An Offline-Transfer-Online Framework for Cloud-Edge Collaborative Distributed Reinforcement Learning

Tianyu Zeng, Xiaoxi Zhang, *Member, IEEE*, Jingpu Duan, *Member, IEEE*, Chao Yu, Chuan Wu, *Senior Member, IEEE*, and Xu Chen, *Senior Member, IEEE*

***Abstract*—Recent advances in deep reinforcement learning (DRL) have made it possible to train various powerful agents to perform complex tasks in real-time environments. With the next-generation communication technologies, making cloud-edge collaborative artificial intelligence service with evolved DRL agents can be a significant scenario. However, agents with different algorithms and architectures in the same DRL scenario may not be compatible, and training them is either time-consuming or resource-demanding. In this article, we design a novel cloud-edge collaborative DRL training framework, named Offline-Transfer-Online, which is a new approach that can speed up the convergence of online DRL agents at the edge by interacting with offline agents in the cloud, with the minimum data interchanged and without relying on high-quality offline datasets. Therein, we propose a novel algorithm-independent knowledge distillation algorithm for online RL agents, by leveraging pre-trained models and the interface between agents and the environment to transfer distilled knowledge among multiple heterogeneous agents efficiently. Extensive experiments show that our algorithm can accelerate the convergence of various online agents in a double to decuple speed, with comparable reward achieved in different environments.**

***Index Terms*—Distributed training, offline-transfer-online, deep reinforcement learning, cloud-edge collaborative networks.**

I. Introduction

With the rapid commercialization of mobile edge computing (MEC) and the fifth-generation (5G) communication technologies, real-time mobile applications such as virtual reality (VR), augmented reality (AR), and video analytics have gained increasing popularity [1]. The foreseen sixth-generation (6G) [2], [3] services are expected to provide even higher data rates [4] and support native artificial intelligence (AI) [5], including deep learning (DL) [6] and deep reinforcement learning (DRL) [7]. However, the inherent resource scarcity of mobile devices prohibits the wide adoption of computation-and-data-intensive approaches at the edge. Therefore, cloud-edge collaborative networks become a promising paradigm to support future 5G/6G based AI applications, leveraging both abundant cloud resources and lightweight edge services in proximity to the data.

As one of the most popular AI approaches for mobile applications, reinforcement learning (RL) is a general and effective algorithm framework for sequential decision-making [7]. With deep learning [6], state-of-the-art deep RL methods not only have demonstrated their exceptional ability in performing complex human tasks [8], [9], but also show great potential in improving real-time mobile edge applications such as autonomous driving and video analytics [3]. In the area of digital twin such as sensing [10] and communication [11], DRL is also adopted, where environmental observation and feedback are fed into virtual objects to learn the optimal policies that should be applied onto physical objects. In this sense, DRL models can be deployed at the edge and used to generate online predictions for these real-time applications. However, different from offline supervised learning, DRL agents are usually trained online through interactions with environments, which could take hours [9] or even months [12] to complete due to the highly-restrictive computation ability of the edge. It then can benefit much from the cloud-edge collaboration by leveraging the computation resources and services in the cloud (or edge clusters, regional clouds) as substitutes. However, existing works that consider deploying DRL in the cloud and edge mainly focus on scheduling resources to support the model training but fall short of enabling communication-efficient interactions between the learning models themselves for boosting online DRL training.

Inspired by the powerful capability of pre-trained models, some recent studies advocate to train DRL models in an offline manner [13], using pre-collected datasets [14] instead of online

Manuscript received 24 March 2023; revised 30 November 2023; accepted 20 January 2024. Date of publication 31 January 2024; date of current version 18 March 2024. This work was supported in part by NSFC under Grant 62102460, Grant 62076259, and Grant U20A20159, in part by Hong Kong RGC under Grant HKU 17208920 and Grant C7004-22G (CRF), in part by the Guangzhou Science and Technology Plan Project under Grant 202201011392 and Grant 2024A04J6367, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515012982 and Grant 2021B151520008, and in part by the Fundamental and Application Research Funds of Guangdong province under Grant 2023A1515012946. Recommended for acceptance by Y. Yang. *(Corresponding author: Xiaoxi Zhang.)*

Tianyu Zeng, Xiaoxi Zhang, Chao Yu, and Xu Chen are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China (e-mail: zengty@mail2.sysu.edu.cn; zhangxx89@mail.sysu.edu.cn; yuchao3@mail.sysu.edu.cn; chenxu35@mail.sysu.edu.cn).

Jingpu Duan is with the Institute of Future Networks, Southern University of Science and Technology, Shenzhen 518055, China, and also with the Department of Communications, Pengcheng Laboratory, Shenzhen 518066, China (e-mail: duanjp@pcl.ac.cn).

Chuan Wu is with the Department of Computer Science, University of Hong Kong, Hong Kong, China (e-mail: cwu@cs.hku.hk).

Digital Object Identifier 10.1109/TPDS.2024.3360438

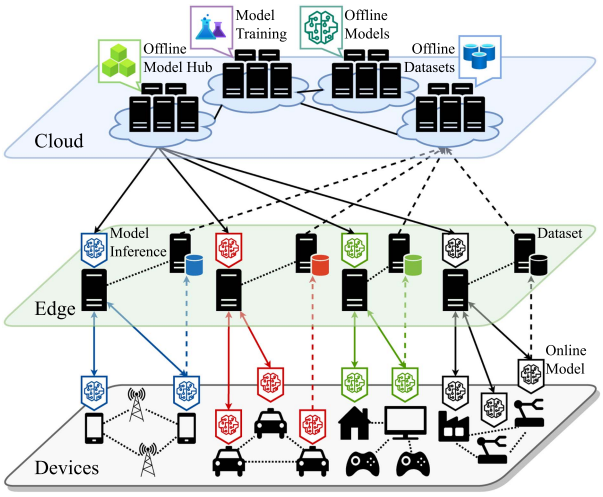


Fig. 1. Cloud-edge collaborative DRL framework.

interactions with the environments. This provides a feasible alternative of DRL training to address the problem that using fresh data only in DRL training can be costly and inefficient for real-time mobile applications. However, it is infeasible to deploy offline RL training in resource-scarce edge devices due to the required large-scale pre-obtained datasets and potentially big learning models. Moreover, these offline RL models may not be able to generate adaptable predictions in online dynamics for real-time applications. Therefore, we propose to exploit the architecture of cloud-edge collaboration, by deploying offline RL in the cloud as a service and performing distributed online RL at the edge to make real-time predictions for smart applications. As shown in Fig. 1, in our distributed cloud-edge collaborative DRL framework, training offline models is supported by the abundant cloud computation resources and storage capacity. In contrast, online RL agents at the edge can efficiently interact with the environments (e.g., if the application scenario is to optimize the mobile cell selection) to continuously obtain fresh data for online model adaptation. Although there are some approaches for improving online RL with offline RL [15], there remains a central question: *how to design communication-efficient methods to boost the online model training through offline models that use potentially different models and policies from the online counterparts and are deployed remotely at geographically distant networks?* To this end, our system is a novel cloud-edge collaborative distributed DRL framework, that is scalable, lightweight, and does not require expert-level policies or supreme-quality datasets from the offline models.

Technically, leveraging the agent-environment interaction interface, we design a three-stage DRL training framework, named Offline-Transfer-Online (OTO). It can effectively transfer the knowledge online from pre-trained RL agents to the target one by achieving exceptional communication efficiency and learning speed. Unlike prior offline-to-online [15] and transfer learning [16], [17] methods, ours does not require to feed pre-obtained offline raw data into the replay buffers of the online agents or the agents with same architectures. To facilitate the training of different state-of-the-art DRL agents in various scenarios (e.g., deploying the training at edge devices or warehouse computer clusters), OTO is designed to be distributed, scalable, and generalizable. That is, it can be applied to train heterogeneous DRL agents (e.g., using deep Q-network (DQN), actor-critic algorithms, etc.) simultaneously through concurrent online interactions with the environments and pre-trained teacher RL agents. Experiments show that our OTO framework can accelerate the training process of various online RL agents by twice to ten times faster with minimum communication overhead incurred, which demonstrates the efficacy and the efficiency of our method. Specifically, we make the following *technical contributions*.

- We propose a novel cloud-edge collaborative DRL framework where heterogeneous DRL models can be trained simultaneously by utilizing both offline agents in the cloud platform and real-time interactions between online edge agents with the environment. Compared with fully online RL algorithms, the essential idea of OTO is to take full advantage of powerful models built at resource-richer locations such as cloud and their valuable offline datasets, but only require transmitting a small volume of data between the edge and cloud. The workflow of OTO contains three stages periodically repeated. First, it trains an offline RL model under pre-obtained datasets, both of which the online agents can be oblivious to. Then, only the inference results of the offline model are transmitted to the online agents at the edge, wherein we nicely integrate conservative Q-learning (CQL) [13] and relational knowledge distillation (RKD) [18] to boost online model training. Finally, online agents (distributed at edge servers or powerful end devices with sufficient resources to support training) continuously interact with environments and train their models just like traditional DRL.
- At the core subroutine of OTO, our designed algorithm-independent knowledge distillation (AIKD) strategy is a novel transfer learning approach for boosting the Q-value learning which can be used as policy evaluation in heterogeneous multi-agent RL. Specifically, we design a new loss function with RKD integrated to learn conservative lower-bounds of the optimal Q-values. Given this, we can effectively transfer the knowledge of the offline models to the online training process and eliminate overestimation. Besides, our loss function for each agent does not sample the inputs from the offline (or other agents') datasets, which can prevent severe action distribution shift problems that classic offline RL suffers and offer better data privacy. Moreover, OTO efficiently utilizes online feedback to correct the bootstrap error of Q-values for the out-of-distribution actions of the learned policy that cannot be efficiently evaluated by using the offline data solely. Besides, one advantage over prior transfer learning methods for offline-to-online RL is that our information transmitted between the teacher models in the cloud and the edge student models is limited to the minimum, i.e., only inference results rather than raw data of the offline agents. Therefore, OTO is communication-efficient and suits well online DRL in edge computing systems with WAN bandwidth being the bottleneck.

- We propose several simple but effective techniques to augment the DRL architecture. For instance, we also revise the angle term to be a new metric which can extend the RKD metric to be compatible with discrete actions. We also design a fusion-policy architecture to address that problem that Q-value based DRL implementation can not be directly applied into on-policy RL. By realizing the offline RL knowledge provision through the agent-environment interaction interface, our OTO framework is lightweight and applicable to large-scale DRL systems with heterogeneous and geo-distributed online agents, which is particularly suitable for online DRL deployed on distributed and resource-constrained edge devices. Moreover, our OTO treats the offline models as blackboxes which are allowed to have different structures and policies from the online models, saving the efforts of model selection on the cloud side.

This paper is organized as follows. Prior works are listed in Section II. In Sections III and IV, we introduce the preliminaries and our methodology, respectively. Sections V and VI show and analyze the results of the conducted experiments. We conclude this paper with Section VII.

II. Related Work

We now discuss prior works that are related to our framework in this section.

A. Cloud-Edge Collaborative Learning Frameworks

Prior studies on cloud-edge collaborative learning frameworks mainly focus on federated learning [19]. Leveraging the hierarchical characteristic of cloud-edge collaborative networks, the authors in [20] proposed a client-edge-cloud hierarchical federated learning system, which allows multiple edge servers to perform partial model aggregation. Focusing on DRL applications, the authors in [21] proposed a distributed cloud-edge real-time training architecture for actor-critic based DRL agents interacting with the realities. Thus, designing a general-purpose cloud-edge collaborative DRL training framework should be on the agenda.

B. Online DRL With Environments

With the ideas of mini-batching and Q-learning [22], the authors in [9] proposed the first practical DRL method, i.e., DQN, to train agents online with environments. Since then, several improvements [23] have been made to accelerate and stabilize the training process. Besides, deep actor-critic methods [24], [25], [26] inspired by [27] are introduced, which can work well for discrete and continuous tasks [26]. Although these methods are of numerous advantages, the training time is long, and their architectures vary, leading to unideal training efficiency especially when applied at edge devices with limited resources.

C. Offline DRL With Datasets

With the growing computational capability, using interaction datasets to train RL agents in an offline way is possible [14], and some novel methods have emerged in recent years [13]. Thereinto, the authors in [13] proposed the CQL algorithm to accelerate the training procedure from the aspect of metric learning. However, these methods require well-recorded datasets which are large and resource-demanding, and are hard to deploy in resource-constrained scenarios when the models are of large sizes.

D. Improving Online RL With Offline RL

As a topical field, prior studies have employed offline DRL agents and datasets to enhance the training performance of online agents from the aspect of sample efficiency. The authors in [15] introduced a novel offline-to-online RL training method based on the CQL algorithm and balanced experience replay. Additionally, the authors in [28] proposed an adaptive update scheme based on the CQL algorithm by alternatively training with online interactions and previously collected datasets. Nevertheless, these methods still rely on optimal datasets and equivalent environments.

E. Transfer Learning in DRL

Using transfer learning to improve the performance of RL agents has achieved expressive success [16]. There are many variants in transfer learning, e.g., inductive transfer learning, transductive transfer learning, unsupervised transfer learning, etc [29], [30]. Thereinto, the authors in [17] made policy and value distillation to transfer the knowledge among actor-critic agents. These methods only consider the knowledge transfer among the agents with same architectures. When it comes to transferring among different algorithms or environments, these methods might be hard to apply readily, as it is a common challenge in transfer learning [29], [30]. Thus, finding new transfer training methods that are compatible with different agents is necessary.

III. Preliminaries

The preliminaries of our proposed framework are presented in this section.

A. Online Reinforcement Learning

Reinforcement learning problems generally consider an online Markov decision process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$, with spaces of states $\mathcal{S}$ and actions $\mathcal{A}$, conditional transition probabilities between states $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, a reward $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$, and a discount factor $\gamma \in [0, 1]$. Formally, at each time step t, the agent observes a state $s_t \in \mathcal{S}$, performs an action $a_t \in \mathcal{A}$ according to her policy $\pi(a|s_t)$, receives a reward r_t from the environment, and then transitions to the next state s_{t+1}. The objective of the agent is to maximize the expected total reward (also called V-value function) $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$.

There are two RL schemes according to the consistency of the current and the data-generation policies, i.e., on-policy and off-policy RL algorithms. We first focus on off-policy algorithms, which evaluate the agent's parametric Q-value function $Q_\theta(s, a)$

and improve its optional parametric policy $\pi_\phi(a|s)$ with samples generated by any given policies, where θ and ϕ could be neural networks. For instance, Q-learning methods (e.g., DQN) learn the Q-value function by iteratively applying the Bellman operator $(\mathcal{B}^\pi Q)(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim T(s'|s,a)}[\max_{a' \in \mathcal{A}} Q(s',a')]$, since $\mathcal{B}^\pi$ is a contraction mapping with a fixed point (the optimal $Q(s,a)$) that exists according to Banach's theorem. The loss function for training Q_θ in DQN is:

$$\mathcal{L}_{\texttt{DQN}}(\theta) = \mathop{\mathbb{E}}_{(s,a,s') \sim \mathcal{B}} \left[\left(\left(Q_\theta - \mathcal{B}^\pi Q_{\bar{\theta}} \right)(s,a) \right)^2 \right], \quad (1)$$

with a replay buffer $\mathcal{B}$ and delayed parameters $\bar{\theta}$ (cf. θ). Differently, actor-critic methods alternate between updating Q^π with the current policy π (critic) and improving π towards maximizing the expected total reward (actor). For example, soft actor-critic [26] (SAC) iteratively minimizes the following losses, with a temperature parameter $\alpha_{\texttt{SAC}}$:

$$\mathcal{L}_{\texttt{SAC}}^{\texttt{critic}}(\theta) = \mathop{\mathbb{E}}_{(s,a,s') \sim \mathcal{B}} [(Q_\theta(s,a) - r(s,a) - \gamma \mathop{\mathbb{E}}_{a' \sim \pi_\phi} \left[Q_{\bar{\theta}}(s',a') - \alpha_{\texttt{SAC}} \log \pi_\phi \left(a' | s' \right) \right])^2], \quad (2)$$

$$\mathcal{L}_{\texttt{SAC}}^{\texttt{actor}}(\phi) = \mathop{\mathbb{E}}_{s \sim \mathcal{B}, a \sim \pi_\phi} \left[\alpha_{\texttt{SAC}} \log \pi_\phi \left(a|s \right) - Q_\theta(s,a) \right]. \quad (3)$$

On-policy RL algorithm is another important research topic in reinforcement learning. Since the data-generation policy is consistent with the current policy, on-policy algorithms generally require small replay buffers, which leads to high performance in training speed. What is more, deep on-policy RL algorithms, such as trust region policy optimization (TRPO) [24] and proximal policy optimization (PPO) [25] algorithms, use V-value function to estimate the utilities of states instead of Q-value function. Typically, off-policy algorithms are more sample-efficient than on-policy ones, while the training performance of on-policy algorithms is more stable than the one of off-policy algorithms [7].

B. Offline Reinforcement Learning

Existing offline RL algorithms are typically off-policy RL algorithms using pre-collected datasets $\mathcal{D}$ for training. Among them, CQL [13] advocates to pessimistically learn the lower bound of the true Q-value functions, using the following loss function:

$$\mathcal{L}_{\texttt{CQL}}(\theta) = \max_\mu \mathcal{R}_{\texttt{CQL}}(\mu, \theta) + \mathcal{L}_{\texttt{task}}(\theta), \quad (4)$$

$$\mathcal{R}_{\texttt{CQL}}(\mu, \theta) = \mathcal{R}(\mu) + \alpha_{\texttt{CQL}} \mathop{\mathbb{E}}_{s \sim \mathcal{D}} \left[\mathop{\mathbb{E}}_{a \sim \mu} [Q_\theta] - \mathop{\mathbb{E}}_{a \sim \hat{\pi}_\beta} [Q_\theta] \right], \quad (5)$$

$$\mathcal{L}_{\texttt{task}}(\theta) = \frac{1}{2} \mathop{\mathbb{E}}_{(s,a,s') \sim \mathcal{D}} \left[\left(\left(Q_\theta - \mathcal{B}^\pi Q_{\bar{\theta}} \right)(s,a) \right)^2 \right], \quad (6)$$

where $\alpha_{\texttt{CQL}}$ is a trade-off factor, $\mu = \mu(a|s)$ denotes the state-marginal in $\mathcal{D}$, $\hat{\pi}_\beta = \hat{\pi}_\beta(a|s) := \frac{\sum_{s',a' \in \mathcal{D}} \mathbf{1}[s'=s, a'=a]}{\sum_{s' \in \mathcal{D}} \mathbf{1}[s'=s]}$ denotes the empirical behavior policy, and $\mathcal{R}(\mu)$ denotes a particular regularizer. When $\mathcal{R}(\mu)$ is set to be the KL-divergence against a prior distribution $\rho(a|s) = \text{Unif}(a)$, i.e., $\mathcal{R}(\mu) = -D_{\text{KL}}(\mu, \rho)$, $\mathcal{R}_{\texttt{CQL}}(\mu, \theta)$ can be simplified further as

$$\mathcal{R}_{\texttt{CQL}}(\theta) = \alpha_{\texttt{CQL}} \mathop{\mathbb{E}}_{s \sim \mathcal{D}} \left[\log \sum_a \exp(Q_\theta) - \mathop{\mathbb{E}}_{a \sim \hat{\pi}_\beta} [Q_\theta] \right], \quad (7)$$

which can be applied to off-policy algorithms.

IV. Offline-Transfer-Online Framework

In this section, we propose our Offline-Transfer-Online (OTO) framework to speed-up online DRL model training through lightweight knowledge transfer from cloud-based offline RL models and datasets.

To accelerate the online training process with offline RL, perhaps the most straightforward way is to store offline and online data into the replay buffer, which is similar to policy distillation. However, it is memory-inefficient and may require strategical sample selection. Meanwhile, this intuitive method can lead to negative transfer, due to the differences between the environments. Therefore, we choose to train the online agents with the data sampled from the replay buffer that only contains the online dynamics, and intend to conservatively learn the lower-bound of the Q-value function when utilizing the inference results from offline agents as offline data to mitigate the over-estimation.

A. AIKD for Online Reinforcement Learning

1) Knowledge Distillation With Interface: As elaborated above, the way of using inference results (actions) from the offline policy rather than the entire dataset shares the spirit of knowledge distillation in that we intend to compress and transfer the learned knowledge from a potentially better model to speed up the training of new models under communication-constraint scenarios. A naive solution to achieve this is to *distill the neural networks in agent models directly*. However, due to the architectural differences in the agent networks, direct knowledge distillation could be lacking in efficiency [29], [30]. Motivated by this, we attempt to explore a knowledge distillation method with a unified interface to boost different online agents with the knowledge transferred from various offline agents, while imposing no restrictions on the types of those agents. Thus, we leverage the agent-environment interaction interface shown in Fig. 2 as the distillation basis, which provides the agents with a general protocol on the actions of the interacted environment without knowing the details of the participating agents. For example, we show in Fig. 2 that online DRL agents deployed at the edge for various applications may adopt different types of models and policies. They interact with the environment through this interface by passing limited information that is in the same form. Thus, the agents with different models or architectures can share the knowledge if the corresponding interacted environments are controlled similarly, leading to a scalable distributed DRL paradigm.

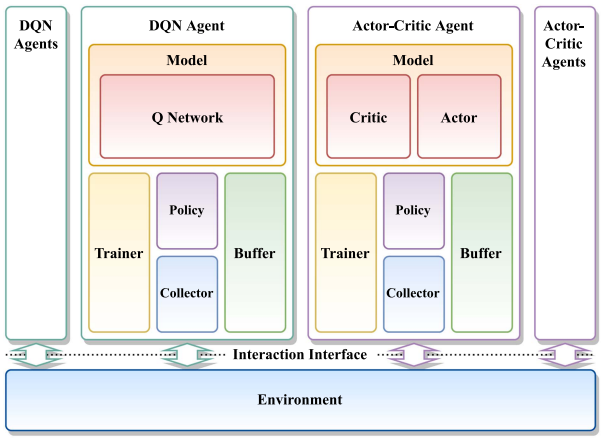


Fig. 2. Agent-environment interaction interface in RL.

2) CQL-Based Loss Function: Different from the one in supervised learning, there is no ground truth data for knowledge distillation in RL, and our teacher-student distillation process is online, making it non-trivial to design a proper loss function. To address this issue and enable our framework to accelerate the training for heterogeneous RL agents, we reform CQL's objective function (5) as follows. First, we substitute the online replay buffer $\mathcal{B}$ for the offline dataset $\mathcal{D}$, and then replace the empirical behavior policy $\hat{\pi}_\beta$ calculated by the state-action dynamics in the offline data $\mathcal{D}$ with a teacher policy $\hat{\pi}_\tau$, which can also be the one used by the offline agents, or an ensemble. This simple modification has the benefits of not only increasing data efficiency and safety, but also mitigating the distribution shift problem due to choosing offline dynamics to train the Q-value function but using current policy's output actions as target Q-values in the mean squared error term. Finally, we choose $\mathcal{R}(\mu) = -D_{\mathrm{KL}}(\mu, \rho)$, which can simplify (5) in that only the actions are needed to calculate the loss, and our base objective function can then be formulated as:

$$\mathcal{L}^{\texttt{BASE}}_{\texttt{AIKD}}(\theta) = \alpha_{\texttt{AIKD}}\mathcal{R}^{\texttt{BASE}}_{\texttt{AIKD}}(\theta) + \alpha_{\texttt{task}}\mathcal{L}_{\texttt{task}}(\theta), \tag{8}$$

$$\mathcal{R}^{\texttt{BASE}}_{\texttt{AIKD}}(\theta) = \max_{\mu} \mathop{\mathbb{E}}_{s\sim\mathcal{B}} \left[\mathop{\mathbb{E}}_{a\sim\mu} [Q_\theta] - \mathop{\mathbb{E}}_{a\sim\hat{\pi}_\tau} [Q_\theta] \right] - D_{\mathrm{KL}}(\mu, \rho) \tag{9}$$

$$= \mathop{\mathbb{E}}_{s\sim\mathcal{B}} \left[\log \sum_{a} \exp\left(Q_\theta\right) - \mathop{\mathbb{E}}_{a\sim\hat{\pi}_\tau} [Q_\theta] \right], \tag{10}$$

where $\mathcal{L}_{\texttt{task}}$ is the task loss (e.g., (1) or (2)), $\alpha_{\texttt{AIKD}}$ and $\alpha_{\texttt{task}}$ are trade-off factors.

3) Additional Regularizer for Improving Estimation: After removing the reliance on the offline (or teachers') raw data and using online collected data for Q-value function learning, we discover that the proposed loss function may lead to over-estimation of Q-values. Specifically, our $\mathcal{R}^{\texttt{BASE}}_{\texttt{AIKD}}$ in (10) is in fact lower than the one that yields a conservative lower-bound of the optimal Q-values previously used in the CQL method, and their gap decreases with the growing of the data size in the replay buffer. We prove this argument in Appendix A, available online. Therefore, extra regularizers should be added to eliminate the over-estimation by compensating for the loss function drift due to using the online collected data for training. In addition, the choice of such regularizer should have the benefits of: 1) bridging the gap between the offline (teachers') actions and the online agent's (students'); 2) decreasing with more data collected to cover the diminishing model drift. To address this, we extend the design of RKD-formed regularizers as follows. First, the regularizer consists of two parts, defined as:

$$\mathcal{R}^{\texttt{RKD}}_{\texttt{AIKD}}(\hat{\pi}_\tau, \hat{\pi}_\sigma) = \mathcal{R}_{\texttt{RKD-D}}(\hat{\pi}_\tau, \hat{\pi}_\sigma) + \mathcal{R}_{\texttt{RKD-A}}(\hat{\pi}_\tau, \hat{\pi}_\sigma), \tag{11}$$

where $\mathcal{R}_{\texttt{RKD-D}}$ measures the distance between the teacher policy $\hat{\pi}_\tau$ and the student policy $\hat{\pi}_\sigma$, and $\mathcal{R}_{\texttt{RKD-A}}$ measures the relational adaptation distance between them. The difficulty is to design a proper formulation of $\mathcal{R}_{\texttt{RKD-A}}$ to suit the scenario where the communicated data between offline teacher policies and online student ones should be in small sizes. Therefore, we propose to use the actions with the maximal probability output by the agents to substitute for the probabilistic policies in (11). In this case, we define $\mathcal{R}_{\texttt{RKD-D}}$ as the negative log likelihood loss for discrete actions and the mean squared error loss for continuous ones, and formally define $\mathcal{R}_{\texttt{RKD-A}}$ as

$$\mathcal{R}_{\texttt{RKD-A}}(\hat{\pi}_\tau, \hat{\pi}_\sigma) = \mathop{\mathbb{E}}_{s,s'\sim\mathcal{B}} \left[l_{\mathrm{MSE}} \left(\psi^\tau_{\mathrm{A}}(s, s'), \psi^\sigma_{\mathrm{A}}(s, s') \right) \right], \tag{12}$$

where l_{MSE} represents the mean squared error loss. To make it work for discrete actions, which are very common for mobile edge applications, we design ψ^τ_{A} and ψ^σ_{A} to be the adaptations by computing the midpoints of the action vectors, which are continuous and friendly to discrete actions. Recall that the student policy of the online agent is associated with Q_θ, the RKD-formed regularizer can be denoted as $\mathcal{R}^{\texttt{RKD}}_{\texttt{AIKD}}(\theta)$.

The final objective function of our AIKD algorithm is

$$\begin{aligned} \mathcal{L}_{\texttt{AIKD}}(\theta) &= \alpha_{\texttt{AIKD}}\mathcal{R}_{\texttt{AIKD}}(\theta) + \alpha_{\texttt{task}}\mathcal{L}_{\texttt{task}}(\theta) \\ &= \alpha_{\texttt{AIKD}} \left(\delta\mathcal{R}^{\texttt{BASE}}_{\texttt{AIKD}} + \eta\mathcal{R}^{\texttt{RKD}}_{\texttt{AIKD}} \right)(\theta) + \alpha_{\texttt{task}}\mathcal{L}_{\texttt{task}}(\theta), \end{aligned} \tag{13}$$

where $\mathcal{R}^{\texttt{BASE}}_{\texttt{AIKD}}$ follows (10), $\mathcal{R}^{\texttt{RKD}}_{\texttt{AIKD}}$ follows (11), and δ and η are trade-off factors. By employing (13) as the loss function to evaluate the Q-value functions, we can realize efficient knowledge distillation for heterogeneous deep RL agents via the agent-environment interaction interface.

B. Offline-Transfer-Online Reinforcement Learning

We next describe our OTO framework in detail. To anable scalable and distributed DRL systems, OTO supports simultaneous training with multiple offline (i.e., teacher) agents and online (i.e., student) agents with the help of the inference system. The overview of the OTO algorithm is depicted in Fig. 3. Each offline agent contains a training agent in the cloud and an online inferring agent at the edge (deployed in the edge server or powerful devices), while each online agent is just a training agent. The online inferring agent is implemented as the same RL model but using delayed model parameters as the training agent. In default, we use agents that are well trained with offline data as our teacher models, but we also allow the models which are

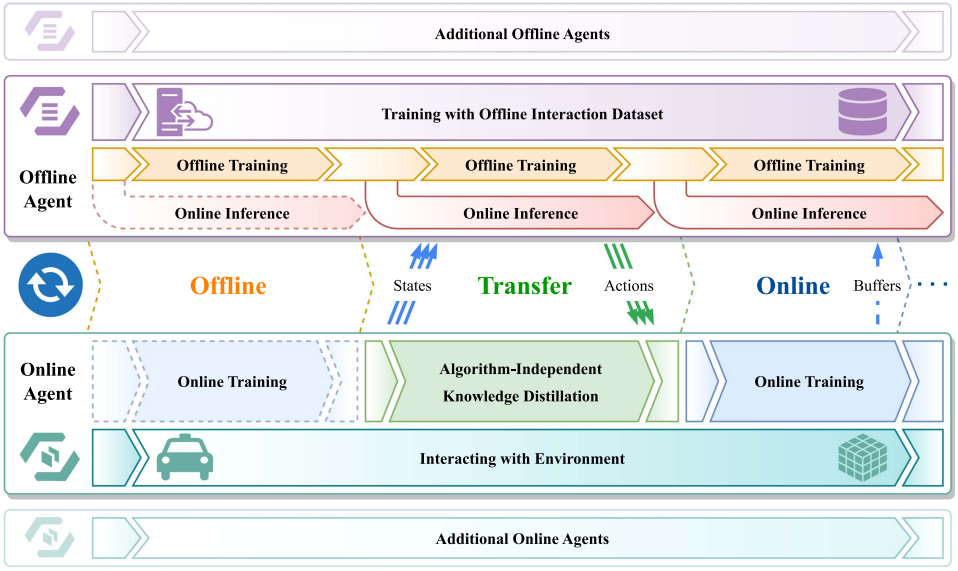


Fig. 3. Overview on Offline-Transfer-Online algorithm.

trained in advance with environments to receive online training, since what we need is only the updated knowledge of the offline agents rather than their raw data. We implemented both types of the teacher agents and show the viability in Section V. Illustrated in Fig. 3, the procedure of the OTO algorithm consists of three stages: Offline, Transfer, and then Online, which can be repeated through each episode. To facilitate the following reading, we categorize the procedure into Algorithms 1 and 2 for offline and online agents, respectively.

1) Offline Stage: Offline stage can be regarded as a pre-train stage, where each offline agent is trained with its own pre-obtained dataset. At the first Offline stage, only the offline training procedure is performed. But since the three stages are continuously transitioning, online inference provided by the offline agent and online training of the online agent (the red and blue arrows in this stage, both with the dashed lines) will be concurrently conducted with the offline training procedure in the following Offline stages. The objective functions in the offline teacher agents and the ones in the online student agents remain the same as their original versions (without our designed augmentation in the loss for the OTO algorithm). For example, if the offline teacher agent is CQL-based and the online student agent is powered by the SAC algorithm, the teacher agent will take (5) as its objective function, and the student agent will update its parameters according to (2) and (3). When the offline teacher agent satisfies the stage-ending requirement (it can be either a time-out or test rewards exceeding a threshold), the Transfer stage will be started immediately.

2) Transfer Stage: Transfer stage is the core of our OTO algorithm. During this stage, the offline teacher agents will transfer their knowledge in the form of inferred actions according to the requests of states generated by the online student agents. At the beginning of the stage, every offline agent will update the online inferring agent on the inference system by replacing it with its training agent. After the deployment, the offline training agent will continue to be trained with its dataset. As for the online student agent, it interacts with the environment but updates its parameters according to the AIKD algorithm. When interacting with the environment, the online agent will collect the states (denoted as s) into batches, and send requests to the corresponding online inferring agent asynchronously. After receiving the inferred actions (denoted as $\hat{a}$) from the offline agent, the online student agent will calculate the loss according to the function defined in (13). In this case, we mainly consider the situation that the state representation and the control method of the online environments are similar to the ones in offline interaction datasets. However, the states in the requests and the inferred actions need to be transferred to the representations of offline and online agents, respectively, if that is not the case. When receiving actions from multiple teacher agents, (13) can be extended into

$$\mathcal{L}_{\texttt{AIKD}}(\theta) = \frac{\alpha_{\texttt{AIKD}}}{N} \sum_{i=1}^{N} \mathcal{R}^{i}_{\texttt{AIKD}}(\theta) + \alpha_{\texttt{task}} \mathcal{L}_{\texttt{task}}(\theta), \quad (14)$$

where N is the teacher agent count, and $\mathcal{R}^{i}_{\texttt{AIKD}}$ is the AIKD regularizer between the ith teacher agent's inference result and the actual chosen action inputted to the environment. Using (14), we manage to transfer the knowledge from multiple teachers to the online student agents, and the OTO algorithm will transition to the next stage when the requirements are met.

3) Online Stage: Online stage is similar to what agents in fully online RL encounter, in which online RL agents interact with the environments and perform model training. In our framework, the offline agents can offer inference to other online agents and perform training concurrently in this stage. Different from the Offline stage, our framework provides another feature, which is that the online agents can optionally submit their data in the replay buffer to the offline agents for renewing their datasets in this stage. The OTO algorithm can choose to end this episode

Algorithm 1: Offline-Transfer-Online (Offline Agents).

Input: Loop `loop`, environment $\mathcal{M}$, and ending requirements ($\mathcal{C}_{\texttt{OFFLINE}}$, $\mathcal{C}_{\texttt{TRANSFER}}$, $\mathcal{C}_{\texttt{ONLINE}}$).
Data: Training agent (Q_θ, optional π_τ), dataset $\mathcal{D}$, and online inferring agent ($\hat{Q}_\theta$, optional $\hat{\pi}_\tau$).
Result: Trained offline agent (Q_θ, optional π_τ) and optionally renewed dataset $\mathcal{D}$.

Initialize (Q_θ, π_τ) and ($\hat{Q}_\theta$, $\hat{\pi}_\tau$).
STAGE OFFLINE:
(Optional if not `loop`) Update $\hat{Q}_\theta \leftarrow Q_\theta$ and $\hat{\pi}_\tau \leftarrow \pi_\tau$.
while *True* **do**
  Train (Q_θ, π_τ) with $\mathcal{D}$ and test with $\mathcal{M}$.
  Break if $\mathcal{C}_{\texttt{OFFLINE}}$ is satisfied.
end
STAGE TRANSFER:
Update $\hat{Q}_\theta \leftarrow Q_\theta$ and $\hat{\pi}_\tau \leftarrow \pi_\tau$.
Handle requests s from online student agents with ($\hat{Q}_\theta$, $\hat{\pi}_\tau$) and return inference results $\hat{a}$.
(Parallel) **while** *True* **do**
  Train (Q_θ, π_τ) with $\mathcal{D}$ and test with $\mathcal{M}$.
  Break if $\mathcal{C}_{\texttt{TRANSFER}}$ is satisfied.
end
STAGE ONLINE:
Update $\hat{Q}_\theta \leftarrow Q_\theta$ and $\hat{\pi}_\tau \leftarrow \pi_\tau$.
while *True* **do**
  Train (Q_θ, π_τ) with $\mathcal{D}$ and test with $\mathcal{M}$.
  Break if $\mathcal{C}_{\texttt{ONLINE}}$ is satisfied.
end
Optionally renew $\mathcal{D}$ with received buffers $\mathcal{B}$.
Go to **STAGE OFFLINE** if `loop`.

Algorithm 2: Offline-Transfer-Online (Online Agents).

Input: Loop `loop`, environment $\mathcal{M}$, and ending requirements ($\mathcal{C}_{\texttt{OFFLINE}}$, $\mathcal{C}_{\texttt{TRANSFER}}$, $\mathcal{C}_{\texttt{ONLINE}}$).
Data: Training online agent (Q_θ, optional $\hat{\pi}_\sigma$) and N offline inferring agents ($\hat{Q}_\theta$, optional $\hat{\pi}_\tau$).
Result: Trained online agent (Q_θ, optional $\hat{\pi}_\sigma$).

Initialize (Q_θ, $\hat{\pi}_\sigma$).
STAGE OFFLINE: (Optional)
while *True* **do**
  Train Q_θ and optionally $\hat{\pi}_\sigma$ with $\mathcal{M}$.
  Test Q_θ or $\hat{\pi}_\sigma$ with test environment $\mathcal{M}$.
  Break if $\mathcal{C}_{\texttt{OFFLINE}}$ is satisfied.
end
STAGE TRANSFER:
(Async) Collect interactions with $\mathcal{M}$.
(Async) Periodically request inference results $\hat{a}$ for batched collected states s from offline inferring agents ($\hat{Q}_\theta$, $\hat{\pi}_\tau$).
while *True* **do**
  Interact and train Q_θ using (14) with $\hat{a}$.
  (Actor-critic) Improve $\hat{\pi}_\sigma$ in SAC style.
  Test Q_θ or $\hat{\pi}_\sigma$ with test environment $\mathcal{M}$.
  Break if $\mathcal{C}_{\texttt{TRANSFER}}$ is satisfied.
end
STAGE ONLINE:
while *True* **do**
  Train and test agent with environment $\mathcal{M}$.
  Break if $\mathcal{C}_{\texttt{ONLINE}}$ is satisfied.
end
Optionally submit buffer $\mathcal{B}$ to offline agents.
Go to **STAGE OFFLINE** if `loop`.

or transition to the next Offline stage as in the loop of our Algorithms 1 and 2 according to the given tasks and tunable parameters to decide the time length of each stage.

C. Implementation and Deployment

We implement our proposed algorithms (i.e., Algorithms 1 and 2) on top of the Tianshou [31] RL framework, which provides a fast pythonic interface for building DRL agents.

1) Implementation on DRL and Fusion Policy Design: For offline agents, the OTO algorithm is applied to not only the DRL algorithms based on offline training methods like CQL, but also the ones that are pre-trained with environments or simulators. With the agent-environment interaction interface, the offline agents in our OTO framework can have different architectures and implementations with different RL frameworks comparing with the online agents, indicating that the proposed algorithm is cross-platform and scalable.

For off-policy based online agents, the OTO training algorithm is applied to Tianshou's implementations of quantile regression deep Q-network (QRDQN) [23] and SAC [26]. We use default hyper parameters from Tianshou, except that the parameter ε in QRDQN's ϵ-greedy policy is set to be the final value of the decay function of ε used in Tianshou. We observe that this adjustment of ε, which controls the extent of exploration, can yield higher final reward achieved.

For on-policy based online agents, due to the differences elaborated in Section III, the Q-value based OTO algorithm can not be applied directly. Therefore, we propose a *fusion-policy architecture for on-policy based online DRL agents* depicted in Fig. 4, which can merge the benefits from off-policy and on-policy algorithms. With our design of integrating auxiliary critic networks to generate Q-values and auxiliary loss for the actor, the OTO training algorithm can be applied to the on-policy based online agents like PPO [25] with the help of fusion-policy architecture and Tianshou's implementation of SAC [26].

2) System Design and Deployment: On the system design side, we leverage a scalable and distributed computing framework, Ray [32], to implement the real-time interaction between the offline and online agents in the transfer stage. Our implemented framework is well compatible with PyTorch [33], meaning that few modifications are required to realize it from scratch. To enhance system efficiency, we use the future-promise mechanism in Python to realize the asynchronous communication among agents in the Transfer stage of the OTO algorithm.

On the deployment side, we deploy the offline training agents in the cloud, and distribute the online inferring agents of the corresponding offline agents in the edge networks according to

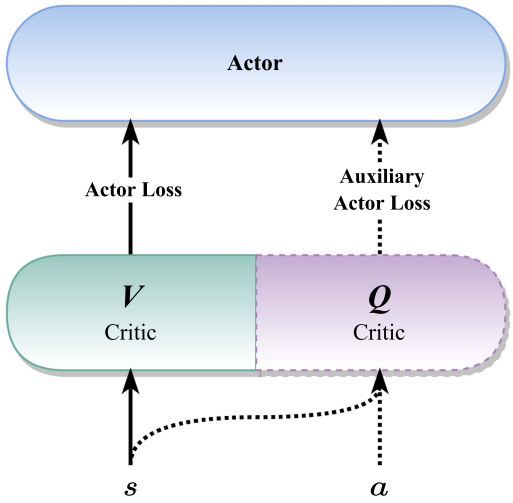


Fig. 4. Fusion-policy architecture for on-policy agents.

their computation demands. To provide low-latency services, the training of online agents can be performed on edge devices that can meet the demands of computation resources.

V. Standard DRL Experiments

In this section, we first conduct discrete and continuous experiments to demonstrate the efficacy of the OTO algorithm using two classic DRL applications. To facilitate the following reading, we suppose that the environments interacted with the online student agents are homogeneous to the ones that are used to train the offline teacher agents.

Our experimental setups are listed as follows. The discrete experiments are conducted with the environments provided by Gym [34] and the Arcade Learning Environment [35]. The corresponding offline datasets are provided by RL Unplugged [36]. The continuous experiments are conducted with the MuJoCo physics engine [37]. All the experiments are conducted on a edge cluster consisting of two devices with 16 CPU cores, an NVIDIA 3090 GPU, and 60 GB memory. The results of the experiments are averaged over ten environments with different random seeds.

A. Discrete Experiments: Atari

For discrete experiments, we apply our OTO algorithm to the agents based on QRDQN and conduct experiments with Atari video games via Gym. As for the offline teacher agents, we choose QRDQN-based CQL agents which are trained with expert-level and random-level offline interaction datasets from RL Unplugged [36]. The results are shown in Figs. 5 and 6, which are trained with expert-level and random-level datasets, respectively.

In ablation studies, we compare the performance of our OTO algorithm with the ones of the knowledge distillation methods that are only with (10) (denoted as AIKD-BASE) and (11) (denoted as AIKD-RKD), respectively. The learning curves in Figs. 5 and 6 indicate that the OTO algorithm can outperform the compared methods with higher speed to become state-of-the-art. We also make comparisons with the online trained benchmarks (denoted as QRDQN in the figures) provided by Tianshou [31]. The results show that the OTO algorithm can accelerate the training procedure by *twice to ten times faster*. Furthermore, in some experiments, such as Qbert and Pacman, the OTO-trained agents can have better performance than the online trained ones whether the dataset in use is expert-level or not. These experiments demonstrate the efficacy of the OTO algorithm.

B. Continuous Experiments: MuJoCo

Next, we apply our OTO algorithm to PPO-based agents with the proposed fusion-policy architecture and conduct the continuous experiments with the MuJoCo physics engine [37]. To illustrate the compatibility of model architecture, we employ pre-trained SAC agents as the teacher agents, instead of using offline RL agents trained with the corresponding datasets. Fig. 7 depicts the conducted experimental results.

In Fig. 7, the OTO algorithm with our designed fusion-policy architecture (denoted as OTO), the knowledge distillation method only with (11) (denoted as AIKD-RKD), and the pure online trained PPO agent are compared together. One can see that the OTO algorithm outperforms in almost all the conducted experiments. Some experiments, such as Half Cheetah and Walker, show that the OTO algorithm can not only speed up training but also enhance the final rewards double to triple. These experiments show the efficacy and the compatibility of the proposed algorithm.

VI. Cross-Environment Experiments

For many cloud-edge collaborative DRL scenarios, the environments of the online student agents and the ones for collecting datasets to train the offline teacher agents are heterogeneous but similar with each other. The reason is that, as the service provider, the cloud platform will try to find the most relevant pre-obtained datasets and teacher model upon the request of the developer of online DRL; however, the environments of the offline model may not exactly match what the online DRL will encounter in the future. Therefore, in this section, we conduct two sets of cross-environment experiments to evaluate the generalizability of our OTO algorithm. We first consider in Section VI-A transferring knowledge from a standard *continuous* control task named Inverted Pendulum (offline teacher in the cloud) to a similar online environment called Pendulum, which requires *discrete* actions chosen by online agents deployed at the edge. We then build in Section VI-B two mobile cell selection environments: one is the 4G LTE scenario where the offline agent was trained, and the other is 5G environment which the online agent will interact with.

A. CartPole Control

We consider a discrete control task, CartPole, and its continuous counterpart, Inverted Pendulum, supported by the Gym [34] environments. To demonstrate the capability of our OTO algorithm on training heterogeneous RL agents simultaneously

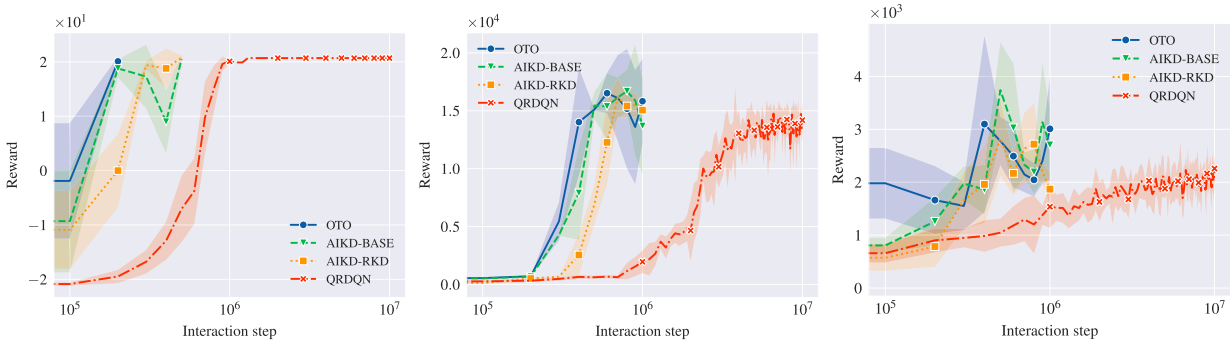


Fig. 5. Atari CQL-QRDQN experiments with expert-level offline datasets.

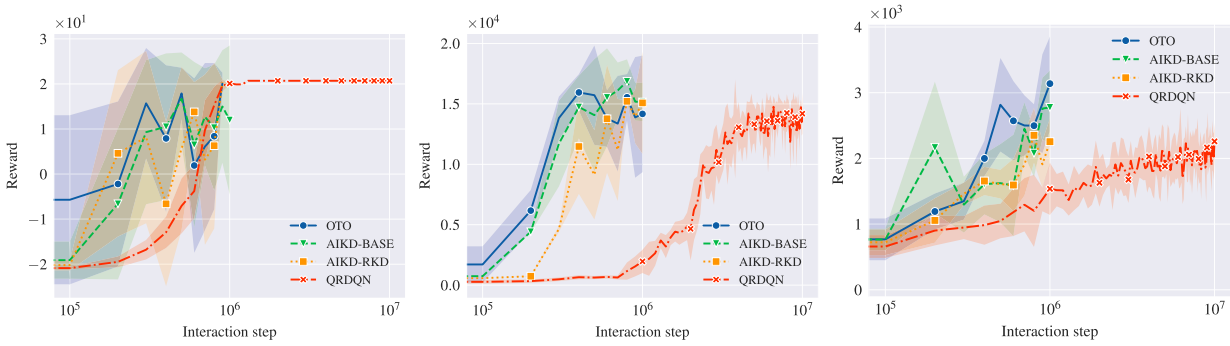


Fig. 6. Atari CQL-QRDQN experiments with random-level offline datasets.

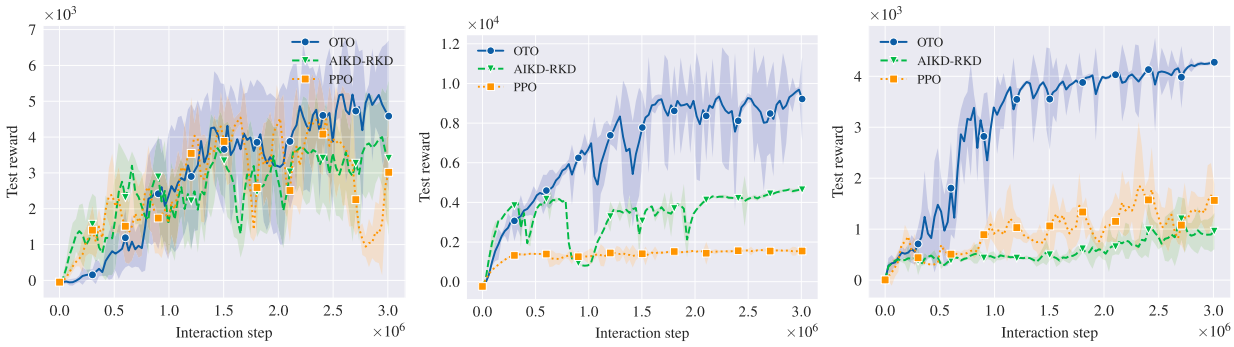


Fig. 7. MuJoCo experiments with PPO-based agents.

in a distributed manner, we employ a continuous SAC agent trained with Inverted Pendulum as the teacher agent, and discrete QRDQN agents as the student agents to train with CartPole. Both environments share the same control method and the same optimization goal, where a cart can be pushed left or right to balance the pole on the top of the cart by applying forces on the cart. However, the definitions of the observation space and the parameters of the environments are heterogeneous. Therefore, additional learnable transfer techniques are needed in the OTO algorithm. There are two variants of CartPole: CartPole-v0 is the environment whose maximum reward and episode length are 200, while CartPole-v1 is the one whose maximum reward and episode length are 500. We use CartPole-v1 for comparative evaluation with the method proposed by [17]. All the results in the tests are averaged over ten seeds.

In Fig. 8, we compare our algorithm with the knowledge distillation method proposed by [17] (denoted as KD in the figure), which utilizes policy distillation and value matching to

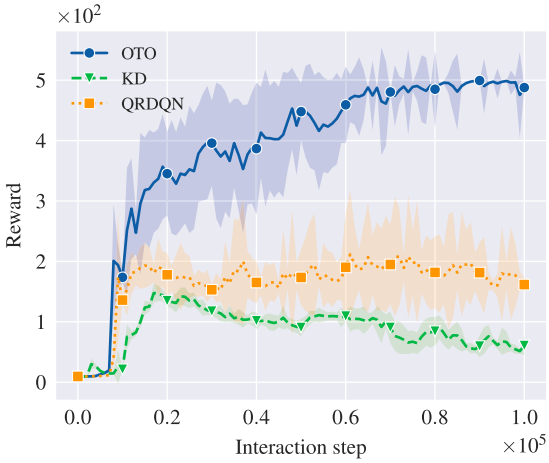


Fig. 8. Performance on CartPole with SAC-QRDQN transfer.

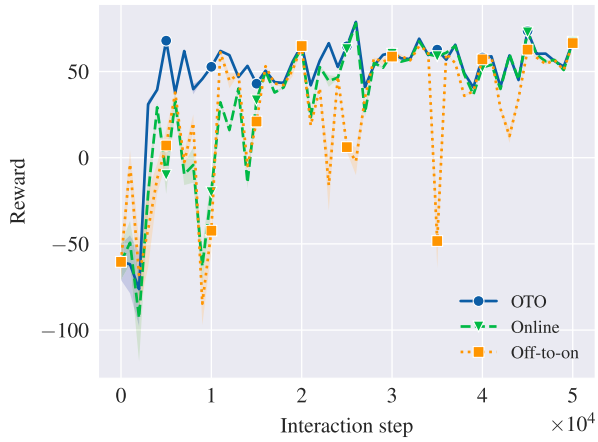


Fig. 9. Cell selection learning curves of QRDQN agents.

transfer the knowledge for online RL agents. Fig. 8 presents that, with the assistance of the inference results from the Inverted Pendulum-trained SAC agent (our teacher model), the agent trained by OTO converges to the optimal policy in a short time, while the KD and the QRDQN baseline algorithms take a long period of time to converge. Specifically, Fig. 8 shows that the OTO algorithm only takes about 90k steps to train an optimal QRDQN agent, while the KD algorithm can be negatively transferred due to the differences in the observation spaces. Therefore, the proposed OTO algorithm is superior in training heterogeneous RL agents with different environments efficiently.

B. Mobile Cell Selection

We next provide a case study in mobile networking. Specifically, cell selection is a very important problem to achieve efficient mobile communications [38]. We leverage the environment mobile-env provided by [39], one can easily build digital twins of mobile communication systems with multiple base stations (BS) and mobile devices.

We conduct the mobile cell selection experiment with the vehicle-to-everything migration scenario from 4G Long Term Evolution (LTE) standard [40] to 5G standard [41]. To simplify the computation, we choose the medium version of mobile-env with one mobile device and seven base stations. For better simulations, the mobility of the device follows the random waypoint movement, where the moving speed of the mobile device in the 5G environment is the six times of the LTE's device moving velocity, indicating the differences between the offline and online environments. The actions of the environment are to toggle the connection between the device and the corresponding base station. The observations are defined as the concatenation of the following.

1) Current connections. This variable indicates to which cells the mobile device is currently connected. The device will hold its connections until it receives connection or disconnection signals.
2) Signal-to-noise ratio (SNR). This normalized variable indicates the BS with the strongest signal. It may be affected by the frequency of the signal and the distance between the device and the BS.
3) Utility. With current connections and SNR, mobile-env can compute the logarithm value of the received data rate and scale to range $[-1, 1]$ as the utility of the mobile device, which is also the step reward of the environment.

With two mobile-env environments of 5G NR 77 standard (we settle 3300 MHz in frequency and 100 MHz in bandwidth) [41] and LTE band 33 standard (we settle 2100 MHz in frequency and 20 MHz in bandwidth) [40], respectively, we conduct comparative experiments with QRDQN agents. For the OTO algorithm, we employ a Stable-Baselines3 [42] PPO agent trained with the LTE environment as the teacher agent and a QRDQN agent trained in the 5G environment as the student agent. We compare the cell selection performance of the OTO algorithm with the QRDQN agent trained with the 5G environment (denoted as online) and the one trained in LTE but tested with 5G (denoted as off-to-on). Fig. 9 depicts the learning curves of QRDQN agents, where the results are averaged over one hundred random seeds. One can see that the OTO algorithm has the highest and stablest performance among the compared agents, with the help of the LTE-trained PPO agent.

We also conduct extensive experiments with the following baselines.

1) Fixed. A rule-based selection method where the device connects to a fixed BS.
2) Random. A selection method where the device connects to a randomly chosen connectable BS.
3) Full. A rule-based selection method where the device connects to all the connectable base stations.
4) LTE [43]. A greedy method which selects the strongest BS in SNR to connect.
5) Clustering [44]. A static clustering approach which groups all cells into fixed-size clusters and then selects the strongest one to connect.

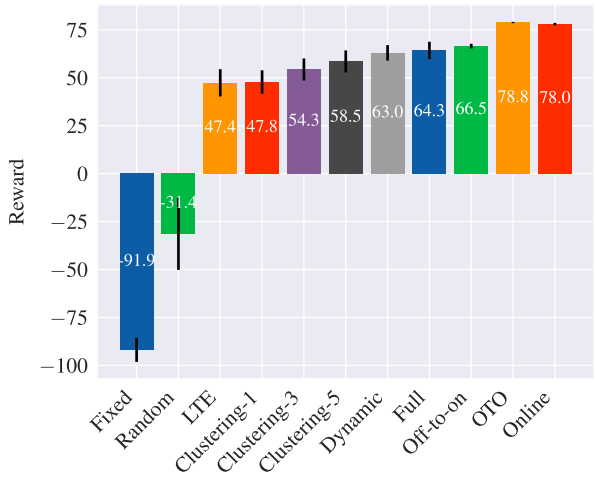


Fig. 10. Mobile cell selection performance benchmarks.

6) Dynamic [45]. A heuristic method for device-centric multiple cell selection, which is a generalization of the LTE greedy method [43].

Fig. 10 shows that our OTO-trained agent outperforms the benchmarks of the compared mobile cell selection algorithms in the 5G environment. OTO not only achieves the highest reward but also and stablest performance on reward during one hundred time steps in the experiments. Compared with rule-based, heuristic, and DRL-based methods, the proposed OTO algorithm is superior in the mobile cell selection task using offline DRL trained in the LTE environments to boost the online agent in a 5G scenario with the minimum communication between these two agents, indicating its generalizability in cross-environment cases and viability for cloud-edge collaborative mobile applications.

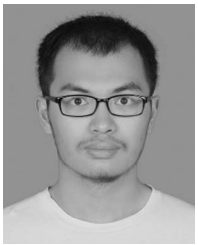
VII. Conclusion

In this paper, we propose a novel Offline-Transfer-Online RL training method for cloud-edge collaborative RL, which bridges online and offline RL by taking advantage of our algorithm-independent knowledge distillation design through an agent-environment interaction interface. Our proposed AIKD loss function integrates the ideas of conservative learning and transferring the relationship patterns between selected actions of the offline policy to online agents. This module can be used in different types of RL algorithms without the need to share agents' buffer data. Furthermore, we implement the proposed methods as a distributed RL system. Extensive experiments demonstrate that OTO framework has great advances in speeding up the model training and achieving the highest reward scores among the state-of-the-art RL algorithms, as well as reducing the communication overhead compared to classic offline-to-online RL methods.

References

[1] Huawei, "5G spectrum public policy position," *Public Policy Position*, 2016.
[2] Samsung Research, "6G: The next hyper connected experience for all," 2020.
[3] E. Peltonen et al., "6G white paper on edge intelligence," 2020, *arXiv: 2004.14850*.
[4] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y. J. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84–90, Aug. 2019.
[5] B. Rong, "6G: The next horizon: From connected people and things to connected intelligence," *IEEE Wireless Commun.*, vol. 28, no. 5, pp. 8–8, Oct. 2021.
[6] I. J. Goodfellow, Y. Bengio, and A. C. Courville, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.
[7] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 285–286, Jan. 2005.
[8] N. C. Luong et al., "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, Fourth Quarter, 2019.
[9] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.
[10] R. Minerva, G. M. Lee, and N. Crespi, "Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models," in *Proc. IEEE*, vol. 108, no. 10, pp. 1785–1824, Oct. 2020.
[11] H. X. Nguyen, R. Trestian, D. To, and M. Tatipamula, "Digital twin for 5G and beyond," *IEEE Commun. Mag.*, vol. 59, no. 2, pp. 10–15, Feb. 2021.
[12] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, pp. 350–354, 2019.
[13] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," 2020, *arXiv: 2006.04779*.
[14] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline deep reinforcement learning," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 104–114.
[15] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin, "Offline-to-online reinforcement learning via balanced replay and pessimistic Q-ensemble," in *Proc. Conf. Robot Learn.*, 2021, pp. 1702–1712.
[16] T. Yang et al., "Learning when to transfer among agents: An efficient multiagent transfer learning framework," 2020, *arXiv: 2002.08030*.
[17] S. Wadhwania, D.-K. Kim, S. Omidshafiei, and J. P. How, "Policy distillation and value matching in multiagent reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 8193–8200.
[18] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3962–3971.
[19] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent IoT applications: A cloud-edge based framework," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 35–44, 2020.
[20] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun.*, Dublin, Ireland, 2020, pp. 1–6.
[21] H. Cao, M. Theile, F. G. Wyrwal, and M. Caccamo, "Cloud-edge training architecture for sim-to-real deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Kyoto, Japan, 2022, pp. 9363–9370.
[22] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, pp. 279–292, 2004.
[23] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2892–2901.
[24] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," 2015, *arXiv:1502.05477*.
[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv: 1707.06347*.
[26] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[27] R. S. Sutton, D. A. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1999, pp. 1057–1063.
[28] H. Zheng, X. Luo, P. Wei, X. Song, D. Li, and J. Jiang, "Adaptive policy learning for offline-to-online reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 11372–11380. [Online]. Available: https://www.microsoft.com/en-us/research/publication/adaptive-policy-learning-for-offline-to-online-reinforcement-learning/
[29] S. Niu, Y. Liu, J. Wang, and H. H. Song, "A decade survey of transfer learning (2010–2020)," *IEEE Trans. Artif. Intell.*, vol. 1, no. 2, pp. 151–166, Oct. 2020.

[30] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," 2021, *arXiv: 2006.05525*.
[31] J. Weng et al., "Tianshou: A highly modularized deep reinforcement learning library," 2021, *arXiv:2107.14171*.
[32] P. Moritz et al., "Ray: A distributed framework for emerging AI applications," 2018, *arXiv: 1712.05889*.
[33] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, Art. no. 721.
[34] G. Brockman et al., "OpenAI Gym," 2016, *arXiv:1606.01540*.
[35] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents (extended abstract)," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 4148–4152.
[36] C. Gulcehre et al., "RL unplugged: A suite of benchmarks for offline reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 7248–7259.
[37] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.
[38] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for stochastic computation offloading in digital twin networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4968–4977, Jul. 2021.
[39] S. Schneider, S. Werner, R. Khalili, A. Hecker, and H. Karl, "Mobile-env: An open platform for reinforcement learning in wireless mobile networks," in *Proc. IEEE/IFIP Netw. Operations Manage. Symp.*, 2022, pp. 1–3.
[40] ETSI, "LTE; evolved universal terrestrial radio access (E-UTRA); user equipment (UE) radio transmission and reception (3GPP TS 36.101 version 14.9.0 release 14)," *Eur. Telecommun.*, 2019.
[41] 3GPP TS 38.101, "5G; NR; user equipment (UE) radio transmission and reception," 2018.
[42] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *J. Mach. Learn. Res.*, vol. 22, pp. 268:1–268:8, 2021.
[43] 3GPP TR 36.814 V9.0.0, "Evolved universal terrestrial radio access (E-UTRA); further advancements for E-UTRA physical layer aspects (release 9)," *Proc. 3rd Gener. Partnership Project*, vol. 9, no. 3, pp. 1–8, 2010.
[44] P. Marsch and G. Fettweis, "Static clustering for cooperative multi-point (CoMP) in mobile communications," in *Proc. IEEE Int. Conf. Commun.*, 2011, pp. 1–6.
[45] A. Beylerian and T. Ohtsuki, "Multi-point fairness in resource allocation for C-RAN downlink CoMP transmission," *Eurasip J. Wireless Commun. Netw.*, vol. 2016, 2016, Art. no. 12.

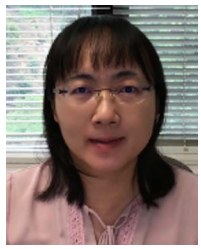


Tianyu Zeng received the bachelor's degree in 2021 from Sun Yat-sen University, Guangzhou, China, where he is currently working toward the master's degree with the School of Computer Science and Engineering. His research interests include cloud and edge computing, distributed machine learning systems, and reinforcement learning.

Xiaoxi Zhang (Member, IEEE) received the BE degree from the Huazhong University of Science and Technology, in 2013, and the PhD degree from the University of Hong Kong, in 2017. She is currently an associate professor with the School of Computer Science and Engineering, Sun Yat-sen University. She was a postdoctoral researcher with Carnegie Mellon University from 2017 to 2020. Her research interests include cloud and edge computing, optimization, online algorithms, online learning, and reinforcement learning.

Jingpu Duan (Member, IEEE) received the BE degree from the Huazhong University of Science and Technology, Wuhan, China, in 2013, and the PhD degree from the University of Hong Kong, Hong Kong, China, in 2018. He is currently a research assistant professor with the Institute of Future Networks, Southern University of Science and Technology, Shenzhen, China. He also works with the Department of Communications, Pengcheng Laboratory, Shenzhen, China. His research interests include designing and implementing high-performance networking systems.

Chao Yu received the PhD degree in computer science from the University of Wollongong, Australia, in 2014. He is currently an associate professor with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. He has published more than 100 articles in prestigious journals, such as the *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Cybernetics*, and *IEEE Transactions on Vehicular Technology*. His research interests include multi-agent systems and reinforcement learning.



Chuan Wu (Senior Member, IEEE) received the PhD degree from the Department of Electrical and Computer Engineering, University of Toronto, Canada, in 2008. Since 2008, she has been with the Department of Computer Science, University of Hong Kong, where she is currently a professor. Her current research interests include the areas of cloud computing, distributed machine learning systems, network function virtualization, and intelligent elderly care technologies.

Xu Chen (Senior Member, IEEE) received the PhD degree in information engineering from the Chinese University of Hong Kong in 2012. He is a full professor with Sun Yat-sen University, Guangzhou, China, and the vice director of National and Local Joint Engineering Laboratory. He worked as a postdoctoral research associate with Arizona State University, Tempe, USA from 2012 to 2014, and a humboldt scholar fellow with the Institute of Computer Science, University of Goettingen, Germany from 2014 to 2016. He received the prestigious Humboldt research fellowship awarded by Alexander von Humboldt Foundation of Germany, 2014 Hong Kong Young scientist Runner-up Award, 2016 Thousand Talents Plan Award for Young professionals of China, 2017 IEEE Communication Society Asia-Pacific Outstanding Young Researcher Award, 2017 IEEE ComSoc Young professional Best Paper Award, Honorable Mention Award of 2010 IEEE international conference on Intelligence and Security Informatics (ISI), Best Paper Runner-up Award of 2014 IEEE International Conference on Computer Communications (INFOCOM), and Best Paper Award of 2017 IEEE Intranational Conference on Communications (ICC). He is currently an area editor of *IEEE Open Journal of the Communications Society*, an associate editor of the *IEEE Transactions Wireless Communications*, *IEEE Internet of Things Journal*, *IEEE Transactions on Vehicular Technology*, and *IEEE Journal on Selected Areas in Communications* (JSAC) Series on Network Softwarization and Enablers.