

# A Survey of Deep Reinforcement Learning in Recommender Systems: A Systematic Review and Future Directions

XIAOCONG CHEN, University of New South Wales, Australia

LINA YAO, University of New South Wales, Australia

JULIAN MCAULEY, University of California, San Diego, USA

GUANGLIN ZHOU, University of New South Wales, Australia

XIANZHI WANG, University of Technology Sydney, Australia

In light of the emergence of deep reinforcement learning (DRL) in recommender systems research and several fruitful results in recent years, this survey aims to provide a timely and comprehensive overview of the recent trends of deep reinforcement learning in recommender systems. We start with the motivation of applying DRL in recommender systems. Then, we provide a taxonomy of current DRL-based recommender systems and a summary of existing methods. We discuss emerging topics and open issues, and provide our perspective on advancing the domain. This survey serves as introductory material for readers from academia and industry into the topic and identifies notable opportunities for further research.

CCS Concepts: • **Information systems** → **Recommender systems**; • **Computing methodologies** → **Reinforcement learning**; **Neural networks**.

Additional Key Words and Phrases: Deep Reinforcement Learning, Deep Learning, recommender systems

## ACM Reference Format:

Xiaocong Chen, Lina Yao, Julian Mcauley, Guanglin Zhou, and Xianzhi Wang. 2021. A Survey of Deep Reinforcement Learning in Recommender Systems: A Systematic Review and Future Directions. *J. ACM* 37, 4, Article 111 (September 2021), 32 pages. <https://doi.org/10.1145/1122445.1122456>

## 1 INTRODUCTION

Recent years have seen significant progress in recommendation techniques, from traditional recommendation techniques, e.g., collaborative filtering, content-based recommendation and matrix factorization [62], to deep learning based techniques. In particular, deep learning show strong advantages in solving complex tasks and dealing with complex data, due to its capability to capture non-linear user-item relationships and deal with various types of data sources such as images and text. It has thus been increasingly used in recommender systems. **Deep learning-based recommender systems have limitations in capturing interest dynamics [17, 115] due to distribution shift, i.e., the training phase is based on an existing dataset which may not reflect real user preferences that undergo rapid change. In contrast, deep reinforcement learning (DRL) aims to train an agent that can learn from interaction trajectories provided by the environment by combining the power of**

Authors' addresses: Xiaocong Chen, [xiaocong.chen@unsw.edu.au](mailto:xiaocong.chen@unsw.edu.au), University of New South Wales, Sydney, NSW, Australia; Lina Yao, [lina.yao@unsw.edu.au](mailto:lina.yao@unsw.edu.au), University of New South Wales, Sydney, NSW, Australia; Julian Mcauley, [jmcauley@eng.ucsd.edu](mailto:jmcauley@eng.ucsd.edu), University of California, San Diego, CA, USA; Guanglin Zhou, [guanglin.zhou@unsw.edu.au](mailto:guanglin.zhou@unsw.edu.au), University of New South Wales, Sydney, NSW, Australia; Xianzhi Wang, [xianzhi.wang@uts.edu.au](mailto:xianzhi.wang@uts.edu.au), University of Technology Sydney, Sydney, NSW, Australia.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2021 Association for Computing Machinery.

0004-5411/2021/9-ART111 \$15.00

<https://doi.org/10.1145/1122445.1122456>

deep learning and reinforcement learning. Since an agent in DRL can actively learn from users' real-time feedback to infer dynamic user preferences, DRL is especially suitable for learning from interactions, such as human-robot collaboration; it has also driven significant advances in a range of interactive applications ranging from video games, Alpha Go to autonomous driving [3]. In light of the significance and recent progresses in DRL for recommender systems, we aim to timely summarize and comment on DRL-based recommendation systems in this survey.

A recent survey on reinforcement learning based recommender systems [2] provides a general review about reinforcement learning in recommender systems without a sophisticated investigation of the growing area of deep reinforcement learning. Our survey distinguishes itself in providing a systematic and comprehensive overview of existing methods in DRL-based recommender systems, along with a discussion of emerging topics, open issues, and future directions. This survey introduces researchers, practitioners and educators into this topic and fostering an understanding of the key techniques in the area.

The main contributions of this survey include the following:

- We provide an up-to-date comprehensive review of deep reinforcement learning in recommender systems, with state of the art techniques and pointers to core references. To the best of our knowledge, this is the first comprehensive survey in deep reinforcement learning based recommender systems.
- We present a taxonomy of the literature of deep reinforcement learning in recommender systems. Along with the outlined taxonomy and literature overview, we discuss the benefits, drawbacks and give suggestions for future research directions.
- We shed light on emerging topics and open issues for DRL-based recommender systems. We also point out future directions that could be crucial for advancing DRL-based recommender systems.

The remainder of this survey is organized as follows: Section 2 provides an overview of recommender systems, DRL and their integration. Section 3 provides a literature review with a taxonomy and classification mechanism. Section 4 reviews emerging topics, and Section 5 points out open questions. Finally, Section 6 provides a few promising future directions for further advances in this domain.

## 2 BACKGROUND

In this section, we introduce key concepts related to dynamic recommender systems (RS) and deep reinforcement learning (DRL), and motivate the introduction of DRL to dynamic recommender systems.

### 2.1 Why Deep Reinforcement Learning for Recommendation?

Recommender systems require coping with *dynamic* environments by estimating rapidly changing users' preferences and proactively recommending items to users. Let  $\mathcal{U}$  be a set of users of cardinality  $|\mathcal{U}|$  and  $\mathcal{I}$  be a set of items of cardinality  $|\mathcal{I}|$ . For each user  $u \in \mathcal{U}$ , we observe a sequence of user actions  $\mathbb{X}^u = [x_1^u, x_2^u, \dots, x_{T_u}^u]$  with item  $x_t^u \in \mathcal{I}$ , i.e., each event in a user sequence comes from the item set. We refer to a user making a decision as an interaction with an item. Suppose the feedback (e.g., ratings or clicking behavior) provided by users is  $\mathcal{F}$ , then a dynamic recommender system maintains the corresponding recommendation policy  $\pi_i^u$ , which will be updated systematically based on the feedback  $f_i^u \in \mathcal{F}$  received during the interaction for item  $i \in \mathcal{I}$  at the timestamp  $t$ .

The marriage of deep learning and reinforcement learning has fueled breakthroughs in recommender systems. DRL-based RS consists of a pipeline with three building blocks: environment construction, state representation and recommendation policy learning. Environment construction

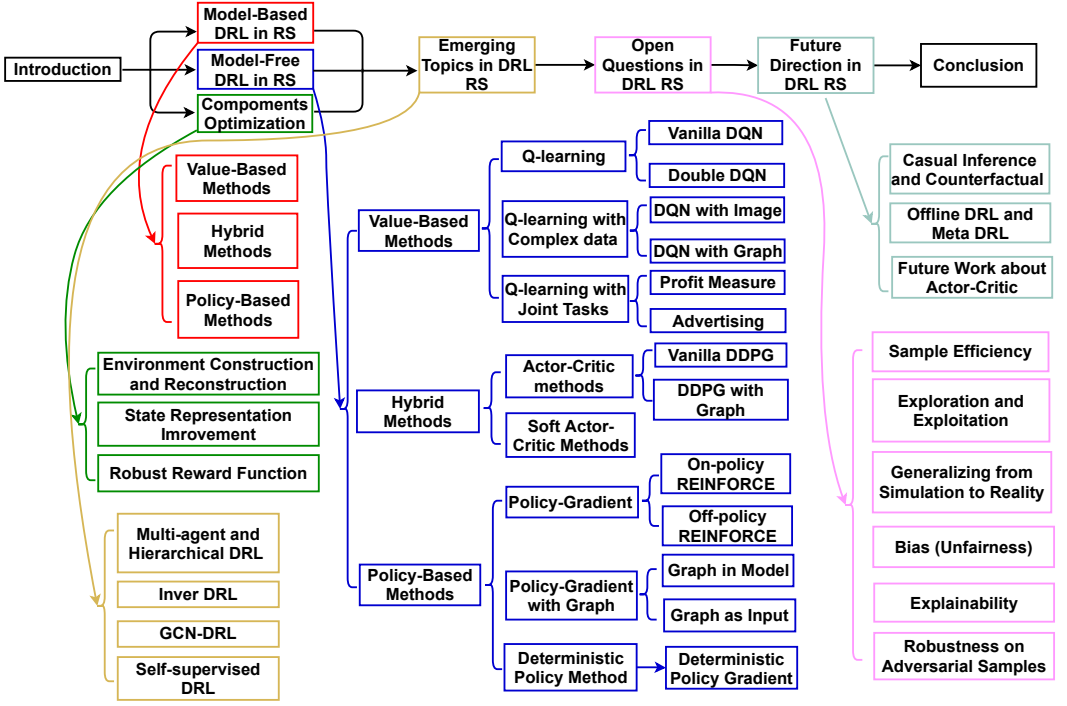


Fig. 1. Taxonomy of Deep Reinforcement Learning based Recommender Systems in this survey

aims to build an environment based on a set of users' historical behaviors. State representation is provided by the environment containing certain user information including historical behavior, demographic data (etc.). Recommendation policy learning is the key component to understand and predict users' future behavior. DL-based RS receives user feedback (e.g., ratings or clicks) to reflect users' interests and update the recommender, while DRL-based RS receives the reward provided by the environment to update the policy. The reward provided by the environment is a pre-defined function containing several factors. The detailed process of DL based RS and DRL-based RS mapping can be found in Figure 3.

## 2.2 Preliminaries of Deep Reinforcement Learning

The typical defining feature of DRL is to use the deep learning to approximate reinforcement learning's value function and solve high-dimensional Markov Decision Processes (MDPs) [3]. Formally, a MDP can be represented as a tuple  $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ . The agent chooses an action  $a_t \in \mathcal{A}$  according to the policy  $\pi_t(s_t)$  at state  $s_t \in \mathcal{S}$ . The environment receives the action and produces a reward  $r_{t+1} \in \mathcal{R}$  and transfers the reward into the next state  $s_{t+1}$  according to the transition probability  $P(s_{t+1}|s_t, a_t) \in \mathcal{P}$ . The transition probability  $\mathcal{P}$  is unknown beforehand in DRL. Such a process continues until the agent reaches the terminal state or exceeds a pre-defined maximum time step. The overall objective is to maximize the expected discounted cumulative reward,

$$\mathbb{E}_{\pi}[r_t] = \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \right] \quad (1)$$

where  $\gamma \in [0, 1]$  is the discount factor that balances the future reward and the immediate reward.

Deep reinforcement learning can be divided into two categories: *model-based* and *model-free* methods (a detailed taxonomy can be found in Figure 2). The major difference between the two is whether the agent can learn a model of the environment. Model-based methods aim to estimate the transition function and reward function, while model-free methods aim to estimate the value function or policy from experience. In model-based methods, the agent accesses the environment and plans ahead while model-free methods gain sample efficiency from using models which are more extensively developed and tested than model-based methods in recent literature [3].

Deep reinforcement learning approaches are divided into three streams: *value-based*, *policy-based* and *hybrid* methods. In value-based methods, the agent updates the value function to learn a policy; policy-based methods learn the policy directly; and hybrid methods combine value-based and policy-based methods called *actor-critic* methods. Actor-critic contains two different networks where an actor network uses a policy-based method and the critic uses a value-based method to evaluate the policy learned by the agent.

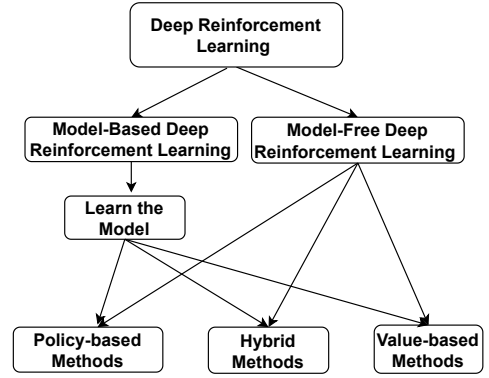


Fig. 2. Taxonomy of Deep Reinforcement Learning in Recommender Systems

Table 1. Notations

Notations	Name	Notations	Name	Notes
$Q(\cdot)$	Q-Value Function	$s$	State	users' preference
$V(\cdot)$	Value Function	$a$	Action	Recommended item(s)
$\gamma$	Discount Factor	$\pi, \mu(\cdot)$	Policy	Recommendation policy
$\mathbb{E}$	Expected Value	$r(\cdot, \cdot)$	Reward	users' click behavior
$\theta$	Model Parameter	$\alpha$	constant $\in [0, 1]$	-
$p(\cdot)$	Transition Probability	$\tau$	Sampled Trajectory	A tuple $(s_t, a_t, s'_t, r_t)$

Deep reinforcement learning can be divided into *on-policy* and *off-policy* methods. In off-policy methods, the behavior policy  $\pi_b$  is used for exploration while the target policy  $\pi$  is used for decision-making. For on-policy methods, the behavior policy is the same as the target policy.

**Q-learning** [101] is an off-policy value-based learning scheme for finding a greedy target policy:

$$\pi(s) = \arg \max_a Q_\pi(s, a) \quad (2)$$

where  $Q_u(s, a)$  denotes the  $Q$ -value and is used in a small discrete action space. For a deterministic policy, the  $Q$  value can be calculated as follows

$$Q(s_t, a_t) = \mathbb{E}_{\tau \sim \pi} [r(s_t, a_t) + \gamma Q(s'_t, a'_t)]. \quad (3)$$

Deep Q learning (DQN) [67] uses deep learning to approximate a non-linear  $Q$  function parameterized by  $\theta_q$ :  $Q_{\theta_q}(s, a)$ . DQN designs a network  $Q_{\theta_q}$  that is asynchronously updated by minimizing the MSE:

$$\mathcal{L}(\theta_q) = \mathbb{E}_{\tau \sim \pi} \left[ Q_{\theta_q}(s_t, a_t) - (r(s_t, a_t) + \gamma Q_{\theta_q}(s'_t, a'_t)) \right]^2 \quad (4)$$

where  $\tau$  is the sampled trajectory containing  $(s, a, s', r(s, a))$ . In particular,  $s'_t$  and  $a'_t$  come from the behavior policy  $\pi_b$  while  $s, a$  comes from the target policy  $\pi$ . It is worth mentioning that the value function  $V_\pi(s)$  represents the expected return.  $V_\pi(s)$  is used to evaluate the goodness of the state while  $Q_\pi(s_t, a_t)$  is used to evaluate the action.  $V_\pi(s)$  can be defined as

$$V_\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^T \gamma^t r(s, a) | s_0 = s \right]. \quad (5)$$

$V_\pi(\cdot)$  and  $Q_\pi(\cdot)$  have the following relationship:

$$V_\pi(s) = \mathbb{E}_{a \sim \pi} [Q_\pi(s, a)]. \quad (6)$$

The value function is updated using the following rule with the Temporal Difference (TD) method,

$$V_\pi(s_t) \leftarrow V_\pi(s_t) + \alpha \underbrace{[r(s'_t, a'_t) + \gamma V_\pi(s'_t) - V_\pi(s_t)]}_{\text{TD-error}} \quad (7)$$

where  $\alpha$  is a constant.

**Policy gradient** [102] is an on-policy policy-based method which can handle high-dimensional or continuous actions which cannot be easily handled by Q-learning. Policy gradient aims to find the parameter  $\theta$  of  $\pi_\theta$  to maximize the accumulated reward. To this end, it maximizes the expected return from the start state:

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} [r(\tau)] = \int \pi_\theta(\tau) r(\tau) d\tau \quad (8)$$

where  $\pi_\theta(\tau)$  is the probability of the occurrence of  $\tau$ . Policy gradient learns the parameter  $\theta$  by the gradient  $\nabla_\theta J(\pi_\theta)$  as defined below:

$$\begin{aligned} \nabla_\theta J(\pi_\theta) &= \int \pi_\theta(\tau) r(\tau) d\tau = \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim d_{\pi_\theta}} \left[ \sum_{t=1}^T r(s_t, a_t) \sum_{t=1}^T \nabla_\theta \log \pi_\theta(s_t, a_t) \right]. \end{aligned} \quad (9)$$

The above derivations contain the following substitution,

$$\pi_\theta(\tau) = p(s_1) \prod_{t=1}^T \pi_\theta(s_t, a_t) p(s_{t+1} | s_t, a_t) \quad (10)$$

where  $p(\cdot)$  are independent from the policy parameter  $\theta$ , which is omitted during the derivation. Monte-Carlo sampling has been used by previous policy gradient algorithm (e.g., REINFORCE) for  $\tau \sim d_{\pi_\theta}$ .

**Actor-critic networks** combine the advantages from Q-learning and policy gradient. They can be either on-policy [49] or off-policy [21]. An actor-critic network consists of two components: i) *an actor*, which optimizes the policy  $\pi_\theta$  under the guidance of  $\nabla_\theta J(\pi_\theta)$ ; and ii) *a critic*, which evaluates the learned policy  $\pi_\theta$  by using  $Q_{\theta_q}(s, a)$ . The overall gradient is represented as follows:

$$\mathbb{E}_{s \sim d_{\pi_\theta}} [Q_{\theta_q}(s, a) \nabla_\theta \log \pi_\theta(s, a)]. \quad (11)$$

When dealing with off-policy learning, the value function for  $\pi_\theta(a|s)$  can be further determined by deterministic policy gradient (DPG) as shown below:

$$\mathbb{E}_{s \sim d_{\pi_\theta}} [\nabla_a Q_{\theta_q}(s, a) |_{a=\pi_\theta(s)} \nabla_\theta \pi_\theta(s, a)]. \quad (12)$$

While traditional policy gradient calculates the integral for both the state space  $\mathcal{S}$  and the action space  $\mathcal{A}$ , DPG only requires computing the integral to the state space  $\mathcal{S}$ . Given a state  $s \in \mathcal{S}$ , there

will be only one corresponding action  $a \in \mathcal{A} : \mu_\theta(s) = a$  using DPG. Specifically, deep Deterministic Policy Gradients (DDPG) is an algorithm that combines techniques from DQN and DPG. DDPG contains four different neural networks: Q Network  $Q$ , policy network, target Q network  $Q^{tar}$ , and target policy network. It uses the target network for both the Q Network  $Q$  and policy network  $\mu$  to ensure stability during training. Assume  $\theta_q, \theta_\pi, \theta_{q'},$  and  $\theta_{\pi'}$  are parameters of the above networks; then DDPG soft-updates the parameters for the target network [56]:

$$\text{Actor: } \theta_{\pi'} \leftarrow \alpha \theta_\pi + (1 - \alpha) \theta_{\pi'}, \text{ Critic: } \theta_{q'} \leftarrow \alpha \theta_q + (1 - \alpha) \theta_{q'} \quad (13)$$

### 2.3 DRL meets RS: Problem Formulation

DRL is normally formulated as a Markov Decision Process (MDP). Given a set of users  $\mathcal{U} = \{u, u_1, u_2, u_3, \dots\}$ , a set of items  $\mathcal{I} = \{i, i_1, i_2, i_3, \dots\}$ , the system first recommends item  $i$  to user  $u$  and then gets feedback  $f_i^u$ . The system aims to incorporate the feedback to improve future recommendations and needs to determine an optimal policy  $\pi^*$  regarding which item to recommend to the user to achieve positive feedback. The MDP modelling of the problem treats the user as the environment and the system as the agent. The key components of the MDP in DRL-based RS include the following:

- State  $\mathcal{S}$ : A state  $S_t \in \mathcal{S}$  is determined by both users' information and the recent  $l$  items in which the user was interested before time  $t$ .
- Action  $\mathcal{A}$ : An action  $a_t \in \mathcal{A}$  represents users' dynamic preference at time  $t$  as predicted by the agent.  $\mathcal{A}$  represents the whole set of (potentially millions of) candidate items.
- Transition Probability  $\mathcal{P}$ : The transition probability  $p(s_{t+1}|s_t, a_t)$  is defined as the probability of state transition from  $s_t$  to  $s_{t+1}$  when action  $a_t$  is executed by the recommendation agent. In a recommender system, the transition probability refers to users' behavior probability.  $\mathcal{P}$  is only used in model-based methods.
- Reward  $\mathcal{R}$ : Once the agent chooses a suitable action  $a_t$  based on the current state  $S_t$  at time  $t$ , the user will receive the item recommended by the agent. Users' feedback on the recommended item accounts for the reward  $r(S_t, a_t)$ . The feedback is used to improve the policy  $\pi$  learned by the recommendation agent.
- Discount Factor  $\gamma$ : The discount factor  $\gamma \in [0, 1]$  is used to balance between future and immediate rewards—the agent focuses only on the immediate reward when  $\gamma = 0$  and takes into account all the (immediate and future) rewards otherwise.

The DRL-based recommendation problem can be defined by using MDP as follows. *Given the historical MDP, i.e.,  $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ , the goal is to find a set of recommendation policies  $(\{\pi\} : \mathcal{S} \rightarrow \mathcal{A})$  that maximizes the cumulative reward during interaction with users.*

**PROBLEM FORMULATION.** *Given an environment that contains all items  $\mathcal{I}$ , when user  $u \in \mathcal{U}$  interacts with the system, an initial state  $s$  is sampled from the environment which contains a list of candidate items and users' historical data. The DRL agent needs to work out a recommendation policy  $\pi$  based on the state  $s$  and produces the corresponding recommended item list  $a$ . The user will provide feedback on the list which is normally represented as click or not click. The DRL agent will then utilize the feedback to improve the recommendation policy and move to the next interaction episode.*

## 3 DEEP REINFORCEMENT LEARNING IN RECOMMENDER SYSTEMS

DRL-based RS has some unique challenges such as state construction, reward estimation and environment simulation. We categorize the existing work of DRL-based recommendation into model-based and model-free methods (the taxonomy is shown in Figure 2).

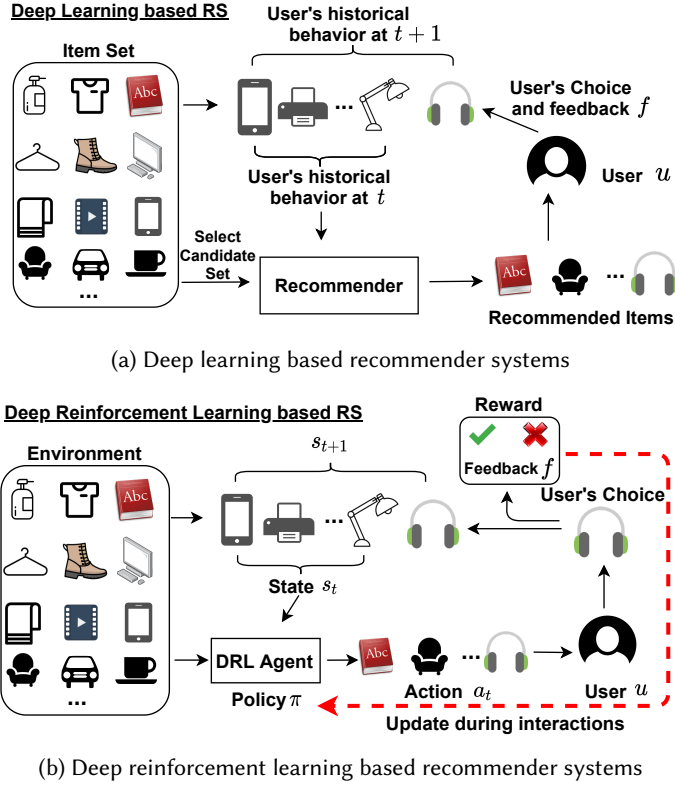


Fig. 3. Difference between deep learning based RS and DRL-based RS. Deep learning based RSs may only update the recommendation policy during the training stage. They often require re-training, which is computationally inefficient, when users' interests change significantly. DRL-based RS will update the recommendation policy time over time as new rewards are received.

### 3.1 Model-based Deep Reinforcement Learning based Methods

Model-based methods assume an expected reward or action available for the next step to help the agent update the policy.

Table 2. List of publications in model-based DRL-based RS

Method	Work
Value-based	[18, 96, 119, 135]
Policy-based	[4, 38]
Hybrid	[125]

*Policy-based methods.* IRecGAN [4] is a model-based method that adopts generative adversarial training to improve the robustness of policy learning. It can reduce the cost of interaction for RS by using offline data instead of the simulated environment. IRecGAN employs a generative adversarial network [32] to generate user data based on the offline dataset. It trains a recommendation agent using a policy gradient-based DRL method called REINFORCE. The agent aims to learn a policy

based on the following gradient,

$$\mathbb{E}_{\tau \sim \{g, data\}} \left[ \sum_{t=0}^T \sum_{t'=t}^T \gamma^{t'-t} q_D(\tau_{0:t}^n) r_t \nabla_{\theta_a} (c_t \in \pi_{\theta_a}(s_t)) \right], q_D(\tau_{0:t}^n) = \frac{1}{N} \sum_{n=1}^N D(\tau_{0:T}^n), \tau_{0:T}^n \in MC^{\mathcal{U}}(N) \quad (14)$$

where the  $MC^{\mathcal{U}}(N)$  represents the sampled  $N$  sequences from the interaction between  $\mathcal{U}$  and the agent using the Monte-Carlo tree search algorithm,  $D$  is the discriminator,  $T$  is the length of  $\tau$ ,  $g$  represents the offline data, and  $data$  represents the generated data.

Hong et al. [38] propose NRSS for personalized music recommendation. NRSS uses wireless sensing data to learn users' current preferences. NRSS considers three different types of feedback: score, option, and wireless sensing data. Because multiple factors are considered as the reward, NRSS designs a reward model which consists of users' preference reward  $r_p$  and a novel transition reward  $r_{trans}$  which are parameterized by  $\theta_{r_p}$  and  $\theta_{r_{trans}}$ . The goal for NRSS is to find the optimal parameters  $\theta_{r_p}$  and  $\theta_{r_{trans}}$  by using the Monte-Carlo tree search thus improving recommendation performance. However, wireless sensing feedback lacks generalization ability as it is only available for certain tasks or scenarios, making it hard to determine dynamic user interest.

*Value-based methods.* Prior to Q-learning, value iteration is a more traditional value-based reinforcement learning algorithm that focuses on the iteration of the value function. Gradient Value Iteration (GVI) [119] is proposed to improve the traditional value iteration algorithm by utilizing the transition probability and a multi-agent setting to predict chronological author collaborations. It introduces a new parameter named 'status' to reflect the amount of knowledge that the agent needs to learn from this state. The policy is updated only when the distance between the new status and the old status is lower than a pre-defined threshold. However, value iteration requires the transition probability, which is hard to obtain in most cases. Hence, Q-learning and its variants are widely used in DRL-based RS. Cascading DQN (CDQN) with a generative user model [18] is proposed to deal with the environment with unknown reward and environment dynamics. The generative user model adopts GANs to generate a user model based on an offline dataset. Different from previous work, it will generate the reward function for each user to explain the users' behavior. The user model can be written as,

$$\arg \max_{\phi \in \Delta^{k-1}} \mathbb{E}_{\phi} [r(s_t, a_t)] - R(\phi) / \eta \quad (15)$$

where  $\Delta^{k-1}$  is the probability simplex,  $R(\phi)$  is the regularization term for exploration and  $\eta$  is a constant.

Pseudo Dyna-Q (PDQ) [135] points out that Monte-Carlo tree search may lead to an extremely large action space and an unbounded importance weight of training samples. Hence, a world model is proposed to reduce the instability of convergence and high computation cost for interacting with users by imitating the offline dataset. With the world model, the agent will interact with the learned world model instead of the environment to improve the sample efficiency and convergence stability. The world model learning process introduced in PDQ can be described as finding the parameter  $\theta_M$ ,

$$\arg \min_{\theta_M} \mathbb{E}_{\xi \in P^{\pi_{\xi}}} \left[ \sum_t \gamma^t \prod_{j=0}^t \frac{\pi(s_j, a_j)}{\pi_b(s_j, a_j)} \Delta_t(\theta_M) \right] \quad (16)$$

where  $\xi$  is generated by the logged policy  $\pi_b$ ,  $\prod_{j=0}^t \frac{\pi(s_j, a_j)}{\pi_b(s_j, a_j)}$  is the ratio used for importance sampling and  $\Delta$  is the difference between the reward in the world model and real reward. Furthermore,



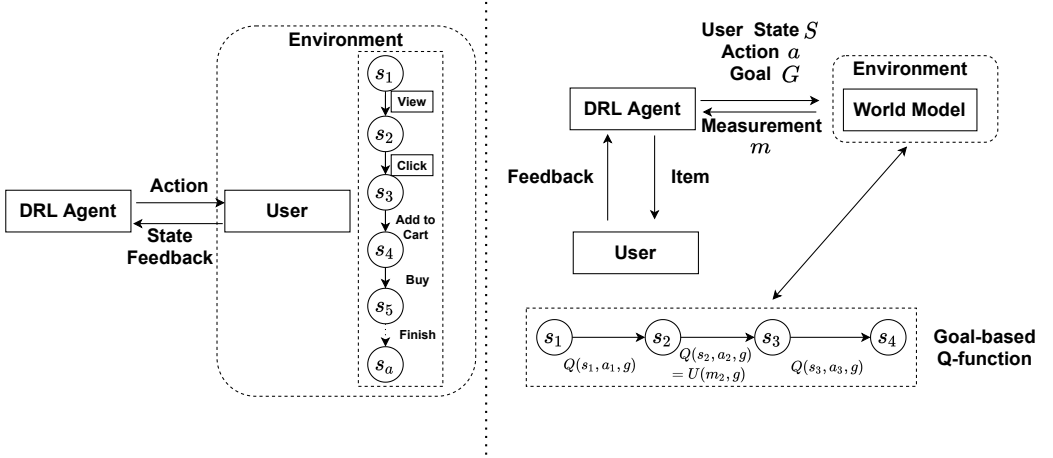


Fig. 4. Left is the general structure of model-free methods. Right is the structure for GoalRec which is a model-based method. A sample trajectory is used to demonstrate the difference between them [96].

GoalRec [96] designs a disentangled universal value function to be integrated with the world model to help the agent deal with different recommendation tasks. The universal value function is defined as

$$V_{\pi}(s) = \mathbb{E} \left[ \sum_{t=0}^{\infty} r(s_t, a_t) \prod_{k=0}^t \gamma s_k | s_0 = s \right]. \quad (17)$$

Moreover, GoalRec introduces a new variable goal  $g \in G$  used to represent users' future trajectory and measurement  $m \in M$ .  $m$  is an indicator that reflects users' response to the given future trajectory based on historical behaviors. Based on that, the optimal action will be selected based on

$$a^* = \max_a U(M(s, a), g) \quad (18)$$

with a customized liner function  $U(\cdot)$ .

*Hybrid methods.* Hybrid method can be recognized as a midpoint between value-based and policy gradient-based methods. DeepChain [125] uses the multi-agent setting to relieve the sub-optimality problem. The sub-optimality problem is caused by the *one for all* setting that optimizes one policy for all users. Hence, DeepChain designs a multi-agent setting that adopts several agents to learn consecutive scenarios and jointly optimizes multiple recommendation policies. The main training algorithm used is DDPG. To this end, users' actions can be formulated in a model-based form as follows:

$$\sum_{m,d} [p_m^s(s_t, a_t) \gamma Q_{\theta}(s'_t, \pi_m(s'_t)) + p_m^c(s_t, a_t)(r_t + \gamma Q_{\theta}(s'_t, \pi_d(s'_t))) + p_m^l(s_t, a_t) r_t] 1_m \quad (19)$$

where  $m$  represents the number of actor networks,  $c, l, s$  represent the three different scenarios,  $1_m$  is used to control the activation of two actors and  $(m, d) \in \{(1, 2), (2, 1)\}$ .

*Discussion.* Model-based methods aim to learn a model or representation to represent the whole environment so that the agent can plan ahead and receive better sample efficiency. The drawback of such a method is that the ground-truth representation of the environment is unavailable in recommendation scenarios as it dynamically changes, leading to a biased representation. Moreover, model-based methods use the transition probability function  $\mathcal{P}$  to estimate the optimal policy. As

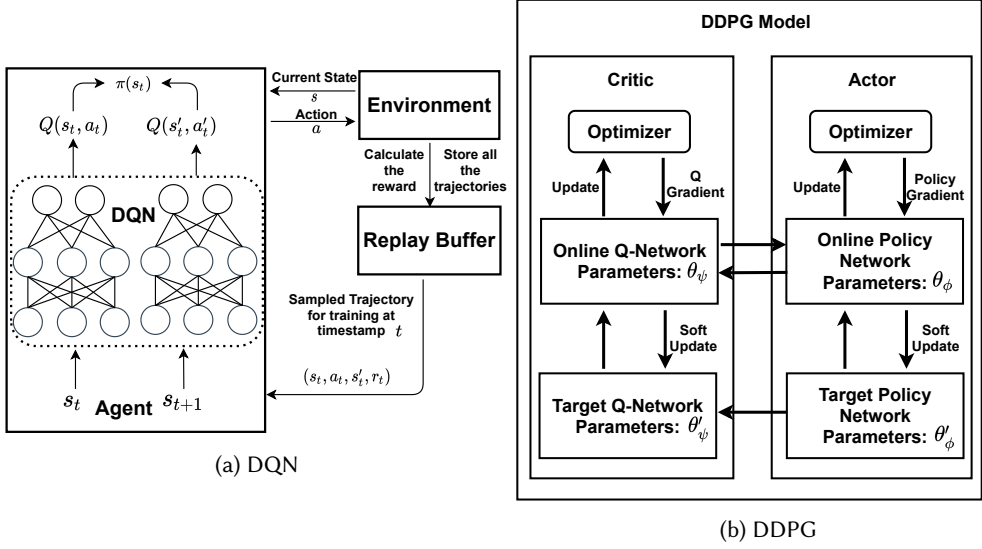


Fig. 5. The typical structure of DQN and DDPG

mentioned, the transition probability function is normally equivalent to users' behavior probability which is hard to determine in a recommender system. Hence, existing works [4, 18, 96, 119, 125, 135] approximate  $\mathcal{P}$  using a neural network or embedding it into the world model. Zhao et al. [125] design a probability network to estimate  $\mathcal{P}$  while [4, 18] uses a GAN to generate user behavior where  $\mathcal{P}$  is embedded in the latent space. Different from them, [96, 135] relies on the world model to predict users' next behavior and feed it into the policy learning process.

The challenges of model-based DRL are not widely used in RS and can be summarized into the following facets:

- $\mathcal{P}$  is hard to determine in real-world recommender systems.
- If approximation is used to estimate  $\mathcal{P}$ , the overall model complexity will substantially increase as it requires approximating two different functions  $\mathcal{P}$  and the recommendation policy  $\pi$  by using a large amount of user behavior data.
- World model-based methods require periodic re-training to ensure the model can reflect user interests in time which increases the computation cost.

### 3.2 Model-free deep reinforcement learning based methods

Compared with model-based methods, model-free methods are relatively well-studied. Different from model-based methods,  $\mathcal{P}$  is unknown and not required in model-free methods. Model-based methods enable the agent to learn from previous experiences. In this subsection, we categorize model-free based DRL in RS into three parts: value-based, policy-based and hybrid methods.

*Value based methods.* As mentioned, Deep Q-learning and its variants are typical value-based DRL methods widely used in DRL-based RS. DRN [130] is the first work utilizing Deep Q-Networks (DQN) in RS. It adopts Double DQN (DDQN) [91] to build a user profile and designs an activeness score to reflect how frequently a user returns after one recommendation plus users' action (click or not) as the reward. DRN provides a new approach to integrating DRL into RS when dealing with a

Table 3. List of reviewed publications in Model-free DRL-based RS

Tasks	Note	Work
Value-based	Vanilla DQN and its extensions	[54, 124, 130]
	DQN with state/action space optimization	[42, 52, 104, 134]
	DQN with graph/image input	[28, 33, 53, 71, 88, 127, 131]
	DQN for joint learning	[73, 123, 129]
Policy-based	Vanilla REINFORCE	[13, 14, 45, 64, 68, 72, 99, 107, 110]
	REINFORCE uses graph structure/input	[11, 98, 100, 103]
	Non-REINFORCE based	[39, 114]
Hybrid	Vanilla DDPG	[9, 59, 97, 124, 128]
	with Knowledge Graph	[17, 24, 35, 36, 36, 105, 117, 120, 121]

dynamic environment. The key objective function can be found as follows,

$$\mathbb{E}[r_{t+1} + \gamma Q_{\theta'_t}(s_{t+1}, \arg \max_{a'} Q_{\theta_t}(s_t, a'))] \quad (20)$$

where  $a'$  is the action that gives the maximum future reward according to  $\theta_t$ ,  $\theta_t$  and  $\theta'_t$  are different parameters for two different DQNs. Zhao et al. [124] points out that negative feedback will also affect recommendation performance which DRN does not consider. Moreover, positive feedback is sparse due to the large number of candidate items in RS. Only using positive feedback would lead to convergence problems. Hence, DEERS is proposed to consider both positive and negative feedback simultaneously by using DQN. Gated Recurrent Units (GRU) are employed to capture users' preferences for both a positive state  $s^+$  and negative state  $s^-$  and the final objective function can be computed as:

$$\mathbb{E}[r_{t+1} + \gamma \max_{a_{t+1}} Q_{\theta_q}(s_{t+1}^+, s_{t+1}^-, a_{t+1}) | s_t^+, s_t^-, a_t]. \quad (21)$$

Lei et al. [54] introduces attention mechanisms into the DQN to leverage social influence among users. To be specific, a social impact representation  $U_v$  is introduced into the state representation. Matrix factorization is adopted to determine similarity among users and hence present the social influence. Social attention is introduced to distill the final state representation. In addition, a few studies focus on user profiling to improve recommendation performance [52, 104, 134]. Lei and Li [52] claims that user feedback contains useful information in the previous feedback even when the user does not like the recommended items. Some existing studies focus on final feedback which ignore the importance from earlier steps to later ones. Hence, user-specific DQN (UQDN) is proposed to consider multi-step feedback from users. It employs Matrix Factorization to generate user-specific latent state spaces. The newly defined objective function with the user-specific latent state space can be represented as

$$\mathbb{E}[r_{t+1} + \gamma \max_{a_{t+1}} \bar{Q}_{\theta_q}(s_{t+1}, a_{t+1}) + \bar{\mathbf{b}}_u - Q_{\theta_q}(s_{t+1}, a_{t+1})] \quad (22)$$

where  $\bar{\mathbf{b}}_u$  is the learned user latent representation. Zou et al. [134] also points out that most studies do not consider users' long-term engagement in the state representation as they focus on the immediate reward. FeedRec is proposed that combines both instant feedback and delayed feedback into the model to represent the long-term reward and optimize the long-term engagement by using DQN. To be specific, time-LSTM is employed to track users' hierarchical behavior over time to represent the delayed feedback which contains three different operations:  $h_{skip}$ ,  $h_{choose}$ ,  $h_{order}$ . The state space is the concatenation of those operations and users' latent representation. Differently, Xiao et al. [104] focuses on the user privacy issue in recommender systems. Deep user profile

perturbation (DUPP) is proposed to add perturbation into the user profile by using DQN during the recommendation process. Specifically, DUPP adds a perturbation vector into users' clicked items as well as the state space, which contains users' previous behavior.

Distinct from previous studies which focus on optimizing user profiles or state spaces, some studies aim to optimize the action space formed by interactions with items. In the situation of basket recommendation, the user is suggested multiple items as a bundle, which is called a recommendation slate. It leads to combinatorially large action spaces making it intractable for DQN based recommendation models.

SlateQ [42] is proposed to decompose slate Q-value to estimate a long-term value for individual items, and it is represented as,

$$Q_{\theta_q}(s_t, a_t) = \sum_{i \in a_t} p(i|s_t, a_t) \bar{Q}_{\theta_q}(s_t, i) \quad (23)$$

where  $\bar{Q}_{\theta}(s, i)$  is the decomposed Q-value for item  $i$ . The decomposed Q-value will be updated by the following rule which is similar to traditional DQN,

$$\bar{Q}_{\theta_q}(s_t, i) \leftarrow \alpha \left( r_t + \gamma \sum_{j \in a_{t+1}} p(j|s_{t+1}, a_{t+1}) \bar{Q}_{\theta_q}(s_{t+1}, j) \right) + (1 - \alpha) \bar{Q}_{\theta_q}(s_t, i). \quad (24)$$

Different from other mode-free methods, Slate-Q assumes that the transition probability  $p(i|s_t, a_t)$  is known.

Vanilla DQN methods may not have sufficient knowledge to handle complex data such as images and graphs. Tang and Wang [89] firstly models users' click behavior as an embedding matrix in the latent space to include the skip behaviors of sequence patterns for sequential recommendation. Based on that, Gao et al. [28] propose DRCGR, which adopts CNN and GAN into DQN to help the agent to better understand high-dimensional data, e.g., a matrix. Two different convolution kernels are used to capture users' positive feedback. In the meantime, DRCGR uses GANs to learn a negative feedback representation to improve robustness. Another typical data format is the graph, which is widely used in RS, including knowledge graphs. Lei et al. [53] propose GCQN which adopts Graph Convolutional Networks (GCN) [48] into the DQN which constructs the state and action space as a graph-aware representation. Differently, GCQN introduces the attention aggregator:  $\sum_{w \in \mathcal{N}(i)} \alpha_{iu} e_u$  which demonstrates better performance than the mean-aggregator and pooling-aggregator. For item  $i$ , the graph-aware representation can be represented as,

$$\sigma \left( W_{fc} [e_i \oplus \sum_{w \in \mathcal{N}(i)} \alpha_{iu} e_u + b_{fc}] \right) \quad (25)$$

where  $W_{fc}, b_{fc}$  are the parameters for the fully-connected layer,  $e_u$  is the embedding for user  $u$  and  $\mathcal{N}(i)$  is the set of one-hot neighbours of item  $i$  in graph  $G(i)$ . Zhou et al. [131] propose KGQR uses a similar strategy to transform the information into a knowledge graph which is fed into the GCN to generate the state representation. Notably, KGQR presents a different state representation generation method. For given node  $i$ , the neighbourhood representation with a  $k$ -hop neighborhood aggregator can be represented as,

$$e_i^k = \sigma \left( W_k \frac{1}{|\mathcal{N}(i)|} \sum_{t \in \mathcal{N}(i)} e_t^{k-1} + B_k e_i^{k-1} \right) \quad (26)$$

where  $\mathcal{N}(i)$  is the set of neighboring nodes,  $W_k, B_k$  are the parameter of the aggregator. Those neighbourhood representations will be fed into a GRU and the state representation will be generated. Another application domain for using graph data is job recommendation which requires considering multiple factors jointly such as salary, job description, job location etc. SRDQN [87] constructs a

probability graph to represent a candidate's skill set and employs a multiple-task DQN structure to process these different factors concurrently.

There are some studies targeting recommendation and advertising simultaneously in e-commerce environments [73, 123, 129]. Pei et al. [73] mentions when deploying RS into real-world platforms such as e-commerce scenarios, the expectation is to improve the profit of the system. A new metric, Gross Merchandise Volume (GMV), is proposed to measure the profitability of the RS to provide a new view about evaluating RS in advertising. Different from GMV, Zhao et al. [129] separates recommendation and advertising as two different tasks and proposes the Rec/Ads Mixed display (RAM) framework. RAM designs two agents: a recommendation agent and an advertising agent, where each agent employs a CDQN to conduct the corresponding task. Zhao et al. [123] find that advertising and recommendation may harm each other and formulate a rec/ads trade-off. Their proposed solution, DEARS, contains two RNNs. Two RNNs are employed to capture user preferences toward recommendations and ads separately. Based on that, DQN is employed to take those two outputs as the input to construct the state and output the advertising.

*Policy-based methods.* Policy-based DRL can be divided into two parts which are Constrained Policy Optimization (CPO) [1] and policy gradient. Zhang et al. [114] uses CPO to identify the contradiction between text feedback and historical preferences. It provides a solution for using DRL in the situation where users' feedback is entirely different from previous feedback in RS. Policy gradient-based methods are the other stream in policy-based DRL methods for RS. These methods aims to optimize the policy  $\pi$  directly instead of estimating the Q-value like DQN. A well-known and widely used policy gradient method in RS is REINFORCE which uses the following rule for policy  $\pi_{\theta_\pi}$ ,

$$\theta \leftarrow \theta + \alpha \mathbb{E}_{\tau \sim d_{\pi_{\theta_\pi}}} \left[ \sum_{t=1}^T r(s_t^i, a_t^i) \sum_{t=1}^T \nabla_{\theta_\pi} \log \pi_{\theta_\pi}(s_t^i, a_t^i) \right] \quad (27)$$

where  $i$  is sampled trajectories from  $\pi_{\theta}(a_t|s_t)$ . Pan et al. [72] propose Policy Gradient for Contextual Recommendation (PGCR), which adopts REINFORCE and considers contextual information. PGCR assumes that the policy follows the multinoulli distribution, in which case the transition probability can be estimated easily through sampling from previously seen context.

Wang et al. [99] incorporate CNNs and attention mechanisms in REINFORCE for explainable recommendation. Specifically, this work designs a coupled agent structure where one agent generates the explanation and the other makes recommendations based on the generated explanation.

Chen et al. [13] increases the scalability of REINFORCE to ensure it can deal with the extremely large action space under recommendation scenarios. To be specific, it introduces a policy correction gradient estimator into REINFORCE to reduce the variance of each gradient by doing importance sampling. The new update rule becomes

$$\theta_\pi \leftarrow \theta_\pi + \alpha \sum_{\tau \sim \beta} \left[ \sum_{t=1}^T \frac{\pi_{\theta_\pi}(s_t, a_t)}{\pi_\beta(s_t, a_t)} r(s_t^i, a_t^i) \sum_{t=1}^T \nabla_{\theta_\pi} \log \pi_{\theta_\pi}(s_t^i, a_t^i) \right] \quad (28)$$

where  $\pi_\beta$  is the behavior policy trained by state-action pairs without the long-term reward and  $\pi_\theta$  is trained based on the long-term reward only. It is worth mentioning that the vanilla REINFORCE algorithm is on-policy, and importance sampling will make REINFORCE behave like an off-policy method with the following gradient format,

$$\mathbb{E}_{\tau \sim d_{\pi_\theta}} \left[ \prod_{t'=1}^t \frac{\pi_\theta(s_t, a_t)}{\pi_{\theta_s}(s_{t'}, a_{t'})} \sum_{t'=t}^T r(s_{t'}, a_{t'}) \sum_{t=1}^T \nabla_{\theta} \log \pi_\theta(s_t, a_t) \right] \quad (29)$$

where  $\pi_{\theta_s}$  is the sample policy parameter. Xu et al. [107] also finds that the REINFORCE method suffers from a high variance gradient problem and Pairwise Policy Gradient (PPG) is proposed. Different from policy correction, PPG uses Monte Carlo sampling to sample two different actions  $a, b$  and compare the gradient to update  $\theta$ ,

$$\mathbb{E}_{\tau \sim d_{\pi_{\theta_\pi}}} \left( \sum_a \sum_b (r(s, a) - r(s, b)) \sum_{t=1}^T (\nabla_{\theta_\pi} \log \pi_{\theta_\pi}(s_t, a_t) - \nabla_{\theta_\pi} \log \pi_{\theta_\pi}(s_t, b_t)) \right). \quad (30)$$

Ma et al. [64] extends the policy correction gradient estimator into a two-stage setting which are  $p(s_t, a^p)$  and  $q(s_t, a|a^p)$  and the policy can be written as

$$\sum_{a^p} p(s_t, a^p) q(s_t, a|a^p). \quad (31)$$

In addition, weight capping and self-normalized importance sampling are used to further reduce the variance. Moreover, a large state space and action space will cause sample inefficiency problems as REINFORCE relies on the current sampled trajectories  $\tau$ . Chen et al. [14] finds that the auxiliary loss can help improve the sample efficiency [44, 81]. Specifically, a linear projection is applied to the state  $s_t$ , the output is combined with action  $a_t$  to calculate the auxiliary loss and appended into the final overall objective function for optimization.

Another prototype of vanilla policy gradient in DRL-based RS is the policy network. Montazer-alghaem et al. [68] designs a policy network to extract features and represent the relevant feedback that can help the agent make a decision. Similar to DQN, this work uses a neural network to approximate the Q-value and the policy directly without theoretical analysis. Ji et al. [45] extend the policy network by introducing spatio-temporal feature fusion to help the agent understand complex features. Specifically, it considers both the current number and the future number of vacant taxis on the route to recommend routes for taxis. Yu et al. [110] introduces multi-modal data as new features to conduct vision-language recommendation by using historical data to train REINFORCE. ResNet and attention are used to encode vision and text information, respectively. Moreover, two rewards are introduced with a customized ratio  $\lambda$  to balance vision and text information.

Knowledge Graphs (KG) are widely used in RS to enrich side information, provide explainability and improve recommendation performance. Similar to DQN, vanilla REINFORCE cannot properly handle graph-like data. Wang et al. [98] propose a method named Knowledge-guided Reinforcement Learning (KERL), which integrates knowledge graphs into the REINFORCE algorithm. To be specific, KERL adopts TransE [6] to transfer the knowledge graph into a graph embedding and utilizes a multilayer perceptron (MLP) to predict future knowledge of user preferences. The state representation can be written as

$$h_t \oplus \text{TransE}(\mathcal{G}) \oplus \text{MLP}(\text{TransE}(\mathcal{G})) \quad (32)$$

where  $h_t$  is the hidden representation from the GRU for sequential behavior and  $\mathcal{G}$  is the knowledge graph.

Different from KERL, Xian et al. [103] propose Policy-Guided Path Reasoning (PGPR), which formulates the whole environment as a knowledge graph. The agent is trained to find the policy to find good items conditioned on the starting user in the KG by using REINFORCE. PGPR uses the tuple  $(u, e_t, h_t)$  to represent the state instead of the graph embedding where  $e_t$  is the entity the agent has reached at  $t$  for user  $u$  and  $h_t$  is the previous action before  $t$ . The action in PGPR is defined as the prediction of all outgoing edges for  $e_t$  based on  $h_t$ .

Wang et al. [100] propose a knowledge graph policy network (KGPoly) which puts the KG into the policy network and adopts REINFORCE to optimize it. In addition, KGPoly uses negative sampling instead of stochastic sampling to overcome the false negative issue—sampled items behave

differently during training and inference. Similar to GCQN, attention is also employed to establish the representation for its neighbors.

Due to the on-policy nature of REINFORCE, it is difficult to apply it to large-scale RS as the convergence speed will be a key issue. To relieve this, Chen et al. [11] propose TPGR, which designs a tree-structured policy gradient method to handle the large discrete action space hierarchically. TPGR uses balanced hierarchical clustering to construct a clustering tree. Specifically, it splits a large-scale data into several levels and maintains multiple policy networks for each level to conduct the recommendation. The results are integrated at the final stage.

As mentioned, policy gradient can be further extended to deterministic policy gradient (DPG) [86]. Hu et al. [39] propose Deterministic Policy Gradient with Full Backup Estimation (DPG-FBE) to complete a sub-task of recommendation. DPG-FBE considers a search session MDP (SSMDP) that contains a limited number of samples, where the stochastic policy gradient method like REINFORCE cannot work well.

*Hybrid methods.* The most common model-free hybrid method used would be the actor-critic algorithm where the critic network uses the DQN and the actor uses the policy gradient. The common algorithm used to train actor-critic is DDPG with the following objective function,

$$\mathbb{E}[r_t + \gamma Q_{\theta'_q}(s_{t+1}, \mu_{\theta'_\pi}(s_{t+1})) - Q_{\theta_q}(s_t, a_t)] \quad (33)$$

where  $\theta_q, \theta'_q$  is the parameter for Q-learning at time  $t, t+1$  while  $\theta'_\pi$  is the parameter for deterministic policy gradient at time  $t+1$ . Zhao et al. [128] propose LIRD, which uses the vanilla actor-critic framework to conduct list-wise recommendations. In order to demonstrate the effectiveness of LIRD, a pre-trained user simulator is used to evaluate the effectiveness of LIRD where the transition probability is approximated using the cosine similarity for a given state-action pair  $s_t, a_t$ . Zhao et al. [124] further extend LIRD into page-wise recommendation and proposed DeepPage. Similar to other previous work, GRU is employed to process the sequential pattern. Moreover, similar to DRCGR, DeepPage formulates the state as a page, then CNNs are employed to capture features and fed to the critic network. The final state representation is the concatenation of the sequential pattern and the page features. Additionally, there are a few studies focusing on different scenarios such as top-aware recommendation [59], treatment recommendation [97], allocating impressions [9] etc. Liu et al. [59] introduces a supervised learning module (SLC) as the indicator to identify the difference between the current policy and historical preferences. SLC will conduct the ranking process to ensure the recommendation policy will not be affected by the positional bias – the item appearing on top receives more clicks. Similarly, Wang et al. [97] also integrates the supervised learning paradigm into DRL but in a different way. An expert action  $\hat{a}_t$  is provided when the critic evaluates the policy and the update rule is slightly different than normal DQN,

$$\theta_q \leftarrow \theta_q + \alpha \sum_t [Q_{\theta_q}(s_t, \hat{a}_t) - r_t - \gamma Q_{\theta'_q}(s_t, \mu_{\theta'_\pi}(s_t))] \nabla_{\theta_q} Q_{\theta_q}(s_t, a_t). \quad (34)$$

However, such a method is not universal as the acquisition of expert action is difficult and depends on the application domain.

Similar to policy gradient and DQN, Knowledge Graphs (KG) are also used in actor-critic-based methods. Chen et al. [17] propose KGRL to incorporate the substantial information of knowledge graphs to help the critic to better evaluate the generated policy. A knowledge graph is embedded into the critic network. Different from previous studies which use the KG as the environment or state representation, KGRL uses KG as a component in the critic, which can guide the actor to find a better recommendation policy by measuring the proximity from the optimal path. Specifically, a graph convolutional network is used to weight the graph and Dijkstra's algorithm is employed to find the optimal path for finally identifying the corresponding Q-value. Zhao et al. [121] claim that

human's demonstration could improve path searching and propose ADAC. ADAC also searches for the optimal path in the KG but further adopts adversarial imitation learning and uses expert paths to facilitate the search process. Feng et al. [24] propose MA-RDPG, which extends the standard actor-critic algorithm to deal with multiple scenarios by utilizing a multi-actor reinforcement learning setting. Specifically, two different actor-networks are initialized while only one critic network will make the final decision. Those two actor networks can communicate with each other to share information and approximate the global state. Zhang et al. [117] find that there are multiple factors can affect the selection of electric charging station. Hence, it uses a similar idea to recommend the electric vehicle charging station by considering current supply, future supply, and future demand. He et al. [36] figure out that the communication mechanism in MA-RDPG will harm actors as they are dealing with independent modules, and there is no intersection. Hence, He et al. [36] extend MA-RDPG into multi-agent settings which contain multiple pairs of actors and critics and remove the communication mechanism to ensure independence.

Different from [24], He et al. [36] use 'soft' actor-critic (SAC) [35], which introduces a maximum entropy term  $\mathcal{H}(\pi(s_t, \phi_t))$  to actor-critic to improve exploration and stability with the stochastic policy  $\pi(s_t, \phi_t)$ . Similar to the multi-agent idea, Zhao et al. [120] use a hierarchical setting to help the agent learn multiple goals by setting multiple actors and critics. In comparison, hierarchical RL uses multiple actor-critic networks for the same task. It splits a recommendation task into two sub-tasks: discovering long-term behavior and capturing short-term behavior. The final recommendation policy is the combination of the optimal policies for the two sub-tasks. Similarly, Xie et al. [105] use the hierarchical setting for integrated recommendation by using different sourced data. The objective is to work out the sub-policies for each source hierarchically and form the final recommendation policy afterward.

*Discussion.* In RS, model-free methods are generally more flexible than model-based methods as they do not require knowing the transition probability. We summarize the advantages and disadvantages of the three kinds of methods described under the model-free category. DQN is the first DRL method used in RS, which is suitable for small discrete action spaces. The problems with DQN in RS are:

- RS normally contains large and high-dimensional action spaces.
- The reward function is hard to determine which will lead to inaccurate value function approximation.

Specifically, the high dimensional action space in context of recommender systems is recognized as a major drawback of DQN [67, 90]. The reason lies in the large number of the candidate items. Hence, DQN, as one of the most popular schemes, is not the best choice for RS in many situations. Moreover, some unique factors need to be considered when designing the reward function for RS such as social inference. It introduces extra parameters to the Q-network and hinders the convergence.

Policy gradient does not require the reward function to estimate the value function. Instead, it estimates the policy directly. However, policy gradient is designed for continuous action spaces. More importantly, it will introduce high variance in the gradient. Actor-critic algorithms combine the advantages of DQN and policy gradient. Nonetheless, actor-critic will map the large discrete action space into a small continuous action space to ensure it is differentiable, which may cause potential information loss. Actor-critic uses DDPG and thus inherits disadvantages from DQN and DPG, including difficulty in determining the reward function and poor exploration ability.



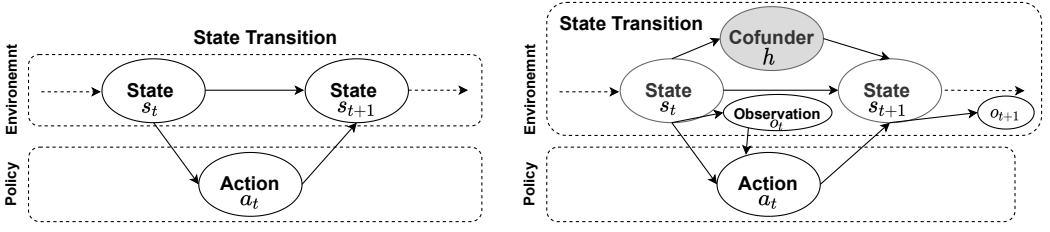


Fig. 6. Left is the traditional MDP transition; Right is the POMDP which considers the environmental confounders such as social influence [82].

### 3.3 Component Optimization in Deep Reinforcement Learning based RS

There are a few studies that use DRL in RS for goals other than improving recommendation performance or proposing new application domains. We split the literature based on the following components: environment, state representation, and reward function. Existing studies usually focus on optimizing one single component in the DRL setting (as illustrated in Figure 1).

Table 4. List of publications reviewed in this section

Component	Work
Environment	[40, 41, 75, 76, 82–84, 126]
State	[58, 60, 61]
Reward	[15]

#### 3.3.1 Environment Simulation and Reconstruction.

Many environments are available for evaluating deep reinforcement learning. Two popular ones are OpenAI gym-based environment [8] and MuJoCo<sup>1</sup>. Unfortunately, there is no standardized simulation platform or benchmark specific to reinforcement learning based recommender systems. Existing work on DRL in RS is usually evaluated through offline datasets or via deployment in real applications. The drawback for evaluating offline datasets include:

- Different studies use different environment construction methods which leads to unfair comparison. For instance, some studies use the KG as the environment while some studies assume the environment is gym-like or design a simulator for specific tasks.
- With offline datasets, users' dynamic interests, and environment dynamics are hard to maintain. Deploying the method into a real application is difficult for academic research as it takes time and costs money. Hence, a standardized simulation environment is a desirable solution.

There are several different studies that provide standardized gym<sup>2</sup>-based simulation platforms for DRL-based RS research in different scenarios. RecSim [41] is a configurable platform that supports sequential interaction between the system and users. RecSim contains three different tasks: interest evolution, interest exploration and long-term satisfaction. RecoGym [75] provides an environment for recommendation and advertising. In addition, RecoGym also provides simulation of online experiments such as A/B-tests. However, RecSim and RecoGym are designed for bandit behavior

<sup>1</sup><http://mujoco.org/>

<sup>2</sup><https://gym.openai.com/>

which means users' interests will not change over time. VirtualTB [84] is proposed to relieve such problems. VirtualTB employs imitation learning to learn a user model to interact with the agent. GANs are employed to generate users' interests. Similar to VirtualTB, Recsimu [126] uses a GAN to tackle the complex item distribution. In addition, PyRecGym [83] accommodates standard benchmark datasets into a gym-based environment. MARS-Gym [76] provides a benchmark framework for marketplace recommendation. [40] suggests that existing simulation environments are biased because of biased logged data. Two common biases are discussed: popularity bias and positivity bias. To reduce the effect from those biases, SOFA introduces an Intermediate Bias Mitigation Step for debiasing purposes.

One work discusses environment reconstruction by considering confounders. [82] claims that users' interests may be affected by social networks which may introduce extra bias to the state and affect the decision-making process. A multi-agent setting is introduced to treat the environment as an agent which can partially relieve the hidden confounder effect. Specifically, a deconfounded environment reconstruction method DEMER is proposed. Different from previously mentioned methods, DEMER assumes the environment is partially observed and models the whole recommendation task as a Partially Observed MDP (POMDP). Different from an MDP, a POMDP contains one more component observation  $o \in \mathcal{O}$  and the action  $a_t$  is derived based on the observation  $o_t$  instead of the state  $s_t$  by  $a_t = \pi_a(o_t)$ . DEMER assumes there is a confounder policy  $\pi_h$  for observation  $o_h$  which is composed by  $a_t$  and  $o_t$ :  $a_h = \pi_h(a_t, o_t)$ . Moreover, another observation  $o_b$  is introduced to observe the transition as well.  $\pi_b$  is the corresponding policy and  $a_b = \pi_b(o_b) = \pi_b(o_t, a_t, a_h)$ . DEMER uses generative adversarial imitation learning (GAIL) to imitate the policy  $A, B$ . Given trajectory  $\{o_t, o_h, o_b\}$  for different policies  $A$  and  $B$ , the objective function is defined as

$$\begin{aligned}
 (\pi_a, \pi_b, \pi_h) &= \arg \min_{(\pi_a, \pi_b, \pi_h)} \mathbb{E}_{s \sim \tau} (L(s, a_t, a_b)) \\
 \text{where } L(s, a_t, a_b) &= \mathbb{E}_{(\pi_a, \pi_b, \pi_h)} [\log D(s, a_t, a_b)] - \lambda \sum_{\pi \in \{\pi_a, \pi_b, \pi_h\}} H(\pi)
 \end{aligned} \tag{35}$$

where  $L(\cdot)$  is the loss function,  $D(\cdot)$  is a discriminator and  $H(\pi)$  is introduced in GAIL.

### 3.3.2 State Representation.

State representation is another component in DRL-based RS which exists in both model-based and model-free methods. Liu et al. [60] find that the state representation in model-free methods would affect recommendation performance. Existing studies usually directly use the embedding as the state representation. Liu et al. [58, 61] propose a supervised learning method to generate a better state representation by utilizing an attention mechanism and a pooling operation as shown in Figure 7. Such a representation method requires training a representation network when training the main policy network, which increases the model complexity.

### 3.3.3 Robustness of Reward Functions.

The reward function is crucial for methods involving DQN. A robust reward function can significantly improve training efficiency and performance. Kostrikov et al. [50] find that the DQN may not receive the correct reward value when entering the absorbing state. That is, when the absorbing state is reached, the agent will receive zero reward and harm policy learning. The reason behind this is that when designing the environment zero reward is implicitly assigned to the absorbing state as it is hard to determine the reward value in such a state. Chen et al. [15] propose a robust DQN method, which can stabilize the reward value when facing the absorbing state. The new

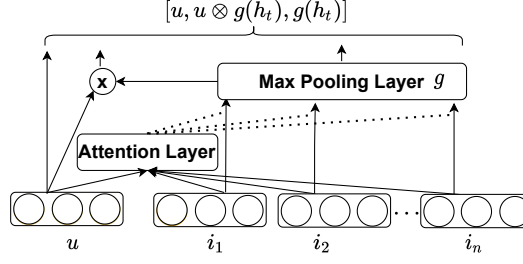


Fig. 7. State representation used in works [58, 61].  $h_t$  is the output of an attention layer that takes the representation of users' history at time  $t$  as input and  $g(\cdot)$  is the pooling operation.

reward formula can improve the robustness, which is defined as follows:

$$r = \begin{cases} r_t & \text{if } s_{t+1} \text{ is an absorbing state} \\ r_t + \gamma Q_{\theta'}(s_{t+1}, a_{t+1}) & \text{otherwise.} \end{cases} \quad (36)$$

The major difference is that  $r_t$  is assigned to the absorbing state to ensure the agent can continue learning. One remaining problem in current DRL-based RS is the reward sparsity, i.e., the large state and action spaces make the reward sparsity problem more serious. One of the possible solution would be a better designed reward by using the reward shaping [69].

## 4 EMERGING TOPICS

While existing studies have established a solid foundation for DRL-based RS research, this section outlines several promising emerging research directions.

### 4.1 Multi-Agent and Hierarchical Deep Reinforcement Learning-based RS

Recommender systems are monolithic systems containing tasks such as searching, ranking, recommendation, advertising, personalization, and diverse stakeholders such as users and items. Most existing methods are based on single agent.

Multi-Agent Reinforcement Learning (MARL) is a subfield of reinforcement learning and it is capable of learning multiple policies and strategies.

While a single-agent reinforcement learning framework can only handle a single task, many studies consider the multi-task situation in RS and employ multi-agent DRL (MADRL) or hierarchical DRL (HDRL). HDRL [51] is proposed to handle complex tasks by splitting such tasks into several small components and asks the agent to determine sub-policies. HDRL belongs to a single-agent reinforcement learning framework such that the agent contains a meta-controller and several controllers. The meta-controller splits the task, and the controllers learn the value and reward functions for designated tasks to get a series of sub-policies. There are a few studies already utilizing HDRL in RS. Xie et al. [105] target integrated recommendation to capture user preferences on both heterogeneous items and recommendation channels. Specifically, the meta-controller is used for item recommendation, and controllers aim to find the personalized channel according to user channel-level preferences. Zhang et al. [113] uses HDRL for course recommendation in MOOCs, which contains two different tasks: profile reviser and recommendation. The meta-controller aims to make course recommendations by using the revised profile pruned by the controllers.

Different from HDRL, MADRL [23] introduces multiple agents to handle the sub-tasks. Gui et al. [33] uses the MADRL for twitter mention recommendation where three agents are initialized. The three agents need to generate different representations for the following tasks: query text, historical

text from authors and historical text from candidate users. Once the representations are finalized, the model will conduct the recommendation based on the concatenation of representations. Feng et al. [24] and He et al. [36] provide two different views of the communication mechanism in MADRL and demonstrate that agents could work collaboratively or individually. Zhao et al. [129] designs a MADRL framework for two tasks where two agents are designed to conduct advertising and recommendation respectively. Zhang et al. [119] uses MADRL for collaborative recommendation where each agent is responsible for a single user. MADRL is adopted to help the recommender consider both collaboration and potential competition between users. Zhang et al. [117] designs a charging recommender system for intelligent electric vehicles by using decentralized agents to handle sub-tasks and a centralized critic to make the final decision.

Hierarchical multi-agent RL (HMARL) [66] proves that MARL and HRL can be combined. Recently, Yang et al. [109] introduces HMDRL into the continuous action space, which provides a direction for RS. Zhao et al. [120] uses HMARL for multi-goal recommendation where the meta-controller considers users' long-term preferences and controllers focus on short-term click behavior. While the meta-controller and controllers in HDRL deal with sub-tasks that belong to a single task, HMARL focuses on multi-task or multi-goal learning where the meta-controller and controllers belong to different agents and deal with different tasks or goals.

HMDRL would be a suitable solution for future research work in DRL-based RS where HDRL can be used to split a complex task into several sub-tasks such as users' long-term interests and short-term click behavior, and MADRL can jointly consider multiple factors such as advertising Zhao et al. [120].

## 4.2 Inverse Deep Reinforcement Learning for RS

As mentioned, the reward function plays a critical role in DRL-based recommender systems. In many existing works, reward functions are manually designed. The common method uses users' click behavior to represent the reward and to reflect users' interests. However, such a setting can not represent users' long-term goals [134] as clicking or not only depicts part of the feedback information from users. It requires significant effort to design a reward function, due to the large number of factors that can affect users' decision, such as social engagement or bad product reviews, which may adversely affect recommendation performance. It is difficult to include all potential factors into the reward function because not every factor can be represented properly. A few works [19, 31] show that manually designed reward functions can be omitted by employing inverse reinforcement learning (IRL) [70] or generative adversarial imitation learning (GAIL) [37]. Such inverse DRL-based methods require using expert demonstration as the ground truth. However, expert demonstration is often hard to obtain for recommendation scenarios. Those two studies conduct experiments in an offline dataset-based simulation environment that can access expert demonstration. In contrast, Chen et al. [19] use IRL as the main algorithm to train the agent while Gong et al. [31] use both demonstration and reward to train the agent. Zhao et al. [121] also employ GAIL to improve recommendation performance. In this work, GAIL is used to learn the reasoning path inside the KG to provide side information to help the agent learn the policy. Although IRL achieves some progress in RS, the lack of demonstration is a key shortcoming that impedes adoption in RS. One possibility is to use the IRL method in casual reasoning to help improve interpretability [5] thus boosting recommendation performance. Alternately, IRL may be suitable for learning users' long-term and static behavior to support the reward function.

## 4.3 Graph Neural Networks for Boosting DRL-based RS

Graph data and KG are widely used in RS. Graph modeling enables an RS to leverage interactions between users and the recommender for reasoning or improving interpretability. According to

existing studies about deep learning-based RS [115], embedding is a technique used to get the representation for the input data. Graph embedding is a common solution to handle graph-like data. GCN is a type of graph embedding method which are broadly used in RS to process graph data. Wang et al. [94] propose a variant of GCN to learn the embedding for KG. Specifically, they propose knowledge graph convolutional networks (KGCN) to capture the high-order structural proximity among entities in a knowledge graph.

In DRL-based RS, graph data are handled similarly—s underthe transformed into an embedding and fed to the agent. Wang et al. [98] uses a traditional graph embedding method TransE [6] to generate the state representation for DRL-based RS. There are several studies that use GCN in DRL for recommendations under different settings. Jiang et al. [46] propose a graph convolutional RL (DGN) method which integrates the GCN into the Q-learning framework for general RL problems by replacing the state encoding layer with the GCN layer. Lei et al. [53] extend this method into the deep learning field and apply it to recommender systems. To be specific, multiple GCN layers are employed to process the sub-graphs for a given item  $i$ . Chen et al. [17] employs KG inside the actor-critic algorithm to help the agent learn the policy. Specifically, the critic network contains a GCN layer to give weight to the graph and conduct searches in the graph to find an optimal path and hence guide the optimization of policy learning. However, such a method is relatively computationally expensive as it requires jointly training the GCN and the actor-critic network. Gong et al. [31] adopts a Graph Attention Network (GAT) [92] into the actor-critic network to conduct recommendation. In addition, the GAT is used as an encoder to obtain a state representation.

A common way of using GCN or its variants in DRL-based RS is the state encoder. The related challenge is the difficulty for the environment to provide a graph-like input to the GCN.

#### 4.4 Self-Supervised DRL-based RS

**Self-supervised learning (SSL) is a technique in which the model is trained by itself without external label information.** SSL-DRL is receiving growing interest in robotics [47, 112]. Kahn et al. [47] shows that SSL can be used to learn the policy when doing navigation by providing real-world experience. Zeng et al. [112] demonstrates that SSL-DRL can be used to help the agent learn synergies between two similar policies, thus empowering the agent to conduct two different tasks. Recent advances in SSL RL show that SSL can also provide interpretability for RL, which is promising for interpretable RS research [85]. Shi et al. [85] shows that SSL based RL can highlight the task-relevant information to guide the agent's behavior. Moreover, Xin et al. [106] shows that SSL can be used to provide negative feedback for DRL-based RS to improve recommendation performance. To be specific, a self-supervised loss function is appended into the normal DRL loss function,

$$- \sum_{i=1}^n Y_i \log \left( \frac{e^{y_i}}{\sum_{i'=1}^n e^{y_{i'}}} \right) + L_{DRL} \quad (37)$$

where  $Y_i$  is an indicator function to show users interact with the item  $i$  or not.  $L_{DRL}$  could vary, if the DQN is adopted, Equation (4) should be used. SSL demonstrates promising performance in visual representation in recent years, which would be a possible solution to generate the state representation as there are a few DRL-based RS studies that adopt CNNs to process image-like data and transform it into a state [28, 59]. Furthermore, as an unsupervised learning approach, SSL would provide a new direction about defining the reward function by learning common patterns between different states as well as multi-task learning.

## 5 OPEN QUESTIONS

In this section, we outline several open questions and challenges that exist in DRL-based RS research. We believe these issues could be critical for the future development of DRL-based RS.

## 5.1 Sample Efficiency

Sample inefficiency is a well-known challenge in model-free DRL methods. Model-free DRL requires a significant number of samples as there is no guarantee that the received state is useful. Normally, after a substantial number of episodes, the agent may start learning as the agent finally receives a useful state and reward signal. A common solution is the experience replay technique, which only works in off-policy methods. Experience replay still suffers the sample inefficiency problem [77] as not every past experience is worth replaying. Isele and Cosgun [43] propose selected experience replay (SER) that only stores valuable experience into the replay buffer and thus improves sample efficiency. While traditional DRL environments only contain several<sup>3</sup> candidate items, in DRL-based RS, the agent must deal with a significantly larger action space as RS may contain lots of candidate items. Existing DRL-based RS studies on traditional experience replay methods often demonstrate slow converge speed. Chen et al. [14] design a user model to improve the sample efficiency through auxiliary learning. Specifically, they apply the auxiliary loss with the state representation, and the model distinguishes low-activity users and asks the agent to update the recommendation policy based on high-activity users more frequently.

On the other hand, model-based methods are more sample efficient. However, they introduce extra complexity as the agent is required to learn the environment model as well as the policy. Due to the extremely large action space and possibly large state space (depending on users' contextual information) in RS, approximating the environment model and policy simultaneously becomes challenging.

## 5.2 Exploration and Exploitation

The exploration and exploitation dilemma is a fundamental and challenging problem in reinforcement learning research and receives lots of attention in DRL. This dilemma describes a trade-off between obtaining new knowledge and the need to use that knowledge to improve performance. Many DQN-based methods focus on exploration before the replay buffer is full and exploitation afterward. Consequently, it requires an extremely large replay buffer to allow all possibilities in recommendation can be stored. DRN employs Dueling Bandit Gradient Descent (DBGD) [111] to encourage exploration while [18, 36] introduces a regularization or entropy term into the objective function to do so. [42] uses the sheer size of the action space to ensure sufficient exploration. [98, 100, 103] uses a separate KG or elaborated graph exploration operation to conduct exploration. [13] employs Boltzmann exploration to get the benefit of exploratory data without negatively impacting user experience. In addition,  $\epsilon$ -greedy is the most common technique used to encourage exploration [9, 52–54, 59, 96, 105, 134, 135]. Remaining studies rely on a simulator to conduct exploration. However, it may suffer from noise and over-fitting [105] because of the gap between simulation and real online application. For most DRL-based methods such as vanilla DQN, policy gradient, or actor-critic-based methods,  $\epsilon$ -greedy would be a good choice for exploration. In addition, injecting noise into the action space would also be helpful for those actor-critic-based methods [56]. For methods involving KGs,  $\epsilon$ -greedy may help, but the elaborated graph exploration methods may receive better performance.

## 5.3 Generalizing from Simulation to Real-World Recommendation

Existing work generally trains DRL algorithms in simulation environments or offline datasets. Deploying DRL algorithms into real applications is challenging due to the gap between simulation and real-world applications. Simulation environments do not contain domain knowledge or social impact. They can not cover the domain knowledge and task-specific engineering in the real-world

<sup>3</sup>For example, the number of actions in MuJoCo is less than one hundred.

recommendation. How to bridge the gap between simulation and real applications is a challenging topic. Sim2real [122] is a transfer learning approach that transfers DRL policies from simulation environments to reality. Sim2real uses domain adaption techniques to help agents transfer learned policy. Specifically, it adopts GANs to conduct adaption by generating different samples. RL-CycleGAN [74] is a sim2real method for vision-based tasks. It uses CycleGAN [132] to conduct pixel-level domain adaption. Specifically, it maintains cycle consistency during GAN training and encourages the adapted image to retain certain attributes of the input image. In DRL-based RS, sim2real would be a possible solution for generalizing the learned policy from simulation environments to reality. However, sim2real is a new technique still under exploration. It shows an adequate capability in simple tasks and requires more effort to handle the complex task such as recommendation. We believe it is a workable solution for generalizing from simulation to reality.

#### 5.4 Bias (Unfairness)

Chen et al. [12] observe that user behavior data are not experimental but observational, which leads to problems of bias and unfairness.

There are two reasons why bias is so common. First, the inherent characteristic of user behavior data is not experimental but observational. In other words, data that are fed into recommender systems are subject to selection bias [78]. For instance, users in a video recommendation system tend to watch, rate, and comment on those movies that they are interested in. Second, a distribution discrepancy exists, which means the distributions of users and items in the recommender system are not even. Recommender systems may suffer from 'popularity bias', where popular items are recommended far more frequently than the others. However, the ignored products in the "long tail" can be equally critical for businesses as they are the ones less likely to be discovered.

Friedman and Nissenbaum [26] denote the unfairness as that the system systematically and unfairly discriminates against certain individuals or groups of individuals in favor of others.

A large number of studies explore dynamic recommendation systems by utilizing the agent mechanism in reinforcement learning (RL), considering the information seeking and decision-making as sequential interactions. How to evaluate a policy efficiently is a big challenge for RL-based recommenders. Online A/B tests are not only expensive and time-consuming but also sometimes hurt the user experience. Off-policy evaluation is an alternative strategy that historical user behavior data are used to evaluate the policy. However, user behavior data are biased, as mentioned before, which causes a gap between the policy of RL-based RS and the optimal policy.

To eliminate the effects of bias and unfairness, Chen et al. [13] use the inverse of the probability of historical policy to weight the policy gradients. Huang et al. [40] introduce a debiasing step that corrects the biases presented in the logged data before it is used to simulate user behavior. Zou et al. [135] propose to build a customer simulator that is designed to simulate the environment and handle the selection bias of logged data.

#### 5.5 Explainability

Although deep learning-based models can generally improve the performance of recommender systems, they are not easily interpretable. As a result, it becomes an important task to make recommender results explainable, along with providing high-quality recommendations. High explainability in recommender systems not only helps end-users understand the items recommended but also enables system designers to check the internal mechanisms of recommender systems. Zhang and Chen [118] review different information sources and various types of models that can facilitate explainable recommendation. Attention mechanisms and knowledge graph techniques currently play an important role in realizing explainability in RS.

Attention models have great advantages in both enhancing predictive performance and having greater explainability [116]. Wang et al. [99] introduce a reinforcement learning framework incorporated with an attention model for explainable recommendation. Firstly, it achieves model-agnosticism by separating the recommendation model from the explanation generator. Secondly, the agents that are instantiated by attention-based neural networks can generate sentence-level explanations.

Knowledge graphs contain rich information about users and items, which can help to generate intuitive and more tailored explanations for the recommendation system [118]. Recent work has achieved greater explainability by using reinforcement and knowledge graph reasoning. The algorithm from [103] learns to find a path that navigates from users to items of interest by interacting with the knowledge graph environment. Zhao et al. [121] extract imperfect path demonstrations with minimum labeling effort and propose an adversarial actor-critic model for demonstration-guided path-finding. Moreover, it achieves better recommendation accuracy and explainability by reinforcement learning and knowledge graph reasoning.

## 5.6 Robustness on Adversarial Samples and Attacks

Adversarial samples demonstrate that deep learning-based methods are vulnerable. Hence, robustness becomes an open question for both RS and DRL. Specifically, adversarial attack and defense in RS have received a lot of attention in recent years [22] as security is crucial in RS. Moreover, DRL policies are vulnerable to adversarial perturbations to agent's observations [57]. Gleave et al. [30] provide an adversarial attack method for perturbing the observations, thus affecting the learned policy. Hence, improving the robustness is the common interest for DRL and RS, which would be a critical problem for DRL-based RS. Cao et al. [10] provide an adversarial attack detection method for DRL-based RS which uses the GRU to encode the action space into a low-dimensional space and design decoders to detect the potential attack. However, it only considers Fast Gradient Sign Method (FGSM)-based attacks and strategically-timed attacks [57]. Thus, it lacks the capability to detect other types of attack. Moreover, it only provides the detection method while the defence is still an opening question.

We believe zero-shot learning techniques would be a good direction for training a universal adversarial attack detector. For defence, it is still an open question for DRL-based RS, though recent advances in adversarial defence in DRL may provide some insights [16, 63, 93].

## 6 FUTURE DIRECTIONS

In this section, we provide a few potential future directions of DRL-based RS. Benefiting from recent advances in DRL research, we believe those topics can boost the progress of DRL-based RS research.

### 6.1 Causal and Counterfactual Inference

Causality is a generic relationship between a cause and effect. Moreover, inferring causal effects is a fundamental problem in many applications like computational advertising, search engines, and recommender systems [7].

Recently, some researchers have connected reinforcement learning with learning causality to improve the effects for solving sequential decision-making problems. Besides, Learning agents in reinforcement learning frameworks face a more complicated environment where a large number of heterogeneous data are integrated. From our point of view, causal relationships would be capable of improving the recommendation results by introducing the directionality of cause and the effect. The users' previous choices have impact on the subsequent actions. This can be cast as an interventional data generating the dynamics of recommender systems. By viewing a policy in RL as an intervention,



we can detect unobserved confounders in RL and choose a policy on the expected reward to better estimate the causal effect [82]. Some studies improve RL models with causal knowledge as side information. Another line of work uses causal inference methods to achieve unbiased reward prediction [34].

Yang et al. [108] propose a Causal Inference Q-network which introduces observational inference into DRL by applying extra noise and uncertain inventions to improve resilience. Specifically, in this work, noise and uncertainty are added into the state space during the training state, and the agent is required to learn a causal inference model by considering the perturbation. Dasgupta et al. [20] give the first demonstration that model-free reinforcement learning can be used for causal reasoning. They explore meta-reinforcement learning to solve the problem of causal reasoning. The agents trained by a recurrent network able to make causal inferences from observational data and output counterfactual predictions. Forney et al. [25] bridge RL and causality by data-fusion for reinforcement learners. Specifically, online agents combine observations, experiments and counterfactual data to learn about the environment, even if unobserved confounders exist. Similarly, Gasse et al. [29] make the model-based RL agents work in a causal way to explore the environment under the Partially-Observable Markov Decision Process (POMDP) setting. They consider interventional data and observational data jointly and interpret model-based reinforcement learning as a causal inference problem. In this way, they bridge the gap between RL and causality by relating common concepts in RL and causality.

Regarding explainability in RL, Madumal et al. [65] propose to explain the behavior of agents in reinforcement learning with the help of causal science. The authors encode causal relationships and learn a structural causal model in RL, which is used to generate explanations based on counterfactual analysis. With counterfactual exploration, this work is able to generate two contrastive explanations for ‘why’ and ‘why not’ questions.

It is so important to search for a Directed Acyclic Graph (DAG) in causal discovery. Considering traditional methods rely on local heuristics and predefined score functions, Zhu et al. [133] propose to use reinforcement learning to search DAG for causal discovery. They use observational data as an input, RL agents as a search strategy and output the causal graph generated from an encoder-decoder NN model.

## 6.2 Offline DRL and Meta DRL

Recommender systems often need to deal with multiple scenarios such as joint recommendation and adverting, offline DRL and meta DRL provide a promising direction for achieving multiple scenarios at the same time.

Offline DRL is a new paradigm of DRL that can be combined with existing methods such as self-supervised learning and transfer learning to move toward real-world settings. Offline DRL [55] (also known as batch DRL) is designed for tasks which contain huge amounts of data. Given a large dataset that contains past interactions, offline DRL uses the dataset for training across many epochs but does not interact with the environment. Offline DRL provides a solution that can be generalized to new scenarios as it was trained by a large sized dataset. Such generalization ability is critical to RSs, which may need to deal with multiple scenarios or multiple customers. While offline DRL could provide a new direction for DRL-based RS, it still faces a few problems regarding handling the distributional shifts between existing datasets and real-world interactions.

Meta DRL [95] is defined as meta learning in the filed of DRL. Meta DRL is another approach to help agents to generalize to new tasks or environments. Different from offline DRL, meta DRL contains a memory unit which is formed by the recurrent neural network to memorize the common knowledge for different tasks. Different from offline DRL, meta DRL does not require a large amount of data to train.

### 6.3 Further Developments in Actor-Critic Methods

An actor-critic method uses the traditional policy gradient method, which suffers from the high variance problem due to the gap between behavior policy (i.e., the policy that is being used by an agent for action select) and target policy (i.e., the policy that an agent is trying to learn). A method commonly used to relieve the high variance problem is Advantage Actor-critic (A2C). Different from traditional actor-critic methods, A2C uses an advantage function to replace the Q-function inside the critic network. The advantage function  $A(s_t)$  is defined as the expected value of the TD-error. The new objective function for policy gradient can be written as,

$$\mathbb{E}_{\tau \sim d_{\pi_\theta}} \left[ \sum_{t=1}^T \underbrace{(Q(s_t, a_t) - V(s_t))}_{A(s_t)} \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(s_t, a_t) \right]. \quad (38)$$

However, A2C still uses DDPG as the main training algorithm, which may suffer function approximation errors when estimating the Q value. Twin-Delayed DDPG (TD3) [27] is designed to improve the function approximation problem in DDPG which uses clipped double Q-learning to update the critic. The gradient update can be expressed as,

$$\mathbb{E}_{\tau \sim d_{\pi_\theta}} \left[ \sum_{t=1}^T r(s_t, a_t) + \gamma \min(Q_1(s_t, a_t + \epsilon), Q_2(s_t, a_t + \epsilon)) \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(s_t, a_t) \right]. \quad (39)$$

where  $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma, -c, c))$ ,  $\sigma$  is the standard deviation and  $c$  is a constant for clipping.

Another two ways to improve actor-critic methods are Trust Region Policy Optimization (TRPO) [79] and Proximal Policy Optimization (PPO) [80], which focus on modification of the advantage function. TRPO aims to limit the step size for each gradient to ensure it will not change too much. The core idea is to add a constraint to the advantage function,

$$\frac{\pi(a|s)}{\pi_{old}(a|s)} A(s), \quad (40)$$

where the KL divergence will be used to measure the distance between the current policy and the old policy is small enough. PPO has the same goal as TRPO which is to try to find the biggest possible improvement step on a policy using the current data. PPO is a simplified version of TRPO which introduces the clip operation,

$$\min \left( \frac{\pi(a|s)}{\pi_{old}(a|s)} A(s), \text{clip} \left( \frac{\pi(a|s)}{\pi_{old}(a|s)} A(s), 1 - \epsilon, 1 + \epsilon \right) A(s) \right). \quad (41)$$

Soft Actor-Critic (SAC) [35] is another promising variant of the actor-critic algorithm and is widely used in DRL research. SAC uses the entropy term to encourage the agent to explore, which could be a possible direction to solve the exploration and exploitation dilemma. Moreover, SAC assigns an equal probability to actions that are equally attractive to the agent to capture those near-optimal policies. An example of related work [36] uses SAC to improve the stability of the training process in RS.

## 7 CONCLUSION

In this survey, we provide a comprehensive overview the use of deep reinforcement learning in recommender systems. We introduce a classification scheme for existing studies and discuss them by category. We also provide an overview of such existing emerging topics and point out a few promising directions. We hope this survey can provide a systematic understanding of the key concepts in DRL-based RS and valuable insights for future research.

## REFERENCES

- [1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. In *International Conference on Machine Learning*. PMLR, 22–31.
- [2] M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2021. Reinforcement learning based recommender systems: A survey. *arXiv preprint arXiv:2101.06286* (2021).
- [3] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.
- [4] Xueying Bai, Jian Guan, and Hongning Wang. 2019. A Model-Based Reinforcement Learning with Adversarial Training for Online Recommendation. *Advances in Neural Information Processing Systems* 32 (2019), 10735–10746.
- [5] Ioana Bica, Daniel Jarrett, Alihan Hüyük, and Mihaela van der Schaar. 2020. Learning" What-if" Explanations for Sequential Decision-Making. In *International Conference on Learning Representations*.
- [6] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NIPS)*. 1–9.
- [7] Léon Bottou, Jonas Peters, Joaquin Quiñero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual Reasoning and Learning Systems: The Example of Computational Advertising. *Journal of Machine Learning Research* 14, 11 (2013).
- [8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).
- [9] Qingpeng Cai, Aris Filos-Ratsikas, Pingzhong Tang, and Yiwei Zhang. 2018. Reinforcement Mechanism Design for e-commerce. In *Proceedings of the 2018 World Wide Web Conference*. 1339–1348.
- [10] Yuanjiang Cao, Xiaocong Chen, Lina Yao, Xianzhi Wang, and Wei Emma Zhang. 2020. Adversarial Attacks and Detection on Reinforcement Learning-Based Interactive Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1669–1672.
- [11] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. 2019. Large-scale interactive recommendation with tree-structured policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3312–3320.
- [12] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2020. Bias and Debias in Recommender System: A Survey and Future Directions. *arXiv:arXiv:2010.03240*
- [13] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 456–464.
- [14] Minmin Chen, Bo Chang, Can Xu, and Ed H Chi. 2021. User Response Models to Improve a REINFORCE Recommender System. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 121–129.
- [15] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. 2018. Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1187–1196.
- [16] Tong Chen, Jiqiang Liu, Yingxiao Xiang, Wenjia Niu, Endong Tong, and Zhen Han. 2019. Adversarial attack and defense in reinforcement learning-from AI security view. *Cybersecurity* 2, 1 (2019), 1–22.
- [17] Xiaocong Chen, Chaoran Huang, Lina Yao, Xianzhi Wang, Wenjie Zhang, et al. 2020. Knowledge-guided deep reinforcement learning for interactive recommendation. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
- [18] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. 2019. Generative adversarial user model for reinforcement learning based recommendation system. In *International Conference on Machine Learning*. PMLR, 1052–1061.
- [19] Xiaocong Chen, Lina Yao, Aixin Sun, Xianzhi Wang, Xiwei Xu, and Liming Zhu. 2020. Generative Inverse Deep Reinforcement Learning for Online Recommendation. *arXiv preprint arXiv:2011.02248* (2020).
- [20] Ishita Dasgupta, Jane Wang, Silvia Chiappa, Jovana Mitrovic, Pedro Ortega, David Raposo, Edward Hughes, Peter Battaglia, Matthew Botvinick, and Zeb Kurth-Nelson. 2019. Causal reasoning from meta-reinforcement learning. *arXiv preprint arXiv:1901.08162* (2019).
- [21] Thomas Degris, Martha White, and Richard S Sutton. 2012. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839* (2012).
- [22] Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. 2021. A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks. *ACM Computing Surveys (CSUR)* 54, 2 (2021), 1–38.
- [23] Maxim Egorov. 2016. Multi-agent deep reinforcement learning. *CS231n: convolutional neural networks for visual recognition* (2016), 1–8.
- [24] Jun Feng, Heng Li, Minlie Huang, Shichen Liu, Wenwu Ou, Zhirong Wang, and Xiaoyan Zhu. 2018. Learning to collaborate: Multi-scenario ranking via multi-agent reinforcement learning. In *Proceedings of the 2018 World Wide*

*Web Conference*. 1939–1948.

- [25] Andrew Forney, Judea Pearl, and Elias Bareinboim. 2017. Counterfactual data-fusion for online reinforcement learners. In *International Conference on Machine Learning*. PMLR, 1156–1164.
- [26] Batya Friedman and Helen Nissenbaum. 1996. Bias in computer systems. *ACM Transactions on Information Systems (TOIS)* 14, 3 (1996), 330–347.
- [27] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*. PMLR, 1587–1596.
- [28] Rong Gao, Haifeng Xia, Jing Li, Donghua Liu, Shuai Chen, and Gang Chun. 2019. DRCGR: Deep reinforcement learning framework incorporating CNN and GAN-based for interactive recommendation. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1048–1053.
- [29] Maxime Gasse, Damien Grasset, Guillaume Gaudron, and Pierre-Yves Oudeyer. 2021. Causal Reinforcement Learning using Observational and Interventional Data. *arXiv:2106.14421 [cs.LG]*
- [30] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. 2020. Adversarial Policies: Attacking Deep Reinforcement Learning. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=HJgEMpVFwB>
- [31] Yu Gong, Yu Zhu, Lu Duan, Qingwen Liu, Ziyu Guan, Fei Sun, Wenwu Ou, and Kenny Q Zhu. 2019. Exact-k recommendation via maximal clique optimization. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 617–626.
- [32] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661* (2014).
- [33] Tao Gui, Peng Liu, Qi Zhang, Liang Zhu, Minlong Peng, Yunhua Zhou, and Xuanjing Huang. 2019. Mention recommendation in Twitter with cooperative multi-agent reinforcement learning. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 535–544.
- [34] Ruocheng Guo, Lu Cheng, Jundong Li, P. Richard Hahn, and Huan Liu. 2020. A Survey of Learning Causality with Data: Problems and Methods. *ACM Comput. Surv.* 53, 4, Article 75 (July 2020), 37 pages. <https://doi.org/10.1145/3397269>
- [35] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.
- [36] Xu He, Bo An, Yanghua Li, Haikai Chen, Rundong Wang, Xinrun Wang, Runsheng Yu, Xin Li, and Zhirong Wang. 2020. Learning to Collaborate in Multi-Module Recommendation via Multi-Agent Reinforcement Learning without Communication. In *Fourteenth ACM Conference on Recommender Systems*. 210–219.
- [37] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476* (2016).
- [38] Daocheng Hong, Yang Li, and Qiwen Dong. 2020. Nonintrusive-Sensing and Reinforcement-Learning Based Adaptive Personalized Music Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1721–1724.
- [39] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 368–377.
- [40] Jin Huang, Harrie Oosterhuis, Maarten de Rijke, and Herke van Hoof. 2020. Keeping Dataset Biases out of the Simulation: A Debaised Simulator for Reinforcement Learning based Recommender Systems. In *Fourteenth ACM Conference on Recommender Systems*. 190–199.
- [41] Eugene Ie, Chih-wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. 2019. Recsim: A configurable simulation platform for recommender systems. *arXiv preprint arXiv:1909.04847* (2019).
- [42] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. 2019. SlateQ: A Tractable Decomposition for Reinforcement Learning with Recommendation Sets. In *Proceedings of the Twenty-eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*. Macau, China, 2592–2599.
- [43] David Isele and Akansel Cosgun. 2018. Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
- [44] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. 2016. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397* (2016).
- [45] Shengcong Ji, Zhaoyuan Wang, Tianrui Li, and Yu Zheng. 2020. Spatio-temporal feature fusion for dynamic taxi route recommendation via deep reinforcement learning. *Knowledge-Based Systems* 205 (2020), 106302.
- [46] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. 2020. Graph Convolutional Reinforcement Learning. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=HkxdQkSYDB>

- [47] Gregory Kahn, Adam Villafior, Bosen Ding, Pieter Abbeel, and Sergey Levine. 2018. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5129–5136.
- [48] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.
- [49] Vijay R Konda and John N Tsitsiklis. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*. Citeseer, 1008–1014.
- [50] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. 2019. Discriminator-Actor-Critic: Addressing Sample Inefficiency and Reward Bias in Adversarial Imitation Learning. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=Hk4fpoA5Km>
- [51] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems* 29 (2016), 3675–3683.
- [52] Yu Lei and Wenjie Li. 2019. Interactive recommendation with user-specific deep reinforcement learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 13, 6 (2019), 1–15.
- [53] Yu Lei, Hongbin Pei, Hanqi Yan, and Wenjie Li. 2020. Reinforcement learning based recommendation with graph convolutional q-network. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1757–1760.
- [54] Yu Lei, Zhitao Wang, Wenjie Li, and Hongbin Pei. 2019. Social Attentive Deep Q-network for Recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1189–1192.
- [55] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).
- [56] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).
- [57] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. 2017. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748* (2017).
- [58] Feng Liu, Huifeng Guo, Xutao Li, Ruiming Tang, Yunming Ye, and Xiuqiang He. 2020. End-to-end deep reinforcement learning based recommendation with supervised embedding. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 384–392.
- [59] Feng Liu, Ruiming Tang, Huifeng Guo, Xutao Li, Yunming Ye, and Xiuqiang He. 2020. Top-aware reinforcement learning based recommendation. *Neurocomputing* 417 (2020), 255–269.
- [60] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, and Yuzhou Zhang. 2018. Deep reinforcement learning based recommendation with explicit user-item interactions modeling. *arXiv preprint arXiv:1810.12027* (2018).
- [61] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, Yuzhou Zhang, and Xiuqiang He. 2020. State representation modeling for deep reinforcement learning based recommendation. *Knowledge-Based Systems* 205 (2020), 106170.
- [62] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. 2015. Recommender system application developments: a survey. *Decision Support Systems* 74 (2015), 12–32.
- [63] Björn Lütjens, Michael Everett, and Jonathan P How. 2020. Certified adversarial robustness for deep reinforcement learning. In *Conference on Robot Learning*. PMLR, 1328–1337.
- [64] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Ji Yang, Minmin Chen, Jiaxi Tang, Lichan Hong, and Ed H Chi. 2020. Off-policy learning in two-stage recommender systems. In *Proceedings of The Web Conference 2020*. 463–473.
- [65] Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. 2020. Explainable reinforcement learning through a causal lens. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 2493–2500.
- [66] Rajbala Makar, Sridhar Mahadevan, and Mohammad Ghavamzadeh. 2001. Hierarchical multi-agent reinforcement learning. In *Proceedings of the fifth international conference on Autonomous agents*. 246–253.
- [67] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.
- [68] Ali MontazerAlghaem, Hamed Zamani, and James Allan. 2020. A Reinforcement Learning Framework for Relevance Feedback. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 59–68.
- [69] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, Vol. 99. 278–287.
- [70] Andrew Y Ng, Stuart J Russell, et al. 2000. Algorithms for inverse reinforcement learning. In *Icml*, Vol. 1. 2.

- [71] Richard O Oyeleke, Chen-Yeou Yu, and Carl K Chang. 2018. Situ-centric reinforcement learning for recommendation of tasks in activities of daily living in smart homes. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 2. IEEE, 317–322.
- [72] Feiyang Pan, Qingpeng Cai, Pingzhong Tang, Fuzhen Zhuang, and Qing He. 2019. Policy gradients for contextual recommendations. In *The World Wide Web Conference*. 1421–1431.
- [73] Changhua Pei, Xinru Yang, Qing Cui, Xiao Lin, Fei Sun, Peng Jiang, Wenwu Ou, and Yongfeng Zhang. 2019. Value-aware recommendation based on reinforcement profit maximization. In *The World Wide Web Conference*. 3123–3129.
- [74] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. 2020. RI-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11157–11166.
- [75] David Rohde, Stephen Bonner, Travis Dunlop, Flavian Vasile, and Alexandros Karatzoglou. 2018. Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising. *arXiv preprint arXiv:1808.00720* (2018).
- [76] Marlesson RO Santana, Luckeciano C Melo, Fernando HF Camargo, Bruno Brandão, Anderson Soares, Renan M Oliveira, and Sandor Caetano. 2020. MARS-Gym: A Gym framework to model, train, and evaluate Recommender Systems for Marketplaces. *arXiv preprint arXiv:2010.07035* (2020).
- [77] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015).
- [78] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. *arXiv:arXiv:1602.05352*
- [79] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.
- [80] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
- [81] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. 2018. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1134–1141.
- [82] Wenjie Shang, Yang Yu, Qingyang Li, Zhiwei Qin, Yiping Meng, and Jieping Ye. 2019. Environment reconstruction with hidden confounders for reinforcement learning based recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 566–576.
- [83] Bichen Shi, Makbule Gulcin Ozsoy, Neil Hurley, Barry Smyth, Elias Z Tragos, James Geraci, and Aonghus Lawlor. 2019. PyRecGym: a reinforcement learning gym for recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 491–495.
- [84] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. 2019. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4902–4909.
- [85] Wenjie Shi, Gao Huang, Shiji Song, Zhuoyuan Wang, Tingyu Lin, and Cheng Wu. 2020. Self-Supervised Discovering of Interpretable Features for Reinforcement Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).
- [86] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *International conference on machine learning*. PMLR, 387–395.
- [87] Ying Sun, Fuzhen Zhuang, Hengshu Zhu, Qing He, and Hui Xiong. 2021. Cost-Effective and Interpretable Job Skill Recommendation with Deep Reinforcement Learning. In *Proceedings of the Web Conference 2021*. 3827–3838.
- [88] Ryuichi Takanobu, Tao Zhuang, Minlie Huang, Jun Feng, Haihong Tang, and Bo Zheng. 2019. Aggregating e-commerce search results from heterogeneous sources via hierarchical reinforcement learning. In *The World Wide Web Conference*. 1771–1781.
- [89] Jiayi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.
- [90] Arash Tavakoli, Fabio Pardo, and Petar Kormushev. 2018. Action branching architectures for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
- [91] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.
- [92] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
- [93] Feng Wang, Chen Zhong, M CenK Gursay, and Senem Velipasalar. 2020. Defense strategies against adversarial jamming attacks via deep reinforcement learning. In *2020 54th annual conference on information sciences and systems (CISS)*. IEEE, 1–6.

- [94] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *The world wide web conference*. 3307–3313.
- [95] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dhharshan Kumaran, and Matt Botvinick. 2016. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763* (2016).
- [96] Kai Wang, Zhene Zou, Qilin Deng, Runze Wu, Jianrong Tao, Changjie Fan, Liang Chen, and Peng Cui. 2021. Reinforcement Learning with a Disentangled Universal Value Function for Item Recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4427–4435.
- [97] Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. 2018. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2447–2456.
- [98] Pengfei Wang, Yu Fan, Long Xia, Wayne Xin Zhao, Shaozhang Niu, and Jimmy Huang. 2020. KERL: A knowledge-guided reinforcement learning model for sequential recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 209–218.
- [99] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. 2018. A reinforcement learning framework for explainable recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 587–596.
- [100] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced negative sampling over knowledge graph for recommendation. In *Proceedings of The Web Conference 2020*. 99–109.
- [101] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.
- [102] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.
- [103] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 285–294.
- [104] Yilin Xiao, Liang Xiao, Xiaozhen Lu, Hailu Zhang, Shui Yu, and H Vincent Poor. 2020. Deep reinforcement learning based user profile perturbation for privacy aware recommendation. *IEEE Internet of Things Journal* (2020).
- [105] Ruobing Xie, Shaoliang Zhang, Rui Wang, Feng Xia, and Leyu Lin. 2021. Hierarchical Reinforcement Learning for Integrated Recommendation. In *Proceedings of AAAI*.
- [106] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2020. Self-Supervised Reinforcement Learning for Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 931–940.
- [107] Jun Xu, Zeng Wei, Long Xia, Yanyan Lan, Dawei Yin, Xueqi Cheng, and Ji-Rong Wen. 2020. Reinforcement Learning to Rank with Pairwise Policy Gradient. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 509–518.
- [108] Chao-Han Huck Yang, I Hung, Te Danny, Yi Ouyang, and Pin-Yu Chen. 2021. Causal Inference Q-Network: Toward Resilient Reinforcement Learning. *arXiv preprint arXiv:2102.09677* (2021).
- [109] Zhaoyang Yang, Kathryn Merrick, Lianwen Jin, and Hussein A Abbass. 2018. Hierarchical deep reinforcement learning for continuous action control. *IEEE transactions on neural networks and learning systems* 29, 11 (2018), 5174–5184.
- [110] Tong Yu, Yilin Shen, Ruiyi Zhang, Xiangyu Zeng, and Hongxia Jin. 2019. Vision-language recommendation via attribute augmented multimodal reinforcement learning. In *Proceedings of the 27th ACM International Conference on Multimedia*. 39–47.
- [111] Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 1201–1208.
- [112] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. 2018. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 4238–4245.
- [113] Jing Zhang, Bowen Hao, Bo Chen, Cuiping Li, Hong Chen, and Jimeng Sun. 2019. Hierarchical reinforcement learning for course recommendation in MOOCs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 435–442.
- [114] Ruiyi Zhang, Tong Yu, Yilin Shen, Hongxia Jin, and Changyou Chen. 2019. Text-Based Interactive Recommendation via Constraint-Augmented Reinforcement Learning. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2019/file/52130c418d4f02c74f74a5bc1f8020b2-Paper.pdf>
- [115] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.
- [116] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1, Article 5 (Feb. 2019), 38 pages. <https://doi.org/10.1145/3285029>

- [117] Weijia Zhang, Hao Liu, Fan Wang, Tong Xu, Haoran Xin, Dejing Dou, and Hui Xiong. 2021. Intelligent Electric Vehicle Charging Recommendation Based on Multi-Agent Reinforcement Learning. In *Proceedings of the Web Conference 2021*. 1856–1867.
- [118] Yongfeng Zhang and Xu Chen. 2020. Explainable Recommendation: A Survey and New Perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101. <https://doi.org/10.1561/15000000066>
- [119] Yang Zhang, Chenwei Zhang, and Xiaozhong Liu. 2017. Dynamic scholarly collaborator recommendation via competitive multi-agent reinforcement learning. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 331–335.
- [120] Dongyang Zhao, Liang Zhang, Bo Zhang, Lizhou Zheng, Yongjun Bao, and Weipeng Yan. 2020. MaHRL: Multi-goals Abstraction Based Deep Hierarchical Reinforcement Learning for Recommendations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 871–880.
- [121] Kangzhi Zhao, Xiting Wang, Yuren Zhang, Li Zhao, Zheng Liu, Chunxiao Xing, and Xing Xie. 2020. Leveraging Demonstrations for Reinforcement Recommendation Reasoning over Knowledge Graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 239–248.
- [122] Wenshuai Zhao, Jorge Peña Queraltá, and Tomi Westerlund. 2020. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 737–744.
- [123] Xiangyu Zhao, Changsheng Gu, Haoshenglun Zhang, Xiwang Yang, Xiaobing Liu, Hui Liu, and Jiliang Tang. 2021. DEAR: Deep Reinforcement Learning for Online Advertising Impression in Recommender Systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 750–758.
- [124] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 95–103.
- [125] Xiangyu Zhao, Long Xia, Lixin Zou, Hui Liu, Dawei Yin, and Jiliang Tang. 2020. Whole-Chain Recommendations. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1883–1891.
- [126] Xiangyu Zhao, Long Xia, Lixin Zou, Dawei Yin, and Jiliang Tang. 2019. Toward simulating environments in reinforcement learning based recommendations. *arXiv preprint arXiv:1906.11462* (2019).
- [127] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1040–1048.
- [128] Xiangyu Zhao, Liang Zhang, Long Xia, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2019. Deep reinforcement learning for list-wise recommendations. *DRL4KDD '19* (2019).
- [129] Xiangyu Zhao, Xudong Zheng, Xiwang Yang, Xiaobing Liu, and Jiliang Tang. 2020. Jointly learning to recommend and advertise. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3319–3327.
- [130] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*. 167–176.
- [131] Sijin Zhou, Xinyi Dai, Haokun Chen, Weinan Zhang, Kan Ren, Ruiming Tang, Xiuqiang He, and Yong Yu. 2020. Interactive recommender system via knowledge graph-enhanced reinforcement learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 179–188.
- [132] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.
- [133] Shengyu Zhu, Ignavier Ng, and Zhitang Chen. 2019. Causal discovery with reinforcement learning. *arXiv preprint arXiv:1906.04477* (2019).
- [134] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement learning to optimize long-term user engagement in recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2810–2818.
- [135] Lixin Zou, Long Xia, Pan Du, Zhuo Zhang, Ting Bai, Weidong Liu, Jian-Yun Nie, and Dawei Yin. 2020. Pseudo Dyna-Q: A reinforcement learning framework for interactive recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 816–824.