A Practical Reinforcement Learning Framework for Automatic Radar Detection

YU Junpeng[1], CHEN Yiyu[2]

(1. Nanjing Research Institute of Electronics Technology, Nanjing 210039, China;
2. Nanjing University, Nanjing 210023, China)

DOI: 10.12142/ZTECOM.202303004

https://kns.cnki.net/kcms/detail/34.1294.TN.20230802.1625.002.html, published online August 3, 2023

Manuscript received: 2023-06-16

Abstract: At present, the parameters of radar detection rely heavily on manual adjustment and empirical knowledge, resulting in low automation. Traditional manual adjustment methods cannot meet the requirements of modern radars for high efficiency, high precision, and high automation. Therefore, it is necessary to explore a new intelligent radar control learning framework and technology to improve the capability and automation of radar detection. Reinforcement learning is popular in decision task learning, but the shortage of samples in radar control tasks makes it difficult to meet the requirements of reinforcement learning. To address the above issues, we propose a practical radar operation reinforcement learning framework, and integrate offline reinforcement learning and meta-reinforcement learning methods to alleviate the sample requirements of reinforcement learning. Experimental results show that our method can automatically perform as humans in radar detection with real-world settings, thereby promoting the practical application of reinforcement learning in radar operation.

Keywords: meta-reinforcement learning; radar detection; reinforcement learning; offline reinforcement learning

Citation (Format 1): YU J P, CHEN Y Y. A practical reinforcement learning framework for automatic radar detection [J]. ZTE Communications, 2023, 21(3): 22 - 28. DOI: 10.12142/ZTECOM.202303004

Citation (Format 2): J. P. Yu and Y. Y. Chen, "A practical reinforcement learning framework for automatic radar detection," *ZTE Communications*, vol. 21, no. 3, pp. 22 - 28, Sept. 2023. doi: 10.12142/ZTECOM.202303004.

1 Introduction

The advent of modern radar systems has brought forth a demand for higher efficiency, precision, and automation[1]. However, the current radar detection parameters heavily depend on manual adjustment and empirical knowledge, which significantly hampers automation[2]. Traditional manual adjustment methods are increasingly inadequate to meet these growing demands. This inadequacy necessitates the exploration of a new intelligent radar control learning framework and technology that can enhance the capability and automation of radar detection.

One promising learning approach is reinforcement learning, which has gained popularity in decision-task learning. Reinforcement learning is a major paradigm within the machine learning field, distinct from perceptual learning typified by image processing. Perceptual learning primarily involves supervised learning, while reinforcement learning seeks to address sequential decision-making problems through rewards. The reinforcement learning algorithm, based on the Bellman equation, continually learns and improves through trial and error within an environment, thereby accumulating experience and developing superior strategies for given tasks[3]. In recent years, deep reinforcement learning (DRL), with its powerful feature representation and function-fitting capabilities, has shown remarkable proficiency in various areas such as gaming and robotics. Notable accomplishments include AlphaGo's consecutive victories over human world champions in Go[4], AlphaStar's top master rank in StarCraft II[5], Suphx's rise to the top ten sections of the professional Japanese Mahjong platform "Tianfeng" developed by Microsoft Research Asia[6], and the flexible and universal tokamak magnetic controller architecture developed by the DeepMind team for nuclear fusion projects[7]. Furthermore, deep reinforcement learning has been progressively implemented across various industries.

However, the application of reinforcement learning in radar control tasks is hindered by the shortage of samples. The effectiveness of deep reinforcement learning is currently heavily reliant on the availability of extensive learning data and substantial computing resources. For instance, the chess benchmark algorithm, MuZero, requires approximately 10^6 steps of data[8] to achieve initial results in training. This process takes roughly 11 days at a sampling rate of 60 steps per second. Furthermore, DeepMind utilized 384 tensor processing units (TPUs) running

This work is supported by Science and Technology Innovation 2030 New Generation Artificial Intelligence Major Project under Grant No. 2021ZD0113303, the National Natural Science Foundation of China under Grant Nos. 62192783 and 62276128, and in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization.
The authors contribute equally to this paper.

in parallel over a span of about 44 days to complete the reinforcement learning training for AlphaStar, the StarCraft II algorithm[5]. The high training cost associated with deep reinforcement learning significantly restricts its range of applications.

This paper aims to address these challenges by proposing a practical radar operation reinforcement learning framework that integrates offline reinforcement learning and meta-reinforcement learning methods. The framework consists of the environment modeling of radar detection, the integrated learning structure and the learning objectives. Our experimental results of the MATLAB radar detection simulator indicate that the ability of our method in automatic radar detection has basically reached the level of humans, thus promoting the practical application of reinforcement learning in radar detection. This paper is structured as follows. First, in Section 2, we introduce the related works. Then, in Section 3, we demonstrate our reinforcement learning framework for automatic radar detection. In Section 4, the experimental settings and results are introduced. Finally, in Section 5, we draw a conclusion.

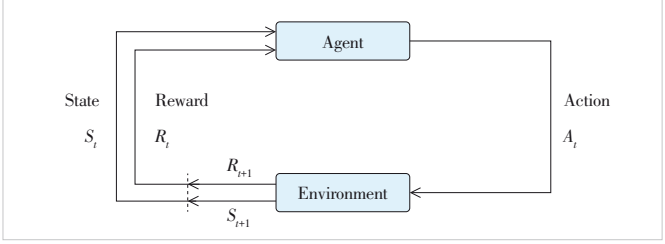
2 Related Works

In this section, we introduce the background and related works about our proposed framework, including reinforcement learning and its correlational research with radar control.

2.1 Reinforcement Learning

Reinforcement learning is one of the popular paradigms of machine learning. The framework of reinforcement learning is shown in Fig. 1, which mainly includes two parts: agent and environment. The operation of reinforcement learning is a process of continuous interaction between agents and the environment, where the environment provides agents with the current state and numerical rewards, while agents output actions to the environment according to existing information (usually the current state). The environment gives the state and rewards after the action is executed, and so forth until the environment terminates (done). In this process, agents often choose actions and learn strategies to maximize expected cumulative rewards.

The environment model of reinforcement learning is generally based on the Markov decision process (MDP). MDP is defined by a quaternion $\langle S, A, R, T \rangle$, where S is the set of environmental states, A is the set of optional actions, the state transition function $T: S \times A \times S \to [0, 1]$ gives the probability of transition from state s and action a to state s', and the reward function R: $S \times A \times S \to \mathbb{R}$ provides the reward value for each step.



▲Figure 1. Framework of reinforcement learning

The agent algorithm of reinforcement learning aims to learn a policy π, and the policy determines the execution of action a (deterministic policy) or the execution probability (non-deterministic policy) in each state s. The classical reinforcement learning algorithm considers that the MDP model of the environment is given in advance, and the optimization goal of the policy π is to maximize the expected cumulative discount reward. The parameters of the parameterization policy π_θ are θ, and the formula for calculating the optimal parameters θ^* is:

$$\theta^* = \arg\max_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t r_t\right], \tag{1}$$

where T refers to the number of time steps that the environment runs, and the discount factor $\gamma \in [0,1]$ is used to balance long-term rewards and short-term rewards. γ significantly stabilizes the reinforcement learning algorithm in an environment with excessive T.

Reinforcement learning algorithms can be divided into two categories: value function-based and policy gradient-based. The reinforcement learning algorithm based on the value function makes decisions according to the state action value function $Q^\pi(s,a)$. In the DRL algorithms based on the value function, $Q^\pi(s,a)$ is constructed by a neural network, supplemented by some designs to enhance the stability of the algorithm[9]. Note that deep networks enable policies to adapt to tasks with a much wider range. This kind of algorithm performs better in the discrete action environment, but it is difficult to expand to the continuous action environment. Common algorithms include the deep Q-network (DQN) [9], dueling double deep Q-network (D3QN)[10], deep recurrent Q-network (DRQN)[11], etc. Reinforcement learning algorithms based on policy gradients directly calculate the policy function $\pi_\theta(a|s)$ modeling and optimization. Commonly used algorithms based on policy gradients are actor-critic architectures, which perform better in continuous action environments, including deep deterministic policy gradient (DDPG) [12], proximal policy optimization (PPO) [13], soft actor-critic (SAC)[14], twin-delayed deep deterministic policy gradient (TD3)[15], etc.

While reinforcement learning algorithms have demonstrated effective performance in simulated environments, two primary challenges exist in copying this performance to real-world scenarios: 1) The inconsistency between the simulator and the actual environment, which is often referred to as the Sim2Real gap, tends to result in catastrophic failure of deploying simulator-trained policies in the real world; 2) the high cost of real-world sampling and the complexity of the real-world tasks result in a significant difference between the collected data and the actual situation. Especially in intelligent radar detection tasks, due to the large scale of the actual environment, it is difficult to collect sufficient training data that are needed. The actual environment can change greatly at any time with factors such as weather, ter-

rain, and goals, which brings learning difficulties. Therefore, we introduce two research directions that help to solve this problem: offline reinforcement learning and meta-reinforcement learning.

2.1.1 Offline Reinforcement Learning

Offline reinforcement learning is a data-driven subset of the broader reinforcement learning field. Its primary objective is to optimize the same objective as reinforcement learning. However, in this context, intelligent agents cannot use behavioral strategies to interact with the environment or gather additional data. Instead, a learning algorithm provides a static transition dataset, denoted as $D = (s, a, r, a')$, which is used to learn the most effective strategies. This approach is more akin to the standard supervised learning problem, with D serving as the policy training set. Essentially, offline reinforcement learning requires the learning algorithm to fully understand the dynamic system that underlies the Markov decision process, using a fixed dataset to formulate a policy. When this policy is applied to interact with the Markov decision process, it aims to yield the maximum cumulative return.

Several existing model-free offline reinforcement learning methods regularize the learned policy to align closely with the behavior policy. This is achieved through techniques such as distributional matching[16], support matching[17], importance sampling[18-19], and learning the lower bounds of true Q-values[20]. On the other hand, model-based algorithms learn policies by leveraging a dynamic model derived from the offline dataset. Ref. [21] directly restricts the learned policy to the behavior policy, similar to model-free algorithms. To penalize the policy for visiting states where the learned model may be incorrect, MOPO[22] and MoREL[23] adjust the learned dynamics, which ensures that the value estimates are conservative when the model uncertainty exceeds a certain threshold. To eliminate the need for uncertainty quantification, COMBO[24] combines model-based policy optimization[25] and conservative policy evaluation[20]. In this paper, we employ a distributional matching method, specifically the straightforward and effective behavior cloning (BC) method, as it simplifies the learning process of meta-reinforcement learning methods.

2.1.2 Meta-Reinforcement Learning

Meta-reinforcement learning methods learn meta-policy on multiple meta-training tasks, aiming to quickly adapt to previously unseen meta-testing tasks, and thus improving the effectiveness and generalizability of reinforcement learning methods. The process of meta-reinforcement learning mirrors that of meta-learning, which consists of two stages: the meta-training stage and the meta-testing stage. During the meta-training stage, the algorithm learns from the meta-training task and prepares the model for the next stage. In the meta-testing phase, the trained model is adaptively applied to the meta-testing task to achieve testing results. Each task corresponds to a reinforcement learning environment model, typically an MDP. The meta-training task is presented in the form of task distribution $p(T)$. At the beginning of meta-training, a certain number of meta training tasks $\{T_{\text{train}}\}$ are sampled from the task distribution $p(T)$, that is, $T_{\text{Train}} \sim p(T)$. The set of meta-training tasks may be fixed by one sampling, or may be generated repeatedly by samplings in multiple rounds of meta-training.

Existing works in this field can be broadly categorized into three types: the model-agnostic-meta-learning-based (MAML-based), recurrent-based, and context-based. Some research focuses on improving and extending the meta-learning framework MAML[26]. For instance, FINN et al. proposed a simplified algorithm FO-MAML that only uses first-order derivatives in their MAML work[26]; NICHOL et al. proposed a more versatile first-order derivative algorithm Reptile[27]; The ES-MAML algorithm proposed by SONG et al. uses an evolutionary algorithm instead of derivation in outer optimization[28]; ANTONIO et al. conducted extensive experiments and concluded on the training problem of MAML[29].

Some other research reduces the uncertainty of inferring the state from observation by memorizing the history of tasks, thus improving the performance of strategies on unknown tasks. For example, the RL^2 algorithm builds a policy model based on the recurrent neural network with memory and trains between multiple tasks[30]; MISHRA et al. combined time series convolution and soft attention mechanisms to form a new depth architecture[31]; PARISOTTO uses the transformer model as a cross episodic memory module[32].

Recent popular research extracts the task context to guide policy across various tasks. SÆMUNDSSON et al. used the Gaussian process and variational inference to model the hidden variables of tasks, combined with the model-based reinforcement learning algorithm to achieve a fast meta-training algorithm ML-GP[33]; ZINTGRAF et al.[34] and LAN et al.[35] combined the MAML algorithm with a task context encoder to improve performance; HUMPLIK et al. utilized long short-term memory (LSTM) to construct a task feature inference module and implemented algorithms similar to PEARL[36]; FAKOOR et al. used gated recurrent units as the history encoder to train their reinforcement learning algorithm meta-Q-learning (MQL) based on the multi-task objective[37]. The PD-VF algorithm proposed by RAILEANU et al. used the prediction environment cumulative reward to supervise the training task hidden variable module[38]; ZINTGRAF et al. used a variational autoencoder to train the task feature inference module and proposed the VariBAD algorithm[39]. Some studies improve the generalization ability of context-based methods through comparative learning. FU et al. constructed the algorithm named contrastive learning augmented context-based meta-RL (CCM) based on MoCo[40] and CURL[41]. WANG et al. proposed a method similar to CCM, TCL, where positive and negative samples are divided according to sampling trajectories rather than task types[42].

In this paper, we utilize the context-based VariBAD algorithm[39] to consider radar detection task characteristics and requirements.

2.2 Radar Control with Reinforcement Learning

Reinforcement learning methods enable automatic learning of complex behaviors, and several studies have focused on introducing deep reinforcement learning into radar control. AZIZ et al. provided a survey of literature proposing the application of reinforcement learning to radar to overcome jamming[2]. WANG et al. suggested a cognitive frequency design method for a compressed-sensing-based frequency agile radar using reinforcement learning[43]. PATTANAYAK et al. introduced an inverse reinforcement learning approach to meta-cognitive radars in an adversarial setting[44]. ZHAI et al. proposed a reinforcement learning-based approach for multi-input multi-output (MIMO) cognitive radar[45]. OTT et al. proposed an uncertainty-based meta-reinforcement learning approach with out-of-distribution environment detection[46]. In the context of multi-agent systems, SNOW et al. proposed a multi-objective inverse reinforcement learning approach for tracking targets with a cognitive radar network[47]. MENG et al. examined the issue of target assignment when a phased-array radar network detects hypersonic-glide vehicles in near space and proposed a method for target assignment based on deep reinforcement learning[48].

The aforementioned studies illustrate that deep reinforcement learning has extensive potential applications in various aspects of radar systems. However, these related works are conducted in simple simulated scenarios, and thus it remains challenging to implement reinforcement learning methods in real-world situations. In this paper, we concentrate on the framework for extensive single radar parameter control, and we introduce realistic sample-limited settings and corresponding reinforcement learning methods to tackle this problem.

3 Reinforcement Learning Framework for Automatic Radar Detection

3.1 Environment Modeling

Environment modeling is the foundation of reinforcement learning. Existing modules of traditional radar control are: a) analog signal → plot processing; b) plot → track processing; c) track → radar parameter control module. The intelligent radar control system mainly requires intelligent automatic control of the radar while observing the processed radar data (e.g., plots, tracks, etc.), and its framework is shown in Fig. 2. In order to enhance the universality and generalization performance of our reinforcement learning algorithm, our agent focuses on processing the input data composed of original analog signals, processed plots, and mixed tracks as states, and outputs controllable radar parameters.

The radar point and track processing algorithms typically operate in cycles. After each radar scan is completed and before the next one begins, our agent makes its decisions. In this context, the input state $s = \left(s^1, s^2, s^3\right)$ includes:

a) A 3-dimensional raw echo analog signal, denoted as $s^1 \in [H, W, V]$. Here, H, W, and V represent the distance, deviation angle, and amplitude of the signal, respectively. This analog signal is the radar's echo signal in each direction. The data for each cycle is a position peak matrix.

b) Dots denoted as $s^2 = \left\{\left(x_1, y_1, v_1\right), \left(x_2, y_2, v_2\right), \cdots, \left(x_n, y_n, v_n\right)\right\}$. These are a series of points identified as target points in the analog signal. Each point has features, such as position and signal-to-noise ratio, extracted by algorithms. The number of points in the plot data for each cycle is uncertain. Each point has one row of features. Although there are much more clutter points in dots compared with tracks, it may cover more potential targets.

c) Tracks denoted as $s^3 = \left\{\left(x'_1, y'_1, v'_1, l'_1\right), \left(x'_2, y'_2, v'_2, l'_2\right), \cdots, \left(x'_{n'}, y'_{n'}, v'_{n'}, l'_{n'}\right)\right\}$. A track is a series of points in a historical track where the target point is recognized as a real target. Each point has features, such as position and velocity, extracted by algorithms. Similar to the format of dots, there are multiple dots in each cycle of dot data, each with a single line of features. However, each target is additionally marked with a unique batch number. The trajectory is the main basis for decision-making, but it often lacks some difficult-to-detect target information and performs with a certain lag.

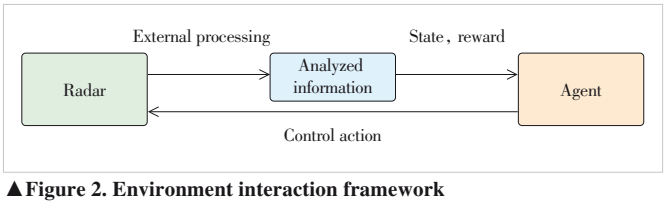
The system's output is the radar's parameter control information, which includes frequency point, speed, pitch angle, and timing transmission. Note that these are generally discrete variables.

3.2 Learning Framework

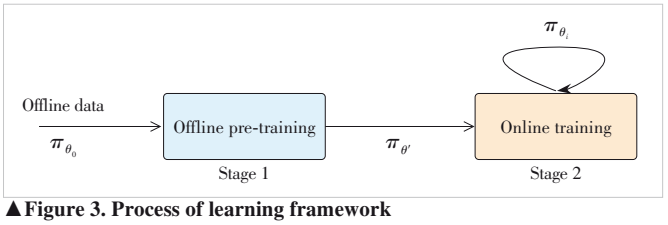
As Fig. 3 illustrates, the framework's primary process is divided into two stages:

The first stage involves data collection and offline pre-training. Although offline reinforcement learning algorithms do not impose strict requirements on offline training data, existing research indicates that the diversity of offline training data significantly impacts the learning outcome[49]. Hence, we aim to conduct offline reinforcement learning pre-training for the decision model, denoted as π_θ (where θ represents model parameters), based on as many diverse and abundant offline training data as possible. This stage initiates with model parameters θ_0 and results in pre-training parameters θ'.

The second stage involves running the pre-trained decision



▲Figure 2. Environment interaction framework



▲Figure 3. Process of learning framework

model $\pi_{\theta'}$ in an actual radar scenario and performing reinforcement learning iterations online. To facilitate a smooth transition from offline data to online scenarios for the decision model, we incorporate a meta-reinforcement learning algorithm to enhance the model's generalization. Given that radar detection requires precise environmental cognition, the introduced inference-based meta-reinforcement learning algorithm includes a task feature inference module and an auxiliary training target. This new meta-reinforcement learning network structure is also utilized in the offline pre-training phase, which indicates the combination of offline reinforcement learning and meta-reinforcement learning.

3.3 Reinforcement Learning Method

Network design: Our design integrates a fundamental RL algorithm and a variational autoencoder (VAE)[50] to encode different task scenarios automatically. These encoded features are subsequently inputted into the intelligent agent[39]. The VAE model consists of an encoder $q_\phi(s_t,a_{t-1},r_t,h_{t-1}) \rightarrow z_t$ and two decoders $p_\psi^R(z_t,s_i,a_i,s_{i+1}) \rightarrow r_{i+1}, p_\psi^T(z_t,s_i,a_i) \rightarrow s_{i+1}$. This model leverages the reconstruction constraints of rewards and states, along with constrained dimension to compress the original inputs $o_t = \{s_t,a_{t-1},r_t\}$ into low-dimensional representation z_t efficiently. The posterior distribution of a task can be interpreted as a representation of specific task characteristics, such as meteorological features, ship-type tendencies and radar models. The decoder takes a posterior distribution of tasks and some prior knowledge as input to predict the subsequent state. The context encoder q_ϕ is required to encode an indefinite length historical sequence. Therefore, a recurrent neural network (RNN) model is employed for approximation, and other models can be approximated using a multi-layer perceptron (MLP). The VAE model outputs a low-dimensional representation of tasks z_t to policy model $\pi_\theta(s_t,z_t) \rightarrow a_t$. The complete network structure is shown in Fig. 4.

Offline reinforcement learning: Although a variety of offline reinforcement learning algorithms are available, we have opted for the behavioral cloning (BC) objective due to its ease of use and scalability. In practice, our policy model takes all historical trajectories as input. Let's denote the previously collected dataset as $D = \{(o_i,a_i)\}$, where the equivalent new state is the historical trajectory $o_i = \{s_i,a_{i-1},r_i\}$, $s_i = (s_i^1,s_i^2,s_i^3)$. Thus, the offline training objective is:

$$J_1 = -E_{(s,a)}\mathrm{Dist}\left(\pi_\theta(o_i),a_i\right), \tag{2}$$

where $E_{(s,a)}$ is the expectation, $\mathrm{Dist}(\cdot)$ is the distance calculation function, which can be set as a 0 – 1 function for discrete variables. Behavioral cloning targets enable the policy model's decision actions, $\pi_\theta(o_i)$, to be closer to expert actions, thereby allowing the parameters θ to learn a certain level of strategic knowledge.

Meta-reinforcement learning: We employ reconstruction and the information bottleneck objective to constrain feature information. The forms of reward and state reconstruction objective functions are:

$$J_2 = -E_{r_i}\mathrm{Dist}\left(p_\psi^R\left(q_\phi(s_i,a_{i-1},r_i,h_{i-1}),s_i,a_i,s_{i+1}\right),r_i\right),$$

$$J_3 = -E_{s_i}\mathrm{Dist}\left(p_\psi^T\left(q_\phi(s_i,a_{i-1},r_i,h_{i-1}),s_i,a_i\right),s_i\right). \tag{3}$$

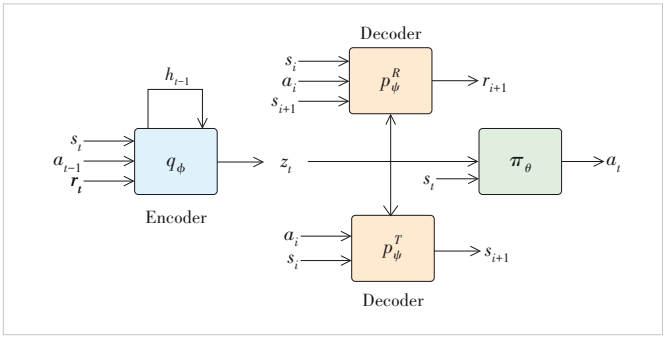
$\mathrm{Dist}(\cdot)$ can be set as the L2 distance function for continuous variables such as the state and reward. The desired feature of the target, $z_i = q_\phi(s_i,a_{i-1},r_i,h_{i-1})$, retains the original input information.

Moreover, the reconstruction objective involves the simultaneous optimization of two models, which may result in gradient descent optimization not achieving the expected results. To expedite training, we additionally introduce the information bottleneck method optimization objective[36], which is in the form of:

$$J_4 = -E_i D_{\mathrm{KL}}\left(q_\phi(s_i,a_{i-1},r_i,h_{i-1}),r(z)\right), \tag{4}$$

where $r(z)$ is the normal distribution.

During the offline pre-training stage, the overall optimization objective is $J_1 + J_2 + J_3 + J_4$ as the VAE is also trained. In the online training phase, the offline reinforcement learning objective is replaced by the traditional reinforcement learning objective J_{RL}, and the overall optimization objective is $J_{\mathrm{RL}} + J_2 + J_3 + J_4$. Note that due to the existing initialization parameters, we need to reduce the learning rate by an order of magnitude during online tuning.



▲Figure 4. Network structure of the agent

4 Experiment

Experimental tasks: Despite the universality of our constructed method, we require explicit task scenarios and objectives for the experiment. Our primary experimental scenario involves enhancing the average signal-to-noise ratio (SNR) of radar detection targets by manipulating the radar's frequency points. Given the radar's relatively short rotation time (either

1 s or 10 s), we assume that the environment will remain largely unchanged even if the radar scans every alternate turn. As for the reward setting issue, we alternate between a circle set as a frequency point generated by the algorithm and the next circle as a fixed frequency point. This approach provides a relatively standardized reward for the reinforcement learning algorithm, $r_t(a) = \mathrm{SNR}(s_t,a) - \mathrm{SNR}(s_t,a_{\mathrm{fix}})$.

Evaluation setting: For offline data, actions are expert strategies, and we compare the overlap between algorithm output actions and offline data actions. For online learning, considering the practical application, we involve relevant radar operation experts to compare the algorithm's parameter control rewards with the expert's parameter control rewards. We ask both the algorithm and experts to test each other on the same task, compare the cumulative rewards of the algorithm with the cumulative rewards of the experts, and take the average of three experiments. To benchmark real-world sample-limited applications, in the online training stage, the sample number is limited to 10 rounds, i.e. 10 000 steps.

Implementation: We employ proximal policy optimization (PPO) as the reinforcement learning algorithm during the online training phase. We use MATLAB as it supports the Radar Toolbox to construct a simulation environment. We achieve code communication between Python and MATLAB via the user datagram protocol (UDP). The environment and algorithms ran on a 2.5 GHz CPU and a single NVIDIA GeForce RTX 3080 graphics card. For offline training data, we have experts control the selection of radar parameters in the task, but we also aim to cover as many action intervals as possible, thereby obtaining data with a total of 100 000 steps with a wide distribution of action.

Experimental results: After achieving convergence in the offline training phase, the action similarity between the decision model and offline data is 99%. In online tests, the average cumulative reward of the proposed method reaches 91% of the experts' method, and the performance of a random policy is unstable and obviously weaker. Detailed online testing results with average cumulative reward are shown in Table 1. Additionally, we observe that the decision model can control different parameters for different targets, and it tends to favor some commonly used radar parameters. The experimental results indicate that the decision model has learned the preliminary radar control policy, but the potential of deep learning may not be fully exploited due to the limitation of the training sample size. According to our experience and expert judgment, our method can act as humans in basic radar automatic detection, and therefore has the potential to be applied in practical radar operation tasks. In future research, we will focus on further enhancing the effectiveness of reinforcement learning and aim to apply it to actual radar.

▼Table 1. Online testing results

Test	Random Policy	Proposed	Experts
Trial 1	−9.24	24.32	26.14
Trial 2	12.78	28.77	29.33
Trial 3	6.34	25.38	30.85
Average	3.29	26.16	28.77

5 Conclusions

In this paper, we have presented a novel practical approach to radar operation that leverages the power of reinforcement learning. By integrating offline reinforcement learning and meta-reinforcement learning methods, we have developed a practical radar operation reinforcement learning framework that can quickly adapt to unseen real-world tasks. Our experimental results have demonstrated the ability to act as humans in basic radar automatic detection with real-world settings, thereby validating our approach. Our work not only addresses the current challenges in radar operation but also paves the way for the practical application of reinforcement learning in radar operation. The proposed method has the potential to revolutionize radar detection by enhancing its efficiency, precision, and automation. Future work will focus on further refining our framework and exploring its application in real-world radar systems.

References

[1] GENG Z, YAN H, ZHANG J, et al. Deep-learning for radar: a survey [J]. IEEE access, 2021, 9: 141800-141818. DOI:10.1109/ACCESS.2021.3119561
[2] AZIZ M M, MAUD A R M, HABIB A. Reinforcement learning based techniques for radar anti-jamming [C]//International Bhurban Conference on Applied Sciences and Technologies (IBCAST). IEEE, 2021: 1021 – 1025. DOI: 10.1109/IBCAST51254.2021.9393209
[3] SUTTON R S, BARTO A G. Reinforcement learning: an introduction (2nd ed) [M]. Cambridge, USA: MIT press, 2018
[4] SILVER D, HUANG A, MADDISON C J, et al. Mastering the game of Go with deep neural networks and tree search [J]. Nature, 2016, 529(7587): 484 – 489. DOI: 10.1038/nature16961
[5] VINYALS O, BABUSCHKIN I, CZARNECKI W M, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning [J]. Nature, 2019, 575(7782): 350 – 354. DOI: 10.1038/s41586-019-1724-z
[6] LI J J, KOYAMADA S, YE Q W, et al. Suphx: mastering mahjong with deep reinforcement learning [EB/OL]. [2023-04-16]. https://arxiv.org/abs/2003.13590
[7] DEGRAVE J, FELICI F, BUCHLI J, et al. Magnetic control of tokamak plasmas through deep reinforcement learning [J]. Nature, 2022, 602(7897): 414 – 419. DOI: 10.1038/s41586-021-04301-9
[8] SCHRITTWIESER J, ANTONOGLOU I, HUBERT T, et al. Mastering Atari, Go, chess and shogi by planning with a learned model [J]. Nature, 2020, 588 (7839): 604 – 609. DOI: 10.1038/s41586-020-03051-4
[9] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518(7540): 529 – 533. DOI: 10.1038/nature14236
[10] WANG Z Y, SCHAUL T, HESSEL M, et al. Dueling network architectures for deep reinforcement learning [C]//The 33rd International Conference on International Conference on Machine Learning. ACM, 2016: 1995 – 2003. DOI: 10.5555/3045390.3045601
[11] HAUSKNECHT M, STONE P. Deep recurrent Q-learning for partially observable MDPs [J]. AAAI fall symposium, 2015: 29 – 37
[12] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning [EB/OL]. [2023-04-16]. https://arxiv. org/pdf/1509.02971. pdf. DOI: 10.1016/S1098-3015(10)67722-4
[13] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal policy optimization algorithms [EB/OL]. (2017-08-28) [2023-04-16]. https://arxiv.org/abs/1707.06347

[14] HAARNOJA T, ZHOU A, ABBEEL P, et al. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor [EB/OL]. (2018-08-08) [2023-04-16]. https://arxiv.org/abs/1801.01290
[15] FUJIMOTO S, VAN HOOF H, MEGER D. Addressing function approximation error in actor-critic methods [C]//International Conference on Machine Learning. PMLR, 2018: 1587 – 1596
[16] FUJIMOTO S, MEGER D, PRECUP D. Off-policy deep reinforcement learning without exploration [EB/OL]. (2019-08-10) [2023-04-16]. https://arxiv. org/abs/1812.02900
[17] KUMAR A, FU J, TUCKER G, et al. Stabilizing off-policy Q-learning via bootstrapping error reduction [EB/OL]. (2019-11-25) [2023-04-16]. https://arxiv. org/abs/1906.00949
[18] NACHUM O, DAI B, KOSTRIKOV I, et al. AlgaeDICE: policy gradient from arbitrary experience [EB/OL]. (2019-12-04) [2023-04-16]. https://arxiv. org/abs/1912.02074
[19] LIU Y, SWAMINATHAN A, AGARWAL A, et al. Off-policy policy gradient with state distribution correction [EB/OL]. (2019-07-16) [2023-04-16]. https://arxiv.org/abs/1904.08473
[20] KUMAR A, ZHOU A, TUCKER G, et al. Conservative Q-learning for offline reinforcement learning [EB/OL]. (2020-08-19) [2023-04-16]. https://arxiv. org/abs/2006.04779
[21] MATSUSHIMA T, FURUTA H, MATSUO Y, et al. Deployment-efficient reinforcement learning via model-based offline optimization [EB/OL]. (2020-06-05) [2023-04-16]. https://arxiv.org/abs/2006.03647
[22] YU T H, THOMAS G, YU L, et al. MOPO: model-based offline policy optimization [J]. Advances in neural information processing systems, 2020, 33: 14129-14142.
[23] KIDAMBI R, RAJESWARAN A, NETRAPALLI P, et al. MOReL: Model-based offline reinforcement learning [J]. Advances in neural information processing systems, 2020, 33: 21810 – 21823
[24] YU T H, KUMAR A, RAFAILOV R, et al. COMBO: conservative offline model-based policy optimization [EB/OL]. (2022-01-27) [2023-04-16]. https://arxiv. org/abs/2102.08363
[25] JANNER M, FU J, ZHANG M, et al. When to trust your model: Model-based policy optimization [C]//The 33rd International Conference on Neural Information Processing Systems. ACM, 2019: 12519 – 12530
[26] FINN C, ABBEEL P, LEVINE S. Model-agnostic meta-learning for fast adaptation of deep networks [C]//The 34th International Conference on Machine Learning. ACM, 2017: 1126 – 1135. DOI: 10.5555/3305381.3305498
[27] NICHOL A, ACHIAM J, SCHULMAN J. On first-order meta-learning algorithms [EB/OL]. (2018-10-22) [2023-04-16]. https://arxiv.org/abs/1803.02999
[28] SONG X Y, GAO W B, YANG Y X, et al. ES-MAML: simple hessian-free meta learning [EB/OL]. (2020-07-07) [2023-04-16]. https://arxiv.org/abs/1910.01215
[29] ANTONIO A, STORKEY A, EDWARDS H. How to train your MAML [EB/OL]. (2019-03-15) [2023-04-16]. https://arxiv.org/abs/1810.09502
[30] DUAN Y, SCHULMAN J, CHEN X, et al. RL^2: fast reinforcement learning via slow reinforcement learning [EB/OL]. (2016-11-10) [2023-04-16]. https://arxiv.org/abs/1611.02779
[31] MISHRA N, ROHANINEJAD M, CHEN X, et al. A simple neural attentive meta-learner [EB/OL]. (2018-02-15) [2023-04-16]. http://arxiv.org/abs/1707.03141.
[32] PARISOTTO E. Meta Reinforcement Learning through Memory [D]//Pittsburgh: Carnegie Mellon University, 2021
[33] SÆMUNDSSON S, HOFMANN K, DEISENROTH M P. Meta reinforcement learning with latent variable Gaussian processes [EB/OL]. (2018-05-20) [2023-06-16]. https://arxiv.org/abs/1803.07551
[34] ZINTGRAF L, SHIARLIS K, KURIN V, et al. Fast context adaptation via meta-learning [C]//The 36th International Conference on Machine Learning. ICML, 2019: 13262 – 13276
[35] LAN L, LI Z, GUAN X, et al. Meta reinforcement learning with task embedding and shared policy [C]//The 28th International Joint Conference on Artificial Intelligence. ACM, 2019: 2794 – 2800. DOI:10.24963/ijcai.2019/387
[36] HUMPLIK J, GALASHOV A, HASENCLEVER L, et al. Meta reinforcement learning as task inference [EB/OL]. (2019-05-15) [2023-06-16]. https://arxiv.org/abs/1905.06424
[37] FAKOOR R, CHAUDHARI P, SOATTO S, et al. Meta-Q-Learning [EB/OL]. (2019-09-30) [2023-06-16]. http://arxiv.org/abs/1910.00125
[38] RAILEANU R, GOLDSTEIN M, SZLAM A D, et al. Fast adaptation to new environments via policy-dynamics value functions [C]//International Conference on Machine Learning. ICML, 2020: 7920 – 7931
[39] ZINTGRAF L, SCHULZE S, LGL M, et al. VariBAD: variational Bayes-adaptive deep RL via meta-learning [J]. The journal of machine learning research, 2021, 22 (1): 13198 – 13236
[40] HE K M, FAN H Q, WU Y X, et al. Momentum contrast for unsupervised visual representation learning [C]//Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020: 9726 – 9735. DOI: 10.1109/CVPR42600.2020.00975
[41] LASKIN M, SRINIVAS A, ABBEEL P C. Contrastive unsupervised representations for reinforcement learning [C]//The 37th International Conference on Machine Learning. ICML, 2020: 5595 – 5606
[42] WANG B, XU S, KEUTZER K, et al. Improving context-based meta-reinforcement learning with self-supervised trajectory contrastive learning [EB/OL]. (2021-05-10) [2023-04-16]. https://arxiv.org/abs/2103.06386
[43] WANG S S, LIU Z, XIE R, et al. Reinforcement learning for compressed-sensing based frequency agile radar in the presence of active interference [J]. Remote sensing, 2022, 14(4): 968. DOI: 10.3390/rs14040968
[44] PATTANAYAK K, KRISHNAMURTHY V, BERRY C. Meta-cognition. an inverse-inverse reinforcement learning approach for cognitive radars [C]//The 25th International Conference on Information Fusion (FUSION). IEEE, 2022: 1 – 8
[45] ZHAI W T, WANG X R, GRECO M S, et al. Weak target detection in massive MIMO radar via an improved reinforcement learning approach [C]//IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2022: 4993 – 4997. DOI: 10.1109/ICASSP43922.2022.9746472
[46] OTT J, SERVADEI L, MAURO G, et al. Uncertainty-based meta-reinforcement learning for robust radar tracking [C]//The 21st IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2023: 1476 – 1483. DOI: 10.1109/ICMLA55696.2022.00232
[47] SNOW L, KRISHNAMURTHY V, SADLER B M. Identifying coordination in a cognitive radar network-a multi-objective inverse reinforcement learning approach [C]//IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2023: 1 – 5. DOI: 10.1109/ICASSP49357.2023.10096376
[48] MENG F Q, TIAN K S, WU C F. Deep reinforcement learning-based radar network target assignment [J]. IEEE sensors journal, 2021, 21(14): 16315 – 16327. DOI: 10.1109/JSEN.2021.3074826
[49] FU J, KUMAR A, NACHUM O, et al. D4RL: Datasets for deep data-driven reinforcement learning [EB/OL]. (2020-04-15) [2023-04-16]. https://arxiv. org/abs/2004.07219
[50] KINGMA D P, WELLING M. Auto-encoding variational Bayes [EB/OL]. (2013-12-10) [2023-04-16]. https://arxiv.org/abs/1312.6114

Biographies

YU Junpeng received his master's degree in communication and information systems. He is a senior engineer with the Nanjing Research Institute of Electronics Technology, the deputy secretary-general of Intelligent Perception Special Committee of Jiangsu Association of Artificial Intelligence. His research interests include radar systems and intelligent processing technologies based on artificial intelligence. He has participated in many key artificial intelligence projects sponsored by the Ministry of Science and Technology of the People's Republic of China.

CHEN Yiyu (yiyuiii@foxmail.com) is currently a PhD student in the Department of Computer Science and Technology, Nanjing University, China. His research interest includes meta-reinforcement learning and robot control.