Proceedings of CONF-SEML 2026 Symposium: Computational Analysis and Modeling in Complex Intelligent Systems
DOI: 10.54254/2755-2721/2026.GU33493

A Survey of Deep Reinforcement Learning Algorithms: From Value Functions to Transformers

Jinning Jiang

College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China
302023334040@zjut.edu.cn

Abstract. Deep Reinforcement Learning is the product of combination of Deep Learning and Reinforcement Learning, it is aim to solve the sequential decision problem in high-dimensional state space. The article systematically sorts out the development vein of Deep Reinforcement Learning in the past ten years with "From Value Functions to Transformers" as the main line. First of all, the preliminary knowledge is introduced, including the role of Markov's decision-making process, Q-learning and Bellman equations, policy gradient theorem and deep learning. Then, according to the technological evolution route, the DRL algorithm is divided into three generations for detailed explanation: the first generation is based on value function-based methods (DQN and its improvement), the second generation is based on strategy and Actor-Critic (PPO, DDPG, SAC, etc.), and the third generation is based on the method of Transformer (Decision Transformer, GATO, etc.). On this basis, the three-generation methods are compared and analyzed from the core mechanism, action space, sample efficiency, realization complexity, scalability and other dimensions. Finally, it discusses the core challenges faced by DRL, such as sample efficiency, Sim2Real migration, and security assurance, and looks forward to the future.

Keywords: Deep reinforcement learning, Value function, Policy gradient, Transformer

1. Introduction

In 2015, Deep Q-Network (DQN) proposed by Mnih et al. [1], combined deep convolutional network with Q-learning for the first time, directly from the original pixel learning strategy, and reached the level of surpassing human professional testers in the 49 tasks of the Atari 2600 game, marking the official opening of the era of deep reinforcement learning.

Since then, DRL has undergone three generations of evolution from value function learning to strategy optimization and sequence modeling. The first-generation method based on value function-based methods is represented by DQN. It approaches the Q value through deep neural networks and uses experience replay and target networks to achieve stable training. On this basis, Double DQN, Dueling DQN, priority experience replay and other improvements further improve performance and practicality. The second generation directly optimizes the strategy network based on strategy and Actor-Critic methods. It breaks through the limitations of the value function-based method in the continuous action space. Among them, PPO achieves a stable and efficient balance with cropping replacement goals; SAC introduces the maximum entropy framework to explore and utilize the

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.GU33493

balance through automatic temperature regulation; DDPG combines deterministic strategy with Actor-Critic for the first time, laying the foundation for continuous control tasks. The third-generation method based on Transformer represents the latest paradigm shift - redefining decision-making problems as sequence prediction, using the self-attention mechanism to capture long-distance dependence, and opening the era of "big decision-making models".

Deep reinforcement learning has become an important research direction in the field of artificial intelligence, achieving remarkable success in games, robot control, autonomous driving, and other tasks. This article takes "from value function to Transformer" as the main line and systematically reviews the development of deep reinforcement learning algorithms. Section 2 introduces the preliminary knowledge (MDP, Q-learning, policy gradient); Sections 3 to 5 introduce the three generations of methods respectively: value function-based methods, policy-based and Actor-Critic methods, and Transformer-based methods; Section 6 provides a comparative analysis of the three generations; Section 7 discusses the core challenges and future development directions of DRL; Section 8 summarizes the full text.

2. Preliminary knowledge

Reinforcement Learning (RL) is an important branch of machine learning. Its core goal is to allow intelligent bodies to learn optimal decision-making strategies through interaction with the environment [2].

2.1. Markov's decision-making process

Reinforcement learning problems are usually formalized as Markov Decision Process (MDP). MDP can be represented by the five-member group $(SAPR\gamma)$: S is the state space, A is the action space, $P(s^{'} \mid s,a)$ is the state transfer probability, $R(s,a)$ is the instant reward, and $\gamma \in [0,1]$ is the discount factor. The agent observes the current state $s \in S$ at each time step, selects the action $a \in A$, and the environment enters the next state according to the transfer probability and returns the reward. The goal of the agent is to maximize the cumulative discount reward

$$\mathbb{E}\left[\sum_{t=0}^{T} \gamma^{t} r_{t}\right]. \tag{1}$$

2.2. Q-learning and bellman's equation

Q-learning is the most representative reinforcement learning algorithm based on value functions. It evaluates the expected cumulative reward for performing action a under state s by learning the action value function $Q(s,a)$. The optimal action value function satisfies Bellman's optimal equation:

$$Q^{*}\left(s,a\right) = \mathbb{E}\left[r + \gamma \max_{a^{'}} Q^{*}\left(s^{'},a^{'}\right) \mid s,a\right]. \tag{2}$$

Q-learning approaches the optimal Q value through iterative updates, and its update rules are:

$$Q\left(s,a\right) \leftarrow Q\left(s,a\right) + \alpha\left[r + \gamma \max_{a^{'}} Q\left(s^{'},a^{'}\right) - Q\left(s,a\right)\right], \tag{3}$$

DOI: 10.54254/2755-2721/2026.GU33493

where α is the learning rate [2].

2.3. Policy gradient theorem

Unlike the method based on value functions, the policy gradient method directly parameterizes the strategy $\pi_\theta(a \mid s)$ and optimizes the strategy parameters through gradient ascending. The policy gradient theorem gives the gradient calculation formula:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(as) \cdot Q^{\pi_\theta}(s,a)], \tag{4}$$

this theory lays the foundation for modern strategy optimization algorithms such as TRPO and PPO [2].

2.4. The role of deep learning

The traditional reinforcement learning method performs well in a small state space, but it faces a dimensional disaster problem in the face of high-dimensional input such as images - the state space grows with the dimensional index. Deep learning automatically extracts features through multi-layer neural networks and maps high-dimensional inputs (such as image pixels) to low-dimensional characterization, which effectively solves this problem [3]. Deep Q Network (DQN) combines convolutional neural network with Q-learning to realize end-to-end learning from primitive pixels to action for the first time [1].

The above concepts constitute the theoretical basis of deep reinforcement learning, and the various algorithms introduced in the following chapters are based on this framework.

3. First-generation value-function-based methods

3.1. DQN: milestone breakthrough

Before DQN, reinforcement learning mainly relied on linear function approximation or manual design characteristics, and it was difficult to handle high-dimensional input. In 2015, Mnih. The Deep Q-Network (DQN) proposed by others [work combines the deep convolutional network with Q-learning for the first time, realizing end-to-end learning from the original pixel to the action, so that the agent can learn the optimal strategy directly from 84×84 grayscale images, In the 49 missions of the Atari 2600 game, it has reached a level that surpasses human players [1].

The core innovation of DQN is to solve the problem of "combining unstable neural networks with reinforcement learning", which mainly relies on two key technologies [1]:

Experience Replay: The interactive data in reinforcement learning is time-related, and direct use for training will lead to neural network over-Fitting. DQN draws on the playback mechanism of the hippoma in the animal brain to store each interaction $e_t = (s_t, a_t, r_t, s_{t+1})$ of the agent in a playback buffer with a capacity of 10,000 pages. During training, uniform random sampling from the buffer zone forms a small batch of 32 samples. This practice breaks the time correlation of the data and makes the gradient drop more stable. At the same time, each experience can be used many times, which improves the efficiency of the sample [2].

Target Network: In standard Q-learning, the TD target comes from the same network as the currently updated Q, which leads to training instability. DQN introduce an independent target network $\widehat{Q}$. Its parameters are only synchronized with the online network once every $C = 10000$

DOI: 10.54254/2755-2721/2026.GU33493

steps, so that the TD target remains stable for a period of time, which greatly improves the convergence of training [1].

Experiments show that DQN has surpassed all existing methods in 43 games and reached the level of more than 75% of human professional testers in 29 games [1].

3.2. Double DQN: alleviate the estimation problem

Although DQN has achieved breakthrough results in the Atari game, van Hasselt et al. [4], found that there was a serious problem of overestimating the action value. In games such as Asterix and Wizard of Wor, the action value estimated by DQN is much higher than the actual cumulative reward, resulting in training instability and performance decline [4].

The root cause of the overestimation problem is the max operation in the standard update formula of Q-learning - it uses the same network parameters to select and evaluate actions at the same time. When there is an error in the estimation, the max operation will tend to choose the overestimated action, which will cause the overestimation problem to accumulate [4].

The core improvement of Double DQN is to decouple action selection from action evaluation: use the online network to select the optimal action and use the target network to evaluate the value of the action. The target function becomes:

$$Y_t^{\text{DoubleDQN}} = R_{t+1} + \gamma Q\left(S_{t+1}, \underset{a}{\text{argmax}} Q\left(S_{t+1}, a; \theta_t\right); \theta_t^-\right), \tag{5}$$

This design effectively alleviates the problem of overestimation with almost no increase in computing overhead [4]. Experiments show that Double DQN's average score and stability on 49 Atari games are better than DQN, from 233% to 617% in Road Runner games, and from 70% to 180% in Asterix games [4].

3.3. Dueling DQN: improve the accuracy of q value estimation

The network structure of the standard DQN directly outputs the Q value of each action, but in many states, the difference in the Q value of different actions is very small, which makes it difficult for the network to distinguish effectively. The Dueling DQN proposed by Wang et al. solves this problem by separating the state value function and the advantage function.

As shown in Table 1, Dueling DQN decomposes the Q network into two separate parts:

Table 1. Dueling DQN

Value function $V(s)$	only related to the state, evaluate the quality of the current state
Advantage function $A(s,a)$	It is related to the state and action at the same time, and evaluates the relative advantage of each action.

The final action value is combined in the following ways:

$$Q\left(s,a\right) = V\left(s\right) + \left(A\left(s,a\right) - \frac{1}{|\mathcal{A}|}\sum_{a'} A\left(s,a'\right)\right). \tag{6}$$

The core advantage of this design is that even in the state where there is no need to distinguish the value of the action, the network can effectively learn the state value without assigning the Q value to each action separately.

DOI: 10.54254/2755-2721/2026.GU33493

3.4. Priority experience replay: improve sample efficiency

The experience replay of standard DQN adopts a uniform sampling strategy to give the same sampling probability to all experiences. However, the importance of different experiences to learning varies greatly - experiences with large TD errors often contain more learning information.

Prioritized Experience Replay (PER) proposed by Schaul and others links the sampling probability with TD error:

$$P\left(i\right)=\frac{p_i^\alpha}{\sum_j p_j^\alpha},p_i=\mid\delta_i\mid+\epsilon. \tag{7}$$

Among them, δ_i is the TD error of the i th sample, and α controls the priority. Priority sampling will change the data distribution and introduce estimated deviations. To solve this problem, PER introduces the importance sampling weight for deviation correction. Experiments show that PER significantly improves the sample efficiency and convergence speed of DQN.

4. Second-generation policy-based and actor-critic methods

4.1. Policy gradient method

Unlike the method based on value functions, the policy gradient method directly parameterizes the strategy $\pi_\theta(a\mid s)$ and optimizes the strategy parameters through gradient ascending. The policy gradient theorem gives the gradient calculation formula [2]:

$$\nabla_\theta J\left(\theta\right)=\mathbb{E}_{\pi_\theta}\left[\nabla_\theta\log\pi_\theta\left(as\right)\cdot Q^{\pi_\theta}\left(s,a\right)\right]. \tag{8}$$

The REINFORCE algorithm is the most basic policy gradient method, using Monte Carlo sampling to estimate the cumulative return G_t . Its core advantage is unbiased estimation - direct use of real sampling returns. However, the main disadvantage is that the variance is too large: due to the randomness of the path from strategy sampling to the final return, the return of different episodes may vary greatly, resulting in unstable gradient estimation and slow convergence.

Introduce Baseline to reduce variance: In order to reduce the variance, a baseline function $b(s)$ that has nothing to do with the action can be subtracted from it. The common choice is the state value function $V(s)$.

At this time, the gradient becomes:

$$\nabla_\theta J(\theta)=\mathbb{E}_{\pi_\theta}\left[\nabla_\theta\log\pi_\theta(a\mid s)\cdot(Q(s,a)-V(s))\right]. \tag{9}$$

Among them, $A(s,a)=Q(s,a)-V(s)$ is called the advantage function. Subtracting the baseline will not change the expected value of the gradient, but it can significantly reduce the variance and accelerate the convergence. This idea lays the foundation for subsequent Actor-Critic methods (such as TRPO, PPO) [5].

4.2. PPO: balance between stability and efficiency

Trusted area strategy optimization (TRPO) limits the range of strategy updates through KL dispersion constraints and ensures the stability of training, but its implementation is complex - it needs to calculate the Fisher information matrix or conjugation gradient, and the code

DOI: 10.54254/2755-2721/2026.GU33493

implementation threshold is high [5]. In response to this problem, Schulman et al. [6], proposed Proximal Policy Optimization (PPO), which greatly simplified the implementation while maintaining the stability of TRPO.

Core Innovation: Clipped Surrogate Objective

The core idea of PPO is to limit the extent of policy updates and prevent policy collapse due to excessive single-step updates. Its target function is:

$$L^{CLIP}\left(\theta\right)=\mathbb{E}_t\left[\min\left(r_t\left(\theta\right)\widehat{A}_t,\text{clip}\left(r_t\left(\theta\right),1-\epsilon,1+\epsilon\right)\widehat{A}_t\right)\right], \tag{10}$$

where $r_t\left(\theta\right)=\frac{\pi_\theta(a_t s_t)}{\pi_{\theta_{old}}(a_t s_t)}$ is the probability ratio of the old and new strategies, and $\widehat{A}_t$ is the advantage function estimate. The cropping operation limits the probability ratio to the $[1-\epsilon,1+\epsilon]$ range:

When the action advantage is positive, r_t is limited to less than $1+\epsilon$ to prevent excessive increase in the probability of good action.

When the action advantage is negative, r_t is limited to more than $1-\epsilon$ to prevent excessive reduction in the probability of bad action. This design allows PPO to greatly simplify the implementation complexity while maintaining the stability of TRPO, and it only takes a few lines of code to achieve it [6]. Because of this, PPO has become one of the most widely used DRL algorithms at present.

4.3. DDPG: breakthrough of continuous control

After DQN is successfully applied to discrete action space, how to extend deep reinforcement learning to the field of continuous control has become a key issue. The Deep Deterministic Policy Gradient (DDPG) proposed by Lillicrap et al. [7], solves this challenge.

The core innovation of DDPG lies in the combination of deterministic strategy and Actor-Critic framework:

Deterministic strategy $\mu(s;\theta^\mu)$: directly output the determined action value, not the action probability distribution, which greatly reduces the sampling complexity of the continuous action space.

Actor-Critic Architecture: Actor Network Learning Strategy, Critic Network Evaluation Q Value, Two Collaborative Optimization.

In addition, DDPG draws on two key technologies of DQN: experience replay and target network. These designs make DDPG the first deep RL algorithm that can achieve stable effects in continuous control tasks (such as MuJoCo robots) [7], laying the foundation for subsequent SAC, TD3 and other algorithms.

4.4. SAC: maximum entropy framework

Although DDPG has been successful in continuous control tasks, it is sensitive to hyperparameters and has limited exploration ability. The Soft Actor-Critic (SAC) algorithm proposed by Haarnoja et al. [8], introduces the maximum entropy framework and fundamentally solves this problem.

The core innovation of SAC is to introduce entropy regularization terms into the target function:

$$J\left(\pi\right)=\sum_{t=0}^{T}\mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[r\left(s_t,a_t\right)+\alpha\mathscr{H}\left(\pi\left(\cdot s_t\right)\right)\right]. \tag{11}$$

DOI: 10.54254/2755-2721/2026.GU33493

Among them, $\mathcal{H}(\pi(\cdot \mid s_t)) = -\mathbb{E}_{a\sim\pi}[\log\pi(a \mid s_t)]$ is the strategic entropy, and α is the temperature parameter. The significance of entropy regularization is:

Encourage exploration: high entropy corresponds to high randomness, so that the strategy can fully explore the action space at an early stage.

Avoid premature convergence: prevent the strategy from falling into the second-best deterministic behavior.

Robustness improvement: the learned strategy is more insensitive to noise and disturbance.

Automatic temperature regulation: SAC introduces an automatic adjustment mechanism to dynamically adjust the temperature parameters by optimizing the following targets α[9]:

$$J(\alpha) = \mathbb{E}_{s_t\sim D}\left[-\alpha\left(log\pi\left(a_t s_t\right) + \bar{\mathcal{H}}\right)\right]. \tag{12}$$

Among them, $\bar{\mathcal{H}}$ is the target entropy. This allows the algorithm to balance exploration and utilization without manual parameter adjustment, further improving practicality and robustness [9].

5. The third generation of transformer-based methods

5.1. From "learning strategy" to "prediction sequence"

The emergence of Transformer architecture has brought about a paradigm shift. Its core mechanism - self-attention - is naturally suitable for capturing the long-distance dependence between sequence elements. Researchers began to explore: can decision-making problems be directly modeled into sequence prediction problems?

Kool et al. were early explorers in this direction. They use the attention mechanism-based model to directly solve the traveler problem (TSP), regard the path construction as a sequence generation process, and gradually output the node sequence through autoregression, which achieves excellent performance without relying on manual rules.

The core of the paradigm shift:

Traditional RL: $\pi(a \mid s)$ - Learn "what to do in state s"

Transformer paradigm: directly predict the complete trajectory sequence - regard decision-making as a sequence modeling problem.

This transformation laid the foundation for subsequent Decision Transformer, GATO and other work, and ushered in the era of "big model of decision-making".

5.2. Decision transformer: sequence modeling reshaping intensive learning

Decision Transformer is a milestone work that combines intensive learning and Transformer [10]. It redefined reinforcement learning as a conditional sequence modeling problem: given the past state-action-return sequence and the expected future return, the model self-regressively predicts the next action.

Unlike the traditional RL method, Decision Transformer does not learn the value function through the Bellman equation, but directly models the decision-making process into a sequence prediction problem [10]:

$$\text{Input sequence} = (\tau_1, \tau_2, \ldots, \tau_K), \tau_t = (R_t, s_t, a_t). \tag{13}$$

DOI: 10.54254/2755-2721/2026.GU33493

Among them, R_t is the expected cumulative return, s_t is the state, and a_t is the action. The model uses GPT architecture to generate action sequences in turn through autoregression [10].

The core advantage of this paradigm conversion is that there is no need for Bellman iteration, which avoids the accumulation of value function estimation errors; it can directly use large-scale offline data for training. Experiments show that Decision Transformer has achieved performance comparable to professional RL algorithms on multiple benchmark tasks [10], proving the great potential of Transformer architecture in the field of decision-making.

5.3. GATO: the attempt of universal intelligence

After Decision Transformer modeled single-task decision-making into sequence prediction, Reed et al. [11], further proposed GATO (A Generalist Agent) to try to complete a variety of different types of tasks with one model.

GATO unifies all tasks - including playing games, robot control, dialogue, etc. - into sequence modeling problems. Whether it is text, image, joint angle or button instruction, it is converted into a unified token sequence input model, and the model self-regressively predicts the next action token [11].

By serializing the data of different tasks, the same Transformer model can be trained on 604 different tasks at the same time, such as Atari games, robot control, and natural language processing, and show a certain zero-sample generalization ability [11].

GATO proves the feasibility of general intelligence and provides an important example for the development of the "big model of decision-making" [12].

6. Comparison and analysis of methods

Based on the combing of the first three chapters, the three generations of methods of deep reinforcement learning - methods based on value functions, methods based on strategy and Actor-Critic, and methods based on Transformer - have significant differences in core mechanisms, applicable scenarios and performance characteristics. As shown in Table 2, the following table systematically compares the three-generation methods from multiple dimensions.

Table 2. Method comparison

Dimension	value function-based methods	Actor-Critic Method	Transformer Method
Representative Algorithms	DQN,Double DQN	PPO,SAC,DDPG	Decision Transformer,GATO
Action Space	Discrete	Continuous/Discrete	Continuous/Discrete
Core Mechanism	Q-Value Estimation and Bellman Update	Policy Network and Value Network	Sequence Prediction and Self-Attention
Sample Efficiency	Medium	High (Off-Policy)	High (Pre-Training)
Implementation Complexity	Low	Medium	High
Scalability	Low	Medium	High
Training Stability	Stable	PPO is stable, SAC requires hyperparameter tuning	Depends on the data quality

DOI: 10.54254/2755-2721/2026.GU33493

Table 2. (continued)

Interpretability	Low	Low	Attention weights provide a certain degree of interpretability.

In general, the three-generation methods have their own emphasis: the value function-based method is suitable for introductory and simple tasks, the Actor-Critic method is the mainstream of current industrial applications, and the Transformer method represents the future direction of decision-making intelligence. In practical application, appropriate algorithms should be selected according to the characteristics of the task, data scale and computing resources.

7. Challenges and prospects

7.1. Core challenges

Kiran et al. [13], systematically summarized the main challenges of DRL in real-world deployment:

Sample efficiency: DRL algorithms usually require millions or even billions of interactions to learn effective strategies, which is expensive or even unpresible in real scenarios such as robot control and autonomous driving. How to learn efficiently with fewer samples is one of the core problems of DRL [13].

Sim2Real migration: There is a "reality gap" between the simulation environment and the real world - the performance of the strategies trained in simulation is often plummeted when deployed directly to the real system. Domain randomization, domain adaptation and other methods have partially alleviated this problem, but it has not been fundamentally solved [13].

Safety assurance: In key security areas such as autonomous driving and medical care, DRL's exploration behavior may bring disastrous consequences. How to carry out efficient learning under the premise of ensuring safety is a key problem that must be solved by the practicalization of DRL [13].

Interpretability: As a "black box", the decision-making process of deep neural networks is difficult for humans to understand, which limits the application of DRL in high-risk fields [13].

7.2. Future direction

Zha et al. [14], systematically sorted out the new development opportunities of DRL in the basic model era:

Large model + DRL: Large language model has strong reasoning and planning ability, which can be used as a high-level decision-making module of DRL intelligence, decompose complex tasks into sub-task sequences, and then DRL performs specific actions [14].

RLHF: Use human preference data to train reward models, and then use RL to optimize strategies - this is the core technology of large models such as ChatGPT to align human values, which has become an important bridge between large models and reinforcement learning [14].

World model: By learning the environmental dynamics model, the agent can plan and learn in "imagination", which greatly improves sample efficiency. The world model combined with Transformer is one of the current research hotspots [14].

Multi-intelligent bodies and large models: Introducing large models into multi-intelligent bodies systems for communication, coordination and negotiation between intelligent bodies is another direction worth paying attention to [15].

DOI: 10.54254/2755-2721/2026.GU33493

8. Conclusion

This article takes "from value function to Transformer" as the main line, and systematically sorts out the development vein of deep reinforcement learning in the past ten years.

The first-generation value function-based method, represented by DQN, realizes end-to-end learning through experience replay and target networks, laying the methodological foundation of DRL. The second-generation Actor-Critic method directly optimizes the strategy network. PPO achieves stability and efficiency by cutting and replacing the target. SAC has become the mainstream of continuous control tasks through the maximum entropy framework to balance exploration and utilization. The third-generation Transformer method reconstructs decision-making problems into sequence prediction, and Decision Transformer and GATO show the potential of general intelligence.

The three generations of methods have their own emphasis: the value function-based method is simple to implement; the Actor-Critic method is widely applicable; and the Transformer method is scalable. DRL still faces challenges such as sample efficiency, Sim2Real, security, etc. Integration with large models, RLHF, world models and other directions will bring new development opportunities to DRL.

References

[1] V. Mnih et al., "Human-level control through deep reinforcement learning, " Nature, vol. 518, no. 7540, pp. 529–533, 2015.
[2] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
[3] R. Ghimire and D. K. L. Karna, "A comprehensive review of reinforcement learning, " Mid-West Univ. J. Eng. Innov., vol. 1, no. 1, pp. 1–13, 2025.
[4] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning, " in Proc. AAAI, 2016, pp. 2094–2100.
[5] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization, " in Proc. ICML, 2015, pp. 1889–1897.
[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms, " 2017. [Online]. Available: arXiv: 1707.06347
[7] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning, " in Proc. ICLR, 2016.
[8] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, " in Proc. ICML, 2018, pp. 1861–1870.
[9] T. Haarnoja et al., "Soft actor-critic algorithms and applications, " 2018. [Online]. Available: arXiv: 1812.05905
[10] L. Chen et al., "Decision transformer: Reinforcement learning via sequence modeling, " in Proc. ICML, 2021, pp. 1684–1697.
[11] S. Reed et al., "A generalist agent, " Trans. Mach. Learn. Res., 2022.
[12] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems, " in Proc. ICLR, 2019.
[13] B. R. Kiran et al., "Deep reinforcement learning for autonomous driving: A survey, " IEEE Trans. Intell. Transp. Syst., vol. 23, no. 6, pp. 4909–4926, Jun. 2022.
[14] D. Zha, Z. P. Bhat, J. Lai, Y. Feng, and A. Mukhopadhyay, "Deep reinforcement learning in the era of foundation models: A survey, " Computers, vol. 15, no. 1, p. 40, 2026.
[15] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning, " IEEE Signal Process. Mag., vol. 34, no. 6, pp. 26–38, Nov. 2017.