing the Bellman error using the target:

$$y = r(s,a) + \gamma \mathbb{E}_{a' \sim \pi_\theta(\cdot|s')} \left[Q_\phi(s', a') \right], \tag{3}$$

where y is the target value, $r(s,a)$ is the received reward for the state action pair (s,a), s' and a' are next state and action, and γ is the discount factor. The parameters ϕ of the Q-function $Q_\phi(s,a)$ are then updated by minimizing the mean squared error as:

$$\phi = \arg\min_\phi \mathbb{E}_\mathcal{D} \left[\left(Q_\phi(s,a) - y \right)^2 \right]. \tag{4}$$

Next, the policy network is updated based on a weighted maximum likelihood objective, where the weights are computed using the advantage function $A^\pi(s,a) = Q_\phi(s,a) - V_\phi(s)$. The parameters θ of the policy network π_θ are updated as:

$$\theta = \arg\max_\theta \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\log \pi_\theta(a|s) \cdot \exp\left(\frac{1}{\lambda} A^\pi(s,a) \right) \right], \tag{5}$$

where λ is a temperature parameter that controls the concentration of the weighting distribution. This objective function encourages the policy to assign higher probabilities to actions with higher estimated advantages, while constraining deviations from the behavior policy to ensure stable and conservative policy updates. This offline training cycle is repeated until the policy π_θ converges.

B. Transferring Offline Policy to Real World via PNN

After obtaining the offline policy, we use a two-column PNN to transfer the policy to the target task in the real world. We hypothesize that this can address the policy extrapolation error and potentially catastrophic failures because of out-of-distribution (OOD) actions and sim-to-real gap. The first column policy network π_{θ_1} and critic network Q_{ϕ_1} are initialized with the offline policy π_{θ_o} and critic Q_{ϕ_o}, and remains frozen throughout the online training process in the real world. The second column policy network π_{θ_r} and critic network Q_{ϕ_r} are randomly initialized and updated during the interaction with the real environment. The two columns are connected horizontally to achieve feature transfer, allowing the offline policy to adapt to the real environment while avoiding the catastrophic forgetting because of the online update.

In the real-world environment, the AWAC algorithm is used to update the second column policy network π_{θ_r}. Policy weighting mechanism in AWAC is used to weight the policy update by the currently estimated advantage function, so it can deal with the problem of the state-action distribution in the real environment being offset from the simulation environment to some extent. Specifically, at each time step t, the robot selects an action a through the second column policy network π_{θ_r} according to the current state s, transfers to the next state s' after interacting with the real environment and obtains an immediate reward r. Subsequently, the four-tuple $[s, a, r, s']$ is stored in the replay buffer $\mathcal{B}$ of the target task. The update of policy π_{θ_r} and value function Q_{ϕ_r} is based on the small batch interaction samples sampled from $\mathcal{B}$. The pseudocode of the proposed migration algorithm is summarized in Algorithm 1.

Algorithm 1 Hybrid Offline Policy Transfer via Progressive Neural Networks

Require: Sim-dataset $\mathcal{D}_{\text{sim}} = \{(s,a,s',r)\}$, Real-dataset $\mathcal{D}_{\text{real}} = \{(s,a,s',r)\}$, Hybrid dataset $\mathcal{D}_{\text{hybrid}}$, Replay buffer $\mathcal{B}$, Offline policy π_{θ_o} and critic Q_{ϕ_o}, Online policy π_{θ_r} and critic Q_{ϕ_r}

Ensure: Online policy π_{θ_r}

Initialize offline policy π_{θ_o} and critic Q_{ϕ_o}
Collect samples in simulation to construct $\mathcal{D}_{\text{sim}}$
Collect demonstration trajectories in the real environment to construct $\mathcal{D}_{\text{real}}$
for $i = 1, \ldots, N_{\text{offline}}$ **do**
  Sample $(s,a,s',r) \sim \mathcal{D}_{\text{hybrid}}$
  Compute y following Eq. (4)
  Update Q_{ϕ_o} following Eq. (5)
  Update offline policy π_{θ_o} following Eq. (6)
end for
Initialize and freeze the first column π_{θ_1} and Q_{ϕ_1} with π_{θ_o} and Q_{ϕ_o}
Randomly initialize the second column π_{θ_r} and Q_{ϕ_r}
for $j = 1, \ldots, N_{\text{online}}$ **do**
  Interact with the real environment to collect (s,a,r,s')
  Add to $\mathcal{B}$
  Sample $(s,a,s',r) \sim \mathcal{B}$
  Update Q_{ϕ_r} following Eq. (5)
  Update Online policy π_{θ_r} following Eq. (6)
end for

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

V. EXPERIMENTS

We evaluated our proposed method on two robotic manipulation tasks: *Reach* and *Push*, with 4-dimensional continuous action space, and 23- and 26-dimensional continuous state space, respectively. Both tasks use a six-degree-of-freedom Ned robotic arm. In the Reach task, the robot uses its camera to identify a target region within a designated white workspace and attempts to move the center of its gripper fingers to the target while maintaining a downward-facing orientation. In the Push task, the robot first locates a manipulable object and then pushes it toward a designated target region within the same workspace. The state in both tasks is represented with the end-effector coordinates, the distance to the target etc. The action space is composed of the angles of the four movable joints.

To increase and test the effect of dataset diversity for offline policy training, we first built a simulation environment with the six-axis Ned robot on Gazebo for both tasks, and collected simulation datasets by training a PPO agent in the simulation environment from the beginning until convergence. The real-world demonstration dataset was collected by running the optimal policy in the real-world task with the physical robot. Both tasks in simulation and real world share the same state space, but differ in system dynamics. In particular, both Reach and Push tasks in simulation differ from those real-world ones in terms of gravity, joint friction, joint damping coefficients, link mass, and range of joints. Additionally, the observations in real-world tasks may be affected by noises caused by factors such as lighting conditions and background variations.

In our experiments, to investigate the effect of dataset diversity on offline policy training, we constructed three simulation datasets for both Reach and Push tasks: one full dataset consisting of about 20,000 samples collected by training a PPO agent from beginning to convergence in the simulation, one early-stage dataset consisting of approximately 10,000 samples collected from the beginning until the midpoint of the PPO agent training, another late-stage dataset consisting of about 10,000 samples collected from the midpoint until convergence of the PPO agent training. The real-world demonstration dataset consisting of 200 expert demonstrations collected by running the optimal policy in each real-world task. The above three simulation datasets will be combined with the real-world demonstration dataset as three hybrid datasets for offline policy training, respectively. In addition, we performed ablation studies to investigate the effect of PNN and simulation dataset on policy learning with our method. Specifically, we trained six agents as below:

- **HyOFFPNN**: An offline policy is first trained via AWAC on a hybrid dataset composed of full simulation dataset and real-world demonstration dataset, and then transferred to the real world via PNN for further online learning via AWAC.
- **HyOFF**: Offline policy training is the same as HyOFFPNN, but then directly transferred to the real world for further online learning via AWAC.
- **OFF-Real**: An offline policy is trained with only real-world demonstration dataset, and then directly transferred to the real world for further online learning via AWAC.
- **HyOFFPNN-BS**: Same as HyOFFPNN, but only simulation data collected during the early training stage (bottom 50%) of a PPO agent and real-world demonstration data were used for the offline policy training.
- **HyOFFPNN-TS**: Same as HyOFFPNN, but only the simulation data collected during late-stage (top 50%) of a PPO agent and real-world demonstration data were used for the offline policy training.
- **Cal-QL [21]**: The Cal-QL agent is used as a baseline trained with the same hybrid dataset as HyOFFPNN for offline-to-online reinforcement learning.

We adopt the AWAC algorithm as our learning framework. Both actor and critic networks are represented as fully connected networks (FCNs). The actor network outputs the mean of a Gaussian policy for continuous action spaces, with a learnable standard deviation. The critic network outputs scalar Q-values for state-action pairs, where the final output is flattened using the `squeeze` operation. All hidden layers use $ReLU$ activation functions, and the output layers are linear. The network architectures for the two tasks are: FCN (actor) = [23, 256, 256, 4] and FCN (critic) = [27, 256, 256, 1] for Reach; FCN (actor) = [26, 256, 256, 4] and FCN (critic) = [30, 256, 256, 1] for Push. The learning rate for both the actor and critic networks in the two tasks is set to 0.001.

VI. RESULTS AND DISCUSSION

This section presents and analyzes the learning performances of six agents: HyOFFPNN, HyOFF, OFF-Real, HyOFFPNN-BS, HyOFFPNN-TS and Cal-QL, in terms of offline learning process from fixed dataset and further online learning via interacting with the real-world tasks. We examined the learning curves via measuring the accumulated rewards per episode in terms of the reward functions in both tasks. The bold line shows the mean performance over five independent trials and the shaded area shows the standard deviation.

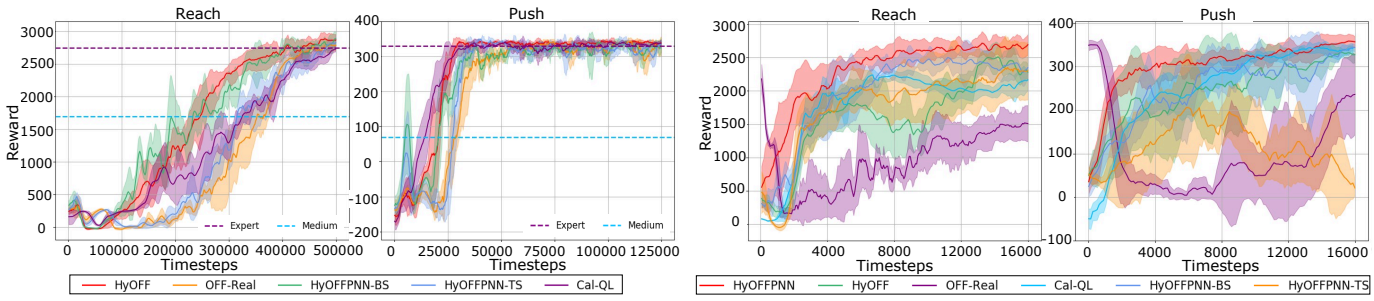
A. Offline Learning Performance

Fig. 2 (a) shows the offline learning curves of HyOFF, OFF-Real, HyOFFPNN-BS, HyOFFPNN-TS and Cal-QL agents for both Reach and Push tasks by testing the learned policies in the simulation environments. As illustrated in Fig. 2 (a), all six agents can finally obtain an optimal expert performance at which the real-world demo dataset was collected. However, our HyOFF agent learns from hybrid dataset consisting of full simulation dataset and real-world demo dataset much faster compared to the OFF-Real agent learning from only real-world demo dataset. This indicates that the additional diverse simulation data has a positive effect on offline agent learning. However, with the same hybrid dataset, the Cal-QL agent learns slower than our HyOFF agent learning via AWAC in the Reach task, but faster in the Push task.

B. Online Learning Performance and Sample Efficiency

Fig. 2 (b) shows the online learning curves of HyOFFPNN, OFF-Real, HyOFFPNN-BS, HyOFFPNN-TS and Cal-QL agents for both Reach and Push tasks after transferring

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



(a) Offline RL

(b) Online RL

Fig. 2: Learning curves of HyOFFPNN, HyOFF, OFF-Real, HyOFFPNN-BS, HyOFFPNN-TS and Cal-QL agents.

the learned offline policy to the real-world environments. As illustrated in Fig. 2 (b), while the performance of the OFF-Real agent learning from real-world demo dataset drops dramatically and obtains the worst performance of all agents after transferring to the real-world tasks, our HyOFFPNN agent learns from hybrid dataset consisting of full simulation dataset and real-world demo dataset at the fastest speed and achieves an optimal performance finally. That might be because the distribution of online training data for OFF-Real is becoming clearly different from the real demonstration dataset used for offline learning (Fig.3 (a)), while the distribution of online training data for our HyOFFPNN method still mostly falls within the diverse distribution of simulation dataset used for offline training (Fig.3 (b)). Moreover, the additional offline policy transfer via PNN facilitates HyOFFPNN to adapt to the real-world task better than Cal-QL baseline.

TABLE I: Number of samples required for HyOFFPNN, HyOFF, OFF-Real and Cal-QL agent to reach a satisfactory or near-optimal performance for the Reach task (2200 episode rewards) and the Push task (310 episode rewards).

Task	HyOFFPNN	HyOFF	OFF-Real	Cal-QL
Reach	5K	12.5K	30K	6.9K
Push	8K	11.9K	25K	9.8K

We also compare the sample efficiency of HyOFFPNN, HyOFF, OFF-Real and Cal-QL agents to reach a satisfactory or near-optimal performance in both Reach and Push tasks, as shown in TABLE I. Results in TABLE I shows that, with only real-world demo dataset, it takes the OFF-Real agent a large amount of samples to achieve a good performance in both tasks, while it generally takes the HyOFF agent a bit fewer than half amount of samples required by the OFF-Real agent to achieve the same performance. With PNN transferring the offline policy to the real world, our HyOFFPNN agent only requires about half the number of samples needed by the HyOFF agent (i.e., a quarter of those required by OFF-Real) to reach the same performance, even fewer than the state-of-the-art Cal-QL baseline.

In addition, we evaluate the success rate by testing learned final policies of our HyOFFPNN method, HyOFF, OFF-Real and Cal-QL for 50 trials in both tasks in the real world. Results in Table II show that our HyOFFPNN method can achieve 98% and 92% success rate for Reach and Push task,

TABLE II: Success rate of completing the real-world Reach task (2200 episode reward) and the Push task (310 episode reward).

Task	HyOFFPNN	HyOFF	OFF-Real	Cal-QL
Reach	98%	82%	68%	86%
Push	92%	76%	54%	80%

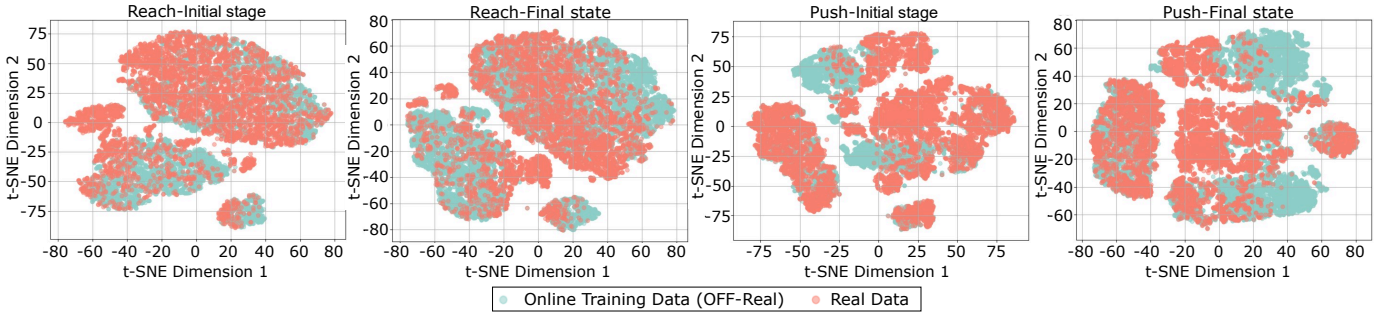
respectively, which are clearly better than other three methods. This indicates that our method is robust to real-world noise to some extent, such as sensor inaccuracies or environmental perturbations.

C. Ablation Study

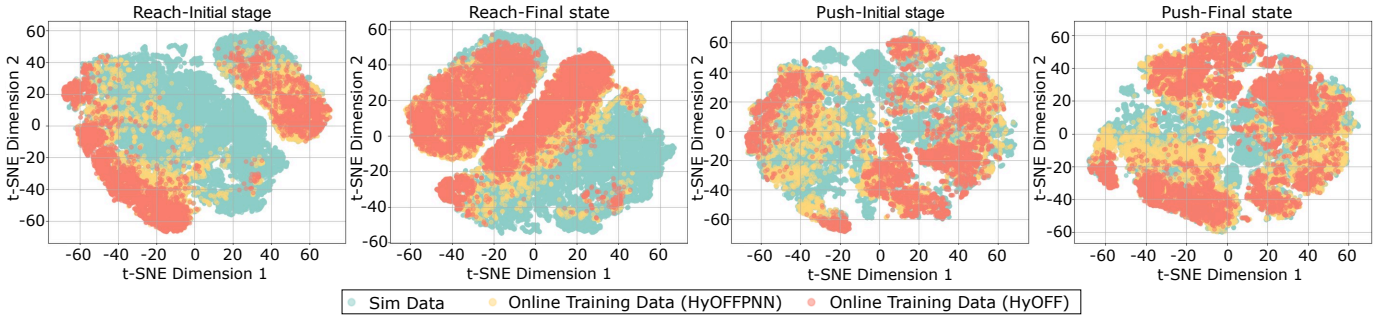
To further study how the diversity of offline dataset affects the offline policy training and online learning, and investigate the effect of PNN and simulation dataset on online fine-tuning, we also compare the learning curves of HyOFFPNN, HyOFF, OFF-Real, HyOFFPNN-BS and HyOFFPNN-TS in the offline and online learning process, and analyze the distribution of samples of simulation dataset, real-world demo dataset and online learning data via t-SNE [35]. The results are shown in Fig. 2, 3 and 4.

Results in Fig. 2 show that, although the OFF-Real agent learns a similar optimal performance to the HyOFFPNN and HyOFF agents after offline training with only real-world demo dataset, its performance decreases dramatically after directly transferring to the real-world tasks. On the other hand, the HyOFF agent learns from hybrid dataset consisting of additional full simulation dataset and real-world demo dataset at a much faster speed and obtains a much better performance than the OFF-Real agent in the real world. This indicates that the diverse simulation data not only improves the offline learning, but also allows the agent to adapt better the real-world tasks. Moreover, with PNN transferring the offline policy to the real world, our HyOFFPNN agent learns even much faster and achieves a better performance than the HyOFF agent. That means transferring via PNN can not only be able to effectively retain the policy learned from the hybrid dataset and bridge the gap between simulation and reality (e.g., dynamics difference in terms of friction, damping, joint constraints), but allows our HyOFFPNN agent to explore in a more diverse distribution of samples during online learning (Fig. 3 (b)), which results in

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

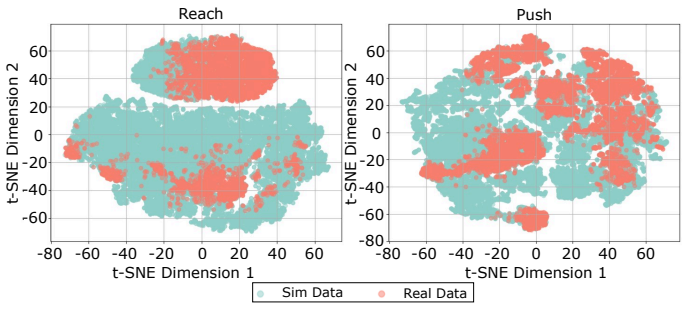


(a) Visualized sample distribution of online learning data for the OFF-Real agent, in comparison to collected real-world demo data

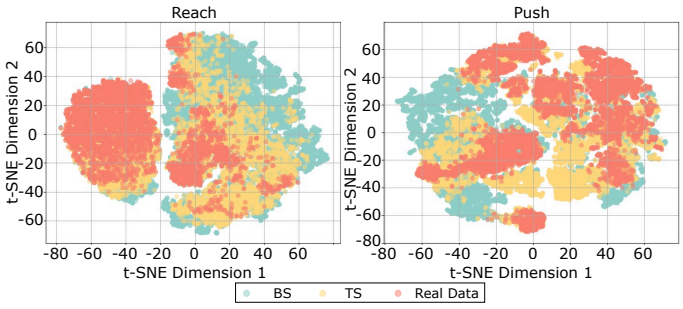


(b) Visualized sample distribution of online learning data for HyOFFPNN and HyOFF agents, in comparison to full simulation data

Fig. 3: Visualized sample distribution of online learning data at the initial, final stages of HyOFFPNN, HyOFF and OFF-Real agents, in comparison to full simulation data, real-world demo data, collected for offline learning (via t-SNE).



(a) Real-world demo and full simulation data



(b) Real demo data, BS and TS simulation data

Fig. 4: Visualized sample distribution of real-world demo data, full simulation data, simulation data collected during the early (bottom 50% simulation data, BS) and later (top 50% simulation data, TS) training stage, used for offline policy learning.

a faster learning speed and higher performance compared to HyOFF agent.

Lastly, we study how the sample distribution of simulation dataset collected during different learning stages affects the offline policy training and online learning. As illustrated in Fig. 2, although the HyOFFPNN-TS and HyOFFPNN-BS agent learns a similar performance to our HyOFFPNN agent from a hybrid dataset consisting of the real-world demo dataset and same amount of simulation samples collected above and below 'Medium' performance respectively during offline training, the HyOFFPNN-TS agent learns only faster and better than the OFF-Real agent from real-world demo dataset, while the learning speed and performance of the HyOFFPNN-BS agent are close to our HyOFFPNN agent, in both offline learning and after transferring to the real-world tasks. Our further analysis in Fig. 4 shows that, the simulation data collected below 'Medium' performance during early training stage is more exploratory and close to the full simulation dataset, while the simulation data collected above 'Medium' performance during the later training stage is more close to the real-world demo dataset. The faster learning speed and better performance of the HyOFFPNN-BS agent in offline and online learning after transferring to the real-world tasks could probably benefit from the more diverse sample distribution of simulation dataset collected during the early training stage of the PPO agent in simulation (below 'Medium' performance), compared to the simulation dataset collected during the later training stage of the PPO agent in simulation (above 'Medium' performance) for the HyOFFPNN-TS agent.

VII. CONCLUSIONS

This paper presents an offline policy learning and transfer via PNN method. The proposed method allows an agent to learn a policy from a hybrid dataset consisting of simulation data of high diversity and real-world demonstration data of high quality via offline reinforcement learning. We assume

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

that the dynamics of the simulation and the real world do not match, but the state space is the same. To address the policy extrapolation error and potentially catastrophic failures because of out-of-distribution actions and sim-to-real gap, PNN was used to transfer the offline policy to the real world for further online learning. Experimental results in two robotic manipulation tasks: Reach and Push, both with a Ned robotic arm show that, high-diversity simulation data and high-quality real-world demo data allow for faster offline policy learning and adapting to the real world task better and faster. Further analysis via ablation study indicates that transferring via PNN can not only be able to effectively retain the policy learned from the hybrid dataset and bridge the gap between simulation and reality, but allows the agent to explore in a more diverse distribution of samples during online learning. Furthermore, results show that our method benefits from simulation data collected during initial exploratory stage of agent training in simulation rather than the later one.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] G. Li and R. Gomez, "Realizing full-body control of humanoid robots," *Nature Machine Intelligence*, vol. 6, no. 9, pp. 990–991, 2024.

[3] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, "Deep reinforcement learning for robotics: A survey of real-world successes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 27, 2025, pp. 28 694–28 698.

[4] R. F. Prudencio, M. R. Maximo, and E. L. Colombini, "A survey on offline reinforcement learning: Taxonomy, review, and open problems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 8, pp. 10 237 – 10 257, 2023.

[5] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[6] S. Y. Arnob, R. Islam, and D. Precup, "Importance of empirical sample complexity analysis for offline reinforcement learning," *arXiv preprint arXiv:2112.15578*, 2021.

[7] K. Schweighofer, M.-c. Dinu, A. Radler, M. Hofmarcher, V. P. Patil, A. Bitto-Nemling, H. Eghbal-zadeh, and S. Hochreiter, "A dataset perspective on offline reinforcement learning," in *Proceedings of Conference on Lifelong Learning Agents*, vol. 199. PMLR, 2022, pp. 470–517.

[8] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[9] Y. Feng, N. Hansen, Z. Xiong, C. Rajagopalan, and X. Wang, "Fine-tuning offline world models in the real world," in *Proceedings of the Conference on Robot Learning*, vol. 229, 2023, pp. 425–445.

[10] H. Ju, R. Juan, R. Gomez, K. Nakamura, and G. Li, "Transferring policy of deep reinforcement learning from simulation to reality for robotics," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1077–1087, 2022.

[11] L. Da, J. Turnau, T. P. Kutralingam, A. Velasquez, P. Shakarian, and H. Wei, "A survey of sim-to-real methods in rl: Progress, prospects and challenges with foundation models," *arXiv preprint arXiv:2502.13187*, 2025.

[12] T. Carr, M. Chli, and G. Vogiatzis, "Domain adaptation for reinforcement learning on the atari," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, vol. 4, 2019, pp. 1859–1861.

[13] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model," *arXiv preprint arXiv:1610.03518*, 2016.

[14] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.

[15] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[16] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[17] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Proceedings of Conference on Robot Learning*, vol. 78. PMLR, 2017, pp. 262–270.

[18] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *Proceedings of International Conference on Machine Learning*, vol. 9. PMLR, 2019, pp. 2052–2062.

[19] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," *Advances in Neural Information Processing Systems*, vol. 32, pp. 11 761–11 771, 2019.

[20] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[21] M. Nakamoto, S. Zhai, A. Singh, M. Sobol Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine, "Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 62 244–62 269, 2023.

[22] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *Proceedings of International Conference on Learning Representations*, 2022.

[23] Z. Qiao, J. Lyu, K. Jiao, Q. Liu, and X. Li, "Sumo: Search-based uncertainty estimation for model-based offline reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 19, 2025, pp. 20 033–20 041.

[24] S. Hu, Z. Fan, C. Huang, L. Shen, Y. Zhang, Y. Wang, and D. Tao, "Q-value regularized transformer for offline reinforcement learning," in *Proceedings of International Conference on Machine Learning*, vol. 235, 2024, pp. 19 165–19 181.

[25] S. E. Ada, E. Oztop, and E. Ugur, "Diffusion policies for out-of-distribution generalization in offline reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3116–3123, 2024.

[26] Y. Song, Y. Zhou, A. Sekhari, J. A. Bagnell, A. Krishnamurthy, and W. Sun, "Hybrid rl: Using both offline and online data can make rl efficient," *arXiv preprint arXiv:2210.06718*, 2022.

[27] H. Niu, Y. Qiu, M. Li, G. Zhou, J. Hu, X. Zhan *et al.*, "When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36 599–36 612, 2022.

[28] T. Xie, N. Jiang, H. Wang, C. Xiong, and Y. Bai, "Policy finetuning: Bridging sample-efficient offline and online reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 395–27 407, 2021.

[29] Y. Mao, C. Wang, B. Wang, and C. Zhang, "Moore: Model-based offline-to-online reinforcement learning," *arXiv preprint arXiv:2201.10070*, 2022.

[30] Y. Zhang, J. Liu, C. Li, Y. Niu, Y. Yang, Y. Liu, and W. Ouyang, "A perspective of q-value estimation on offline-to-online reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 15, 2024, pp. 16 908–16 916.

[31] G. Chaudhary, W. U. Mondal, and L. Behera, "Moorl: A framework for integrating offline-online reinforcement learning," *arXiv preprint arXiv:2506.09574*, 2025.

[32] J. Ren, G. Swamy, S. Wu, D. Bagnell, and S. Choudhury, "Hybrid inverse reinforcement learning," in *Proceedings of International Conference on Machine Learning*, vol. 235, 2024, pp. 42 428–42 448.

[33] X. Wen, X. Yu, R. Yang, H. Chen, C. Bai, and Z. Wang, "Towards robust offline-to-online reinforcement learning via uncertainty and smoothness," *Journal of Artificial Intelligence Research*, vol. 81, pp. 481–509, 2024.

[34] H. Zhang, W. Xu, and H. Yu, "Policy expansion for bridging offline-to-online reinforcement learning," *arXiv preprint arXiv:2302.00935*, 2023.

[35] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.