Towards Reliable Deep Reinforcement Learning for Industrial Applications: A DDPG-based Algorithm with Improved Performance

Mahdi Dolati and Negin Sayyaf[*]

Department of Electrical Engineering, Faculty of Engineering, University of Isfahan, Isfahan, Iran

Abstract:

This paper proposes Improved Model-Based Deep Deterministic Policy Gradient, a novel reinforcement learning algorithm designed to overcome three critical challenges in industrial deep reinforcement learning applications: (1) poor sample efficiency requiring excessive real-world trials, (2) safety risks from unstable policies during training, and (3) difficulty scaling to high-dimensional continuous control spaces. Building on DDPG's strengths for continuous control, the proposed algorithm introduces four key innovations: (i) a virtual environment for data-efficient learning, (ii) a simulation rate mechanism adapting model reliance dynamically, (iii) a simulated experience buffer preventing divergence, and (iv) a performance threshold for fail-safe operation. Evaluated on Cart-Pole benchmark via OpenAI Gym python library, the suggested method demonstrates faster convergence than standard DDPG while maintaining performance degradation under sensor malfunctions or communication losses. These improvements derive from the algorithm's unique ability to simultaneously leverage real-world data and model-generated experiences, reducing physical trial costs while ensuring operational safety. The results establish the novel framework as a practical solution for industrial control systems where reliability and data efficiency are paramount, particularly in applications like chemical process control and precision robotics that demand stable operation amid sensor/communication failures.

Keywords:

[*] Corresponding author's email: n.sayyaf@eng.ui.ac.ir

Deep Reinforcement Learning, Model-Based Method, Deep Deterministic Policy Gradient, Industrial Applications, System Identification

1. Introduction

Industrial control systems serve as a critical component in maintaining operational efficiency and reliability across various applications. Conventional control strategies, including Proportional-Integral-Derivative (PID) controllers [1] and model-based techniques [2], have long been central to industrial automation due to their stability and consistent performance [3]. These approaches are widely adopted for their simplicity, well-established theoretical foundations, and predictable behavior in stable, well-defined processes. However, as industrial environments grow increasingly complex, there is a rising demand for more intelligent and adaptive control solutions that can surpass the limitations of traditional methods [4].

In more details, modern industrial systems - ranging from petrochemical processing plants and fluid power systems to renewable energy infrastructure, automated manufacturing cells, and precision machining equipment - frequently demonstrate highly nonlinear dynamic behavior while being subject to numerous exogenous disturbances [5-7]. These complex systems must maintain continuous operation while adhering to stringent safety protocols and optimal efficiency requirements. Furthermore, time-dependent factors such as mechanical wear, component degradation, and frictional effects introduce additional uncertainties through progressive changes in system dynamics. Consequently, industrial control environments present fundamental challenges characterized by: (1) strong nonlinearities, (2) complex dynamic couplings, (3) rigorous operational constraints, (4) temporal parameter variations, and (5) significant uncertainty factors [8, 9].

The effectiveness of conventional control strategies diminishes significantly when applied to the above-mentioned systems with time-varying parameters, unmodeled disturbances, or rapidly changing operational objectives. While demonstrating excellent performance in static environments with predictable dynamics, these methods lack the cognitive flexibility required for real-time adaptation to evolving system conditions

[10-12]. This inherent rigidity has spurred growing interest in more sophisticated control paradigms capable of autonomous learning and self-optimization [13, 14].

The inherent constraints of classical control methodologies have motivated a fundamental shift toward intelligent control strategies, facilitated by recent breakthroughs in artificial intelligence [15, 16]. While early reinforcement learning approaches like SARSA [17] and TD(λ) [18] laid important theoretical foundations, the integration of deep learning with RL [19] has enabled transformative capabilities in complex control domains. This synergy has given rise to Deep Reinforcement Learning (DRL), which has emerged as a paradigm-shifting approach for industrial systems [20-23], offering three key advantages:

- **Autonomous Policy Learning**: Through iterative environmental interactions, DRL systems develop adaptive control policies that overcome the rigidity of conventional techniques [18].
- **Complex Environment Handling**: DRL excels in nonlinear, time-varying, and partially observable environments that challenge traditional methods [24].
- **Continuous Optimization**: The framework maintains optimal performance amid dynamic conditions without manual intervention [25].

These capabilities have produced landmark algorithms (e.g., AlphaGo [26], NFQ [27]) that surpass human performance in domains ranging from game theory to autonomous control [25, 28]. The essence of DRL's success lies in its biomimetic approach – mirroring human experiential learning while achieving superhuman precision in high-dimensional state spaces [24].

While deep reinforcement learning provides a robust framework for autonomous decision-making through iterative interaction, its practical implementation in industrial settings faces significant hurdles. Three critical challenges merit attention:

- **Sample Efficiency**: Industrial systems often involve costly or time-intensive data acquisition, making extensive trial-and-error learning impractical. As instance, direct applying DRL to physical robotic arms incurs substantial temporal and material costs, unlike simulated environments, due to its sample inefficiency [23].

- **Safety Assurance**: Industrial applications demand rigorous safety guarantees, as unstable control policies may lead to hazardous outcomes. This necessitates supplementary safety mechanisms beyond core DRL algorithms. More critically, maintaining operational stability during the learning phase presents significant challenges, particularly when dealing with sensor inaccuracies, communication latency or data packet losing issues [29-31].
- **High-Dimensional Continuous Spaces**: Real-world systems typically exhibit complex state-action spaces with numerous continuous variables (e.g., multiparameter chemical processes involving temperature, pressure, and flow dynamics). Such dimensionality increases the learning complexity [32].

To contribute the mentioned gap, this study introduces an enhanced Deep Deterministic Policy Gradient (DDPG) framework—selected for its proven capability in handling high-dimensional continuous state and action spaces, a hallmark of industrial control systems [32]. The proposed algorithm, termed "Improved Model-Based Deep Deterministic Policy Gradient (IMB-DDPG)", incorporates an online system identification module and novel components to mitigate vulnerabilities from sensor or communication failures. This architecture maintains an adaptive model that serves as both a backup data generator during system disruptions and a virtual environment to supplement training data—significantly improving reliability and data efficiency for industrial deployment.

The framework's design is motivated by critical industrial requirements where sensor malfunctions or communication losses can cause severe operational failures. By integrating model-based learning with real-world interactions, IMB-DDPG reduces both the costs and risks of physical exploration during training. This dual approach not only enhances robustness against unexpected disturbances but also minimizes the need for risky real-world trials through simulated experience.

The remainder of this paper is structured as follows. Section 2 provides a comprehensive review of deep reinforcement learning fundamentals, with particular emphasis on the DDPG algorithm's theoretical foundations. The following section presents a detailed exposition of the proposed IMB-DDPG framework, systematically examining its novel components and architectural innovations. The fourth section evaluates the framework's performance through extensive experiments using the cart-pole benchmark, including comparative analyses with baseline DDPG. Finally, Section 5 concludes with key findings and outlines promising directions for future research.

2. Deep Reinforcement Learning: Theoretical Foundations and Algorithmic Evolution

In this section, the preliminaries of deep reinforcement learning algorithms are analytically investigated.

2-1- Reinforcement Learning: Basics

Reinforcement Learning, a pivotal branch of machine learning, enables agents to learn optimal decision-making policies through trial-and-error interactions with their environment [18]. In more details, reinforcement Learning problems are formally modeled as Markov Decision Processes (MDPs) [33], defined by the tuple (S, A, P, R, γ), where S , A , P , R and γ respectively denote the state space, action space, state transition dynamics, reward function and discount factor. At each time step t , the intelligent agent observes the environment state s_t , selects an action a_t via its policy π ; after that, the agent receives reward r_{t+1} and encounters the resultant state s_{t+1} and so on. The objective of the learner agent is to find the optimal policy that maximizes the expected cumulative reward or discounted cumulative reward in the long-term [18].

$$R_{cumulative} = E\left\{\sum_{t=0}^{\infty} \gamma^t r_t\right\}, \tag{1}$$

where $E\{.\}$ represents the expected value.

It is worth noting that reinforcement learning algorithms are broadly categorized into model-free (MF) and model-based (MB) approaches. MF methods, which dominate the RL landscape, learn directly through environmental interaction without estimating system dynamics [34]. These algorithms iteratively refine policies or value functions based on accumulated experience, offering implementation simplicity and lower computational complexity by avoiding explicit model learning. However, this comes at the cost of higher sample inefficiency and slower convergence.

Conversely, MB algorithms (e.g., Dyna-Q, trajectory sampling [35, 36]) leverage environmental data to approximate system dynamics, enabling faster policy improvement with fewer interactions. While theoretically more sample-efficient, MB methods face practical challenges such as model inaccuracies when the learned dynamics diverge from real-world behavior. Notably, the foundational RL framework is inherently model-free, with MB approaches constituting a specialized subset designed to address MF limitations.

2-2- Q-learning: A Model-Free Cornerstone

The evolution of value-based RL began with Q-learning [37], a foundational model-free algorithm that estimates the action-value function $Q(s,a)$ through temporal difference updates:

$$\begin{cases} Error = r_{t+1} + \gamma \cdot \max_{a} Q(s_{t+1},a) - Q(s_t,a_t) \\ Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \cdot Error \end{cases}, \quad (2)$$

where α is the learning rate and γ is the discount factor [38].

Its model-free nature and guaranteed convergence (under discrete state-action conditions) made it widely adoptable, with variants like Double Q-learning [39] addressing overestimation biases.

The tabular nature of conventional Q-learning imposes a fundamental constraint: it is only applicable to problems with finite, discrete state and action spaces. This limitation significantly restricts its utility in real-world industrial applications, where continuous domains are prevalent. While discretization of continuous spaces offers a potential solution, this approach inevitably leads to the curse of dimensionality - where computational complexity grows exponentially with increasing resolution of the discretized space [18].

2-3- Deep Q-Networks (DQN): Scaling with Neural Networks

Q-learning's limitations in handling continuous domains motivated the development of Deep Q-Networks

(DQN), revolutionized reinforcement learning by replacing traditional Q-tables with deep neural networks to approximate $Q(s,a)$ in high-dimensional spaces [18, 32]. The DQN architecture introduced two pivotal mechanisms that significantly improved stability and convergence [40-42]:

- **Experience Replay**: A memory buffer that stores and randomly samples past transitions, effectively decorrelating sequential observations and improving data efficiency.
- **Target Networks**: Periodic copies of the main Q-network that provide stable temporal difference targets, mitigating harmful feedback loops during training.

To this end, DQN algorithm as the initiator of deep reinforcement learning, uses the following loss function at iteration i .

$$L_i(\theta_i) = E_{(s,a,r,s') \sim U(D)}\left[\left(r + \gamma \cdot \max_{a'} Q(s',a';\theta_i') - Q(s,a;\theta_i)\right)^2\right] \tag{3}$$

In the latter equation, experiences drawn uniformly from dataset D (replay buffer), where θ_i and θ_i' respectively denote the weights of the Q-network and target Q-network at iteration i . Specifically, the parameters of Q-network are adjusted to reduce mean-squared error, which error is the difference between predicted value of Q-network and target value, similar to Q-learning.

As the first successful integration of deep learning with reinforcement learning, DQN demonstrated that neural networks could effectively approximate action-value functions $Q(s,a)$, receiving state vectors as input and outputting value estimates for discrete actions.

Despite its success in discrete domains (e.g., Atari games [43]), DQN's inability to handle continuous actions limited industrial applicability.

2-4- Deep Deterministic Policy Gradient (DDPG): Bridging the Gap

This gap inspired Deep Deterministic Policy Gradient (DDPG), an actor-critic architecture that combines the DQN-inspired value estimation (for continuous state spaces), policy gradient methods (for continuous action outputs) and the model-free flexibility with improved sample efficiency [33]. The DDPG algorithm is a model-free, off-policy and actor-critic method, which was introduced in 2016 [44, 45]. DDPG was built from combining both the value-based and the policy-based methods, and injecting deep NNs in the Deterministic Policy Gradient (DPG) algorithm [46]. In short, DDPG algorithm extends Q-learning's principles to continuous control by concurrently optimizing:

- A critic network (Q-function approximator)
- An actor network (policy approximator)
- Target networks for training stability

In DDPG procedure, the actor network receives a vector as state and outputs selected action; in other words, the actor network is the representation of policy. The critic network is also a neural network that takes state and action as input and outputs the predicted value of state-action pair, i.e., it is similar to the action value function in Q-leaning method. More precisely, the actor's objective function defined by

$$J(\mu_\omega) = \int_s \rho^B(s) \,.\, V^{\mu}(s)\, ds = \int_s \rho^B(s) \,.\, Q^{\mu}\left(s, \mu_\omega(s);\theta\right)\, ds\,, \tag{4}$$

maximizes the expected return, where Q-values are provided by the critic network. In the latter equation, B is the behavior policy, ρ^B is the state distribution of the behavior policy, and μ is the deterministic policy which parameterized by ω. The mentioned objective function essentially optimizes the actor network's parameters to maximize the expected value of actions across the state distribution encountered by the agent. For the critic network, the loss function is

$$L(\theta) = E_{(s,a,r,s') \sim U(D)}\left[\left(y - Q(s,a;\theta)\right)^2\right], \tag{5}$$

where

$$y = r + \gamma \,.\, Q(s', \mu(s';\omega');\theta')\,, \tag{6}$$

in which ω' denotes the parameters of target-actor network. Specifically, the loss is similar to DQN loss, except that in target term y, the value of action that taken by target actor, i.e. $Q(s', \mu(s'; \omega'); \theta')$, has been used instead of max operator.

As previously discussed, most of the real-world problems and industrial processes are continuous in terms of state and action spaces. Thus, DDPG's ability to learn policies in continuous control tasks (e.g., robotic manipulation, industrial automation) while retaining sample efficiency makes it ideal for real-world systems [18].

3- The Novel IMB-DDPG Algorithm

This section introduces the novel Improved Model-Based DDPG (IMB-DDPG) algorithm through four systematic development stages, detailing its architectural enhancements that overcome industrial control challenges. The proposed modifications address critical limitations of standard DDPG in manufacturing environments, particularly regarding sample efficiency, safety constraints, and fault tolerance. The section concludes with an evaluation of IMB-DDPG's advantages in operational robustness and adaptive control, while acknowledging its computational requirements and implementation considerations specific to industrial automation scenarios.

3-1- Virtual Environment

A fundamental limitation of deep reinforcement learning algorithms, including DDPG, lies in their inherent data dependency. These methods typically require extensive interaction with the environment to properly tune neural network weights for accurate function approximation, resulting in prolonged periods of suboptimal performance during initial exploration phases. This sample inefficiency poses significant challenges for industrial applications where real-world data collection is costly or potentially hazardous. While traditional model-free approaches gradually improve through trial-and-error accumulation in experience buffers, their practical utility remains constrained in industrial settings with stringent safety and efficiency requirements.

To address these limitations, utilizing the model-based techniques can be beneficial. MB methods, such as Dyna-Q [36], propose a structure that benefits from data, both to improve approximation of the value function, i.e. direct RL, and to estimate the dynamics of the environment, i.e. model learning. Given that most real-world problems can be modeled as MDPs with continuous spaces, and considering DDPG's efficacy in such domains, this paper suggests some novel extensions to improve performance of DDPG method to solve industrial problems.

As the most important improvement to DDPG algorithm, in addition to using data directly to train the DRL agent, the collected data should also be utilized to estimate the dynamics of the environment. This new method is called *Model-Based Deep Deterministic Policy Gradient (MB-DDPG)*. To this end, in addition to four networks (actor, actor target, critic, critic target), another network called *Virtual Environment (VE)* is created. The purpose of this network is to estimate the environment's dynamics, and thereupon reduce the cost of trial and error through real world applications. In this case, the agent can sometimes execute a simulated episode instead of the real one, and train the actor and critic networks similar to DDPG, by simulated experiences. To be more precise, the VE network receives state and action vectors at the current time step, i.e. s_t and a_t, as input and outputs the predicted next state, i.e. s_{t+1}, where the next action, i.e. a_{t+1}, is selected by actor network according to s_{t+1}. As this cycle continues, the intelligent agent can execute a simulated episode.

The VE network will be updated by the following innovative loss function

$$L(\beta) = E_{(s,a,r,s') \sim U(D)} \left[\left(s' - \varphi(s,a;\beta) \right)^2 \right], \tag{7}$$

where φ is the VE network which parametrized by β. According to this loss function, the weights of the VE network will be modified in such a way that minimizes the prediction error of the next state. The aforementioned network predicts the next state deterministically, so this method can be more reliable in deterministic MDPs, however, also can be used in stochastic environments.

The overall scheme of MB-DDPG algorithm is depicted in Fig. 1. Conforming to this scheme, there are two learning loops: The right learning loop is similar to the standard DDPG, where the agent interacts with the real environment and stores real experiences in the experience buffer, which results in improving the actor and critic networks. A key distinction from the standard DDPG algorithm lies in the dual utilization of real-world data: beyond policy optimization, it simultaneously learns an explicit environment model. The left learning loop is interacting with the virtual environment, where only the actor and critic networks can be updated according to data which generated by VE.

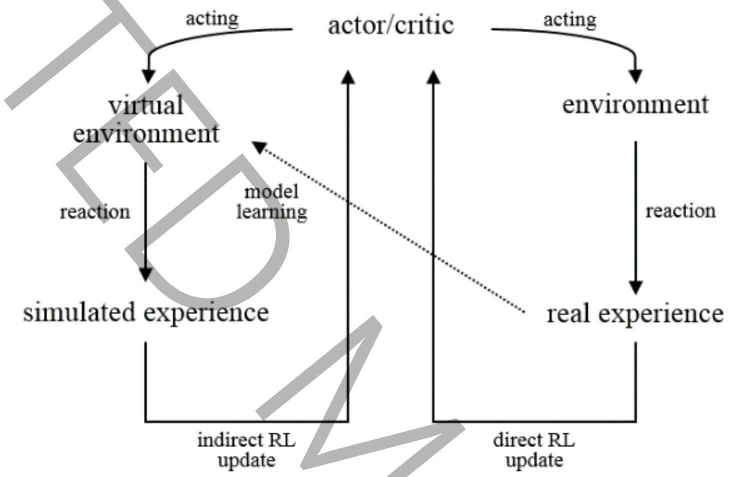

Fig. 1. The overall scheme of MB-DDPG framework

In the proposed MB-DDPG framework, the *Virtual Environment (VE)* module is designed to learn only the environment dynamics model, excluding reward function estimation. This design choice stems from practical industrial applications where rewards are typically derived directly from observable states - such as product quality metrics or robotic tracking accuracy - and can therefore be calculated from predicted states. Notably, the architecture remains flexible to accommodate reward estimation through VE modifications if required.

3-2- Simulation Rate

Despite the benefits that virtual environment provides, exploiting the VE network alongside the basic algorithm faces two fundamental pitfalls. The first challenge is the randomness of its weighting parameters at the beginning steps, due to lack of authentic data, which leads to discrepancy between the prediction and real dynamics of the environment. This difference may mislead both actor and critic networks, potentially slowing the learning process compared to the standard DDPG approach and compromising the algorithm's stability and convergence, too.

To address this challenge, a hyperparameter is suggested in the proposed IMB-DDPG algorithm, called *Simulation Rate (SR)*. This parameter represents the probability of executing a simulated episode versus real episode, after every ended episode. Simulation rate is a real number in the range $[0,1)$, which numbers near one means the high probability of interaction with the VE.

Typically, this rate can be either a constant value or a function within an acceptable range. This paper introduces the novel function (8) to dynamically adjust the parameter based on episode number. The function yields near-zero values during initial episodes, gradually increases its output, and eventually decreases after reaching a specific episode threshold. This design stems from the observation that the VE is initially untrained and inaccurate, requiring the agent to prioritize data collection for improved prediction accuracy while limiting simulated episodes. As more data is accumulated and the model's approximation accuracy improves, the rate should increase to maximize utilization of the estimated environment behavior. Furthermore, once the actor and critic networks have undergone sufficient training through VE interaction, the simulation rate decreases to enable the agent to perform its task through real episodes.

$$SR = \frac{\psi\left(\tanh\left(\eta\ (N_{episode} - EP_{\min})\right) - \tanh\left(\eta\ (N_{episode} - EP_{\max})\right)\right)}{2} \tag{8}$$

In Eq. (8), $EP_{\min}$ and $EP_{\max}$ determine the lower and upper bounds for episodes with the maximum simulation rate, and $N_{episode}$ denotes the episode number. Also, the coefficient ψ, which lies in the range [0,1), defines the maximum value of the SR function. Subsequently, the coefficient η, a positive number typically recommended to be close to zero, controls the increasing and decreasing gradient of the SR function.

To give insight, this function is depicted in Fig. 2, for $\psi = 0.6$, $\eta = 0.1$, $EP_{\min} = 50$ and $EP_{\max} = 150$, supposing a problem that agent wants to perform its task within 300 episodes. In this sample, in early real episodes, agent gathers data and executes simulated episode with lower probability (about zero). After some episodes, simulation rate increases in order to benefit from estimated behavior of the environment. In episode 100, it executes a simulated episode with highest probability (0.6). After enough training actor and critic with VE in episode 50 to 150, the simulation rate decreases in order to agent to perform its task, this time with greater skill and reliability in real environment.

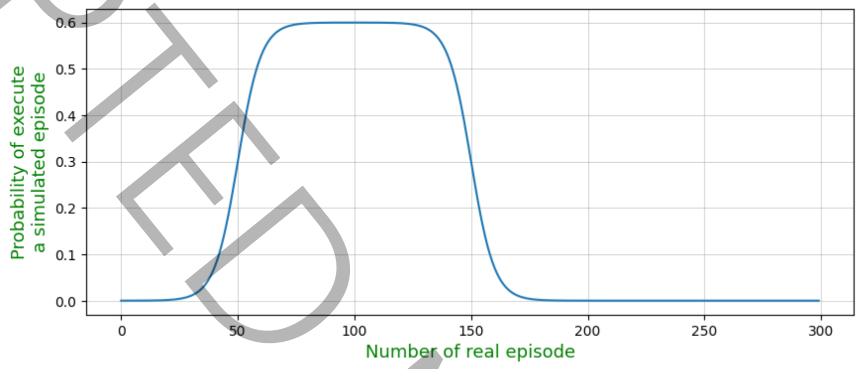

Fig. 2. The variable simulation rate introduced in Eq. (7), where $\psi = 0.6$, $\eta = 0.1$, $EP_{\min} = 50$ *and* $EP_{\max} = 150$

3-3- Simulated Experiences Buffer

As mentioned earlier, the first challenge of using VE was model mismatch in the early stages, which was resolved by SR (For more details, please see Section 3-2). Another challenge arises when simulated experiences are stored alongside real experiences in the replay buffer. If the agent samples a mini-batch uniformly from the buffer to update the actor, critic, and VE networks, the mini-batch may contain simulated experiences. Consequently, the VE network could be updated using data generated by itself, leading to bootstrapping. If the generated data significantly deviates from reality, this can cause divergence or instability in the training of the VE network, subsequently affecting the actor and critic networks.

To address this second challenge, the novel IMB-DDPG algorithm employs two separate buffers: an experience buffer, which stores real experiences (similar to the standard DDPG method), and a *Simulated Experience Buffer (SEB)*, which exclusively stores experiences generated by the VE. The data from the first buffer is used to update the actor, critic, and VE networks, while the data from the SEB is used only for updating the actor and critic—not the VE network. This separation of real and simulated data enhances the reliability and accuracy of the VE network.

Practical Consideration for SEB Size:

Empirically, the size of the simulated experience buffer should be significantly smaller than that of the real experience buffer. This is because, in the early stages of problem-solving, the model's estimation error is typically large, resulting in simulated experiences that deviate substantially from reality. As the VE's accuracy improves over time, the generated data becomes more reliable. Thus, to ensure training stability, older (and potentially inaccurate) data in the SEB should be discarded sooner rather than being used to update the actor and critic networks.

3-4- Performance Threshold

A significant limitation of applying reinforcement learning algorithms, particularly DDPG, in real-world applications is their susceptibility to performance degradation during prolonged training. Even after achieving satisfactory performance, continued learning may temporarily reduce policy effectiveness. In such cases, the weights of the actor and critic networks deviate from their optimal values, causing the DDPG agent to exhibit undesirable behavior while attempting to rediscover the optimal policy. This phenomenon can render the algorithm both costly and unreliable in practical settings [47-49].

To mitigate this issue, the proposed approach introduces a *Performance Threshold (PT)* parameter. This threshold defines a minimum performance level; if the agent's performance falls below this limit, it automatically switches from interacting with the real environment to training exclusively with the learned virtual environment. During this phase, the agent refines its actor and critic networks before returning to the real environment. For instance, in an industrial manufacturing scenario where product quality drops below the defined threshold (after previously reaching near-optimal performance), the agent would cease real-world interactions to prevent further damage. Instead, it would train in the VE until regaining sufficient competence, then resume real environment operations. This innovative approach significantly reduces the costs and risks associated with DRL deployment in industrial applications while enhancing overall reliability.

3-5- Advantages of the Novel IMB-DDPG Algorithm

Having fully developed the innovative IMB-DDPG algorithm, its key benefits and improvements over conventional DDPG are highlighted here. The integration of the virtual environment significantly enhances data efficiency, enabling faster learning while reducing both the cost and risk associated with pure DDPG implementation in real-world control systems.

As outlined in Section 1, industrial applications often face challenges such as sensor inaccuracies, communication latency, and data packet loss [29-31]. The VE effectively addresses these issues. For instance, consider a self-driving car controlled by a standard DDPG agent, where the state space includes speed, road gradient, and surrounding vehicle positions. If a sudden data disruption occurs mid-operation, a traditional RL agent would fail to act due to incomplete state observation. However, with the VE, the agent can leverage its last known state and action to predict environmental behavior, allowing it to either maintain control until connectivity is restored or execute a safe, controlled stop. This capability makes the algorithm both safer and more data-efficient.

The introduction of the simulated experience buffer represents another critical innovation, preventing VE bootstrapping and improving estimation accuracy. This component enhances the convergence and reliability of IMB-DDPG algorithm, making it a more robust solution. Additionally, the innovative SR parameter, i.e. simulation rate, minimizes the VE's initial mismatch with the real environment, while the proposed function in Eq. (8) accelerates the learning process.

Furthermore, the performance threshold parameter serves as a safeguard, automatically switching training to the VE when performance degrades, thereby increasing operational safety and reliability.

Collectively, these advancements make IMB-DDPG a superior choice for industrial control problems characterized by nonlinearity, high dimensionality, continuous state-action spaces, time variance, and uncertainty. The algorithm demonstrates greater robustness, adaptability, safety, and data efficiency compared to conventional DRL approaches, e.g. DDPG [18], TD3 [50], SAC [51] and PPO [52].

It is worth noting that distributed optimization methods effectively address problems in networked systems [53], while the centralized IMB-DDPG approach offer distinct advantages for industrial control systems:

- **Deterministic Execution**: Essential for safety-critical applications, e.g., robotic arms, where consensus delays in distributed systems may cause instability.
- **Resource Efficiency**: Eliminates inter-node communication overhead (critical in low-power edge devices)
- **Temporal Consistency**: Maintains synchronous state updates (challenging in distributed setups with clock drift)

Though distributed methods excel in fault-tolerant, geographically dispersed applications [53], IMB-DDPG's architecture specifically targets industrial scenarios requiring centralized precision. Future work may hybridize these paradigms for multi-agent systems, combining the VE-based robustness with distributed coordination.

3-6- Computational Complexity Analysis

The novel IMB-DDPG algorithm maintains the polynomial complexity of standard DDPG, i.e. $O(n^k)$, where n represents the number of neurons in the largest network layer and k depends on the network architecture (typically 2-3). While the base complexity arises from matrix operations in actor-critic networks and experience replay, the suggested modifications introduce only bounded overhead: the virtual environment requires one additional forward pass per step, i.e. $O(n)$, the simulated experience buffer and performance threshold contribute constant-time operations, i.e. $O(1)$, and the simulation rate parameter is computed in closed-form per episode via Eq. (8), i.e. $O(1)$. Consequently, the total complexity remains $O(n^k)$, preserving the original scalability while significantly enhancing reliability and sample efficiency for industrial deployment. The modular design ensures these enhancements maintain computational tractability without affecting the asymptotic complexity class- crucial for large-scale deployment.

3-7- Industrial Implementation Considerations

The IMB-DDPG architecture is designed to address critical industrial implementation constraints through three key features: First, it maintains quantization robustness by preserving DDPG's continuous action space while using the virtual environment to smooth value estimates and the simulated experience buffer to prevent error accumulation in fixed-point implementations. Second, it ensures computational efficiency through polynomial complexity, optimized memory usage via the limited-capacity SEB (10-20% of main buffer), and minimal-overhead PT operations. Finally, it provides operational reliability through the VE's prediction capability during delays, the SR's noise-resistant calculations (Eq. (8)), and SEB's implicit noise regularization. This modular design enables industrial deployment while preserving DDPG's advantages.

3-8- Comparative Analysis with State-of-the-Art Methods

To systematically evaluate the advantages of IMB-DDPG framework, Table 1 presents a comprehensive comparison with state-of-the-art DRL algorithms across key performance metrics, considering both theoretical properties and practical industrial requirements. The results demonstrate how IMB-DDPG achieves superior performance while addressing limitations of existing approaches.

Table 1. Comparative analysis of IMB-DDPG against state-of-the-art DRL algorithms (Industrial Readiness has been evaluated Based on deterministic timing, memory footprint and hardware compatibility, where ✓ and ✗ respectively denote denotes supported and not supported).

Metric	IMB-DDPG	DDPG [18, 32]	TD3 [18, 50]	SAC [18, 51]	PPO [18, 52]
Time Complexity	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
Convergence Rate	$1/T$	$1/\sqrt{T}$	$1/T$	$1/T$	$1/\sqrt{T}$
Quantization Resilience	✓	✗	✗	✗	✗
Fault Tolerance	✓	✗	✗	✗	✗
Sample Efficiency	High	Medium	High	Medium	Low
Industrial Readiness	✓	Partial	Partial	✗	✗

4- Simulation Results

In this section, the presented novelties and techniques are evaluated in details.

4.1 Benchmark Environment: Cart-Pole Dynamics

To rigorously evaluate the proposed IMB-DDPG algorithm, the cart-pole environment is employed from OpenAI-Gym python library [54] – a canonical benchmark for continuous control systems. As depicted in Fig. 3, this system models an underactuated mechanical assembly where a pole is hinged to a cart moving along a rail. The agent's objective is to apply precise horizontal forces to maintain the pole vertically upright, while preventing cart derailment. The system exhibits two equilibrium points:

- Stable equilibrium: Pole hanging downward (requires no control input)
- Unstable equilibrium: Pole balanced upward (control objective)

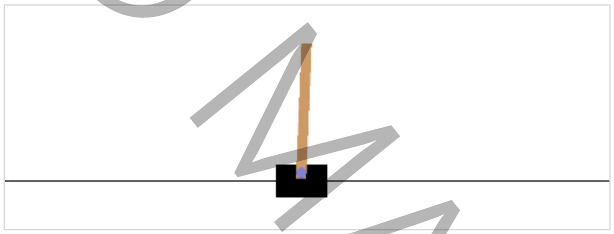

Fig. 3. One frame of cart-pole environment

The challenge lies in stabilizing the unstable equilibrium, governed by nonlinear dynamics:

$$(M+m)\ddot{x}+ml\ddot{\theta}\cos(\theta)-ml\dot{\theta}^2\sin(\theta)+b\dot{x}=u \tag{9}$$

and

$$I\ddot{\theta}+ml\ddot{x}\cos(\theta)-m\,\mathrm{g}\,l\sin(\theta)=0\,, \tag{10}$$

where x and θ respectively denote the horizontal position of the cart and angle of the pole from vertical (upright=0). In the above-mentioned equations, M, m, l, I, b and u determine mass of the cart, mass of the pole, length of the pole's center of mass, moment of inertia of the pole, viscous friction coefficient of the cart and applied horizontal force, in turn. These equations reveal:

- Under actuation: Control only through cart movement

- Nonlinear coupling: Between translational ($\ddot{x}$) and rotational ($\ddot{\theta}$) motion
- Sensitivity: Small disturbances rapidly destabilize the system

Despite its apparent simplicity, cart-pole abstracts critical challenges in [3, 55, 56]:

- Robotic arm stabilization (analogous to pole angle control)
- Overhead crane positioning (similar cart dynamics)
- Autonomous vehicle rollover prevention (related to moment arm physics)

Consequently, this modified benchmark provides an ideal testbed for evaluating IMB-DDPG's capabilities in handling nonlinear mechanical couplings, real-time stabilization requirements and imperfect sensing/actuation - all critical for advanced mechanical systems like [3, 5, 21]:

- Exoskeleton balance control (similar to pole stabilization)
- CNC machine vibration damping (related to cart oscillation control)
- Satellite attitude adjustment (analogous angular dynamics)

Since the state and action spaces in cart-pole environment are both continuous, its optimal solution can be a guide for solving many engineering and real-world problems.

4.2 IMB-DDPG Performance Evaluation

To investigate the novel MB-DDPG framework, the original cart-pole environment of the Gym library has been changed in such a way that each episode will be a maximum of 2000 time steps, and the objective of the agent is to keep the cart within the permissible limits of the rail track and to keep the pole within the permissible limits of the angle. If any of them go out of the allowed range, the current episode will end. The reward function is also defined in such a way that the agent receives reward +1 for each time step that the cart is in the middle of rail track with the pole within the permissible limits of the angle, i.e. reward +1 per timestep for maintaining pole within ±12° and cart within ±2.4 units.

It should be noted that performance of the agent has been evaluated over 500 real episodes, in all simulations. Also, for reducing the effect of randomness in the algorithm results, all modes have been runed in six different seeds, then, average and standard deviation of all of them are plotted in the graphs (seeds are 10, 20, 30, 40, 50 and 60), where the horizontal and the vertical axes in all of the following figures are respectively the episode number and the average reward, i.e. the average of successful time steps in each episode. In order to accurately examine the effect of added components in different modes, all features of actor and critic networks, e.g. structure of NNs, learning rate, exploration noise, are set the same. The IMB-DDPG algorithm is assessed in five progressive stages:

4-2-1- Evaluation of Basic DDPG Algorithm

As the first experiment, the result of agent's performance in the presence of basic DDPG algorithm is evaluated in Fig. 4. In the following figure, the solid graph is the average of 6 different run with 6 different seeds, where the shaded area represents the standard deviation and the diversity of results.

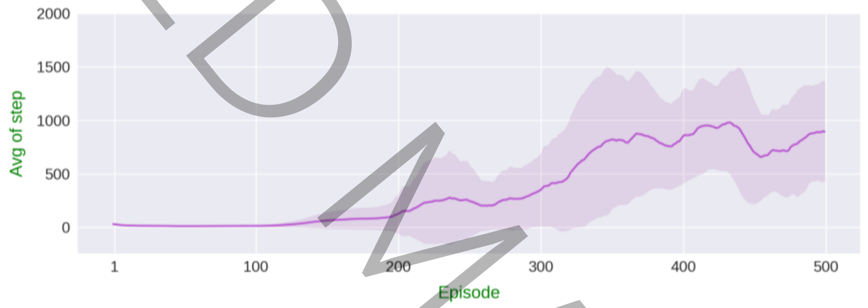

Fig. 4. Performance of the basic DDPG algorithm

As shown in Fig. 4, the agent in the first 200 episodes is gathering data through trial and error with the environment, and it doesn't perform well. In episode 320, the agent has somewhat improved performance and managed to control the system up to 500 steps on average (Since the average value is defined as the average of last 40 episodes, the average reward 500 in episode 320 means that the average number of successful steps from episode 280 to 320 is equal to 500.). But finally, the agent could not reach an average reward higher than 1000 during 500 episodes.

4-2-2- Efficacy of Adding a Virtual Environment with Constant Versus Variable Simulation Rate

As discussed in the previous section, adding another network to the DDPG algorithm as VE results in the first version of IMB-DDPG approach, named MB-DDPG. This network is supposed to approximate the dynamics of the environment and help the agent in predicting the behavior of the environment. When the mentioned virtual environment is added, it is necessary to define a simulation rate for determining the probability of the interaction with the VE. In simulation results, two different modes are evaluated. The first mode is the simulation rate with a constant value (here 0.2 is selected), and the second mode is to select the simulation rate with a variable value suggested in Eq. (8). The results are presented in Fig. 5 (In order to avoid ambiguity, it is emphasized that the horizontal axis represents the number of real episodes, and performance of agent in simulated episodes is not plotted).

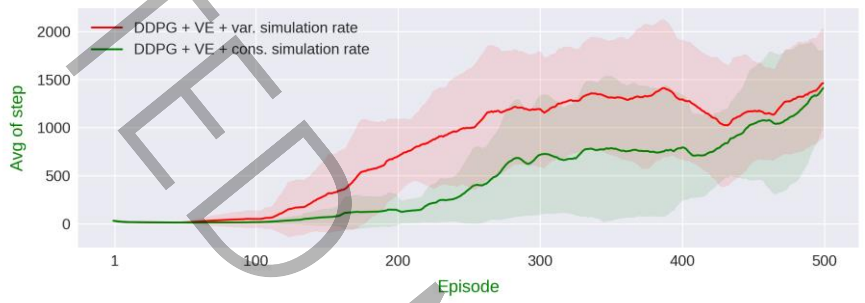

Fig. 5. Performance of the MB-DDPG algorithm with constant and variable simulation rates (The red graph indicates the MB-DDPG algorithm with variable simulation rate, where the green one shows the MB-DDPG method with constant simulation rate).

According to Fig. 5, these two MB-DDPG algorithms give better results than the pure DDPG (Fig. 4). These two MB-DDPG algorithms achieved reward 500, i.e. 500 successful steps, in less than 280 episodes, where both of them reached about 1500 steps in episode 500, too. Therefore, the results confirm that injecting a virtual environment into the DDPG algorithm can greatly improve its performance. Also, as claimed in the third section, exploiting the variable simulation rate with the suggested format can significantly increases the learning speed in comparison with the constant simulation rate.

4-2-3- Efficacy of Adding a Simulated Experience Buffer

Another novelty presented in this paper is to use a separate buffer for simulated experiences, named SEB in the previous section. To investigate the effect of this component on the performance of the novel framework, SEB added to the MB-DDPG algorithm with variable simulation rate (red graph in Fig. 5), and the results have been compared. As seen in Fig. 6, SEB can improve the actor's decisions; because VE network can be updated just through real data in this case, which yields in increasing the accuracy of the virtual environment. More details, the reward graphs demonstrate that the improved algorithm can outperform the former one, after episode 220.

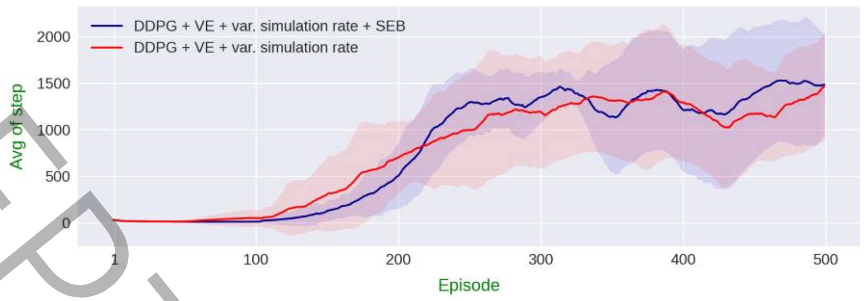

Fig. 6. Effect of adding SEB on the performance of the MB-DDPG algorithm with variable simulation rate (The red and blue graphs show the performance of the MB-DDPG algorithm with variable simulation rate, respectively in the absence and presence of the SEB, i.e. simulated experiences buffer)

4-2-4- Efficacy of Adding the Performance Threshold

In addition to the mentioned extensions for DDPG algorithm, this paper also introduced another key innovation: the performance threshold parameter. This parameter forces the agent to focus on the virtual environment when its real-world performance falls below a certain level, or during emergencies. This can analytically reduce the damage and cost caused by interaction with the real-world environment. Other words, after sufficient training of the agent in VE and achieving the desired performance, the agent interacts with the real environment again. The effectiveness of adding this threshold parameter on enhancing the performance of the latter version of MB-DDPG approach is illustrated in Fig. 7

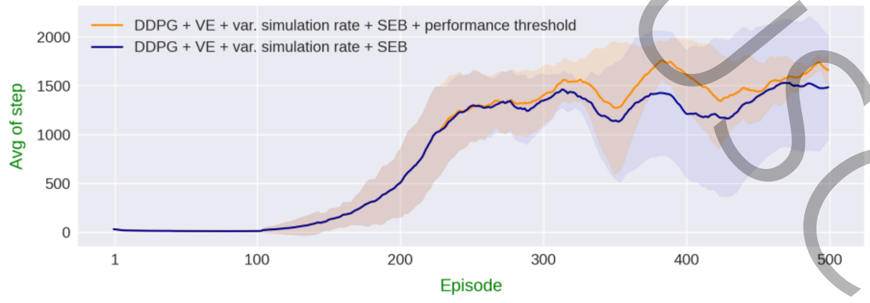

Fig. 7. Efficacy of the performance threshold parameter on the IMB-DDPG algorithm with variable simulation rate and simulated experiences buffer (The blue and orange graphs depict the MB-DDPG algorithm with variable simulation rate and SEB, respectively in the absence and presence of the performance threshold parameter)

Fig. 7 asserts that adding the performance threshold parameter to the MB-DDPG algorithm improves its efficiency, by reducing the cost of trial and error with the real-world environment, which leads to growth of received reward. To be more precise, the performance of both algorithms remains identical for the initial 220 episodes. However, a divergence occurs when the agent's average performance falls below the pre-defined performance threshold (in this case, 1100), where the embedded instruction dictates a shift in the behavior. After the agent is well trained, the agent is more likely to interact with the real-world environment. Furthermore, according to the shaded areas, adding this parameter reduces the variance of the performance to a great extent. Accordingly, such a parameter desirably augments the reliability of the suggested IMB-DDPG algorithm for industrial and real-world environments.

4-2-5- Comprehensive Comparison

For better comparison, the results of gradually adding the proposed components and parameters of the novel IMB-DDPG framework to the basic DDPG algorithm are all shown in Fig. 8. As the results affirm, the suggested novel method can impressively perform better than pure DDPG one. In addition, the presented improvements provide conditions that make this approach more practical for industrial applications.

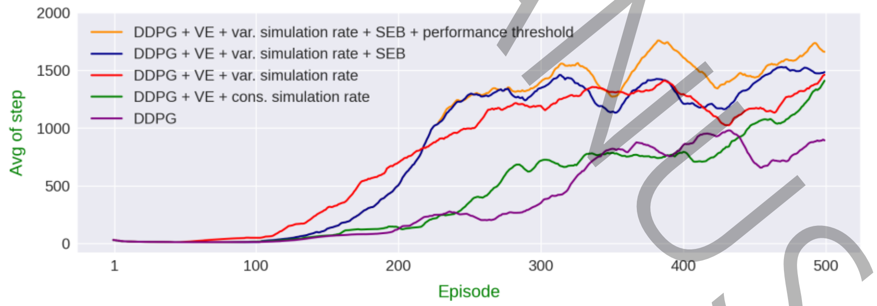

Fig. 8. The performance comparison of different versions of DDPG algorithm via the cart-pole environment

5- Conclusion

The IMB-DDPG framework represents a significant advancement in deep reinforcement learning algorithms for industrial control systems through four synergistic innovations: (1) The Virtual Environment (VE) enables sample-efficient policy improvement while reducing hazardous real-world interactions, (2) The Simulation Rate (SR) scheduler dynamically optimizes real-to-simulated experience ratios to accelerate early-stage learning, (3) The Simulated Experience Buffer (SEB) eliminates model bias through isolated transition storage, and (4) The Performance Threshold (PT) mechanism ensures operational safety by automatically reverting to VE control during performance degradation. It is worth noting that extensive validation on the Cart-Pole benchmark demonstrated these components' collective impact, with IMB-DDPG achieving target rewards much faster than standard DDPG (1500 in 310 episodes versus <1000 in 500 episodes), while maintaining robustness to sensor malfunctions or communication losses. Moreover, detailed computational complexity analysis confirms the framework maintains polynomial complexity, i.e. $O(n^2)$, despite its enhanced capabilities (Section 3.6), while industrial implementation considerations highlight its compatibility with quantized, low-power systems (Section 3.7). In addition, comparative evaluations against state-of-the-art methods further demonstrate superior behavior of IMB-DDPG's approach across key performance metrics, considering both theoretical properties and practical industrial requirements (Section 3.8). While IMB-DDPG excels in centralized industrial control, future extensions may incorporate distributed learning for multi-agent scenarios (e.g., smart grids). Also, future works could involve applying and adapting the novel components and parameters of the suggested framework for other deep reinforcement learning algorithms, like TD3, SAC and PPO.

6- Acknowledgment

Our sincere gratitude to the generous support of the Mobarakeh Steel Technology and Innovation Development (MSTID) Company, under the contract number 1402-1025222, which enabled us to carry out this study.

References

[1] K. H. Ang, G. Chong, Y. Li, PID control system analysis, design, and technology, IEEE Transactions on Control Systems Technology, 13(4) (2005) 559–576.

[2] T. Samad, A. M. Annaswamy, The impact of control technology: Overview, success stories, and research challenges, IEEE Control Systems Magazine, 40(6) (2020) 36–70.

[3] K. Ogata, Modern Control Engineering, 5th ed., Prentice Hall, 2010.

[4] A. Kuhnle, J. P. Kaiser, F. Theiß, N. Stricker, G. Lanza, Designing an adaptive production control system using reinforcement learning, Journal of Intelligent Manufacturing, 32 (2021) 855–876.

[5] M. Panda, P. K. Patra, Nonlinear control and estimation of industrial pneumatic actuator systems: A survey, ISA Transactions, 109 (2021) 177–193.

[6] Z. Su, H. Liu, Nonlinear control of robotic manipulators using adaptive neural network approaches: A review, IEEE/CAA Journal of Automatica Sinica, 8(4), (2021) 678–694.

[7] Y. Liu, Y. Jiang, Q. Zhang, Intelligent process monitoring and control of machining systems using data-driven techniques: A review, Journal of Manufacturing Systems, 56 (2020) 188–206.

[8] E. F. Camacho, C. Bordons, Model Predictive Control, Springer Science & Business Media, Springer London, 2013.

[9] K. J. Åström, R. M. Murray, Feedback Systems: An Introduction for Scientists and Engineers, Princeton University Press, Princeton, New Jersey, 2010.

[10] D. Lee, S. Koo, I. Jang, J. Kim, Comparison of deep reinforcement learning and PID controllers for automatic cold shutdown operation, Energies, 15(8) (2022) 2834.

[11] Z. Wang, T. Hong, Reinforcement learning for building controls: The opportunities and challenges, Applied Energy, 269 (2020) 115036.

[12] X. Tao, D. Zhang, W. Ma, X. Liu, D. Xu, Automatic metallic surface defect detection and recognition with convolutional neural networks, Applied Sciences, 8(9) (2018) 1575.

[13] P. J. Antsaklis, A. Rahnama, Control and machine intelligence for system autonomy, Journal of Intelligent & Robotic Systems, 91 (2018) 23–34.

[14] J. F. Arinez, Q. Chang, R. X. Gao, C. Xu, J. Zhang: Artificial intelligence in advanced manufacturing: Current status and future outlook, Journal of Manufacturing Science and Engineering, 142(11) (2020) 110804.

[15] A. Hussain, H. A. Gabbar, M. R. Khan, Digital twin-based smart monitoring and control of petrochemical processes using AI and IoT, IEEE Access, 9 (2021) 141128–141145.

[16] F. M. Bianchi, L. Livi, C. Alippi, Predictive maintenance for industrial IoT of things: A deep learning approach with attention-based RNNs, IEEE Transactions on Industrial Informatics, 17(9) (2021) 6204–6212.

[17] G. A. Rummery, M. Niranjan, On-Line Q-Learning Using Connectionist Systems, Department of Engineering, University of Cambridge, Cambridge, 1994.

[18] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction. 2nd ed., MIT Press, Cambridge, 2018.

[19] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436-444.

[20] Q. Lyu, Y. Tian, R. Zhao, S. Yin, Deep reinforcement learning for wind turbine control: Challenges and opportunities, Renewable and Sustainable Energy Reviews, 144 (2021) 110948.

[21] H. Yu, Z. Zhou, Y. Liu, C. Li, Y. Liu, Reinforcement learning in industrial applications: Recent advances and prospects, Engineering Applications of Artificial Intelligence, 105 (2021) 104398.

[22] J. R. Vázquez-Canteli, Z. Nagy, Reinforcement learning for demand response: A review of algorithms and modeling techniques, Applied Energy, 276 (2020) 115446.

[23] B. Kiumarsi, H. Modares, F. L. Lewis, A. Karimpour, A. Davoudi, Optimal and autonomous control using reinforcement learning: A survey, IEEE Transactions on Neural Networks and Learning Systems, 29(6) (2018) 2042–2062.

[24] Cs. Szepesvari, Algorithms for Reinforcement Learning, Morgan & Claypool Publishers, Switzerland, 2010.

[25] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Sallab, S. Yogamani, Deep reinforcement learning for autonomous driving: A Survey, IEEE Transactions on Intelligent Transportation Systems, 23 (6) (2021) 4909–4926.

[26] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of Go with deep neural networks and tree search, Nature, 529 (2016) 484–489.

[27] M. Riedmiller, Neural fitted Q-iteration – First experiences with a data efficient neural reinforcement learning method, in: Proceedings of the 16th European Conference on Machine Learning (ECML), Porto, Portugal, 2005, pp. 317–328.

[28] Y. Lin, Y. Liu, F. Lin, L. Zou, P. Wu, W. Zeng, H. Chen, Ch. Miao, A survey on reinforcement learning for recommender systems, IEEE Transactions on Neural Networks and Learning Systems, 35(10) (2024) 13164 - 13184.

[29] Y. J. Park, S. K. S. Fan, C. Y. Hsu, A review on fault detection and process diagnostics in industrial processes, Processes, 8(9) (2020) 1123.

[30] R. A. Gupta, M. Y. Chow, Networked control system: Overview and research trends, IEEE Transactions on Industrial Electronics, 57(7) (2010) 2527–2535.

[31] J. Garcia, F. Fernandez, A comprehensive survey on safe reinforcement learning, Journal of Machine Learning Research, 16 (2015) 1437–1480.

[32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, in: International Conference on Learning Representations (ICLR 2016), San Juan, 2016.

[33] R. Bellman, A Markovian decision process, Journal of Mathematics and Mechanics 6(5) (1957) 679–684.

[34] M. Morales, Grokking Deep Reinforcement Learning, Manning Publications, Shelter Iland, 2020.

[35] R. S. Sutton, Integrated architectures for learning, planning, and reacting based on approximating dynamic programming, in: Proceedings of the Seventh International Conference on Machine Learning, Austin, Texas, 1990, pp. 216–224.

[36] R. S. Sutton, Dyna, an integrated architecture for learning, planning, and reacting, ACM 2(4) (1991) 160–163.

[37] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, D. Hassabis, A general reinforcement learning algorithm that Masters Chess, Shogi, and Go through self-play, Science 362(6419) (2018) 1140–1144.

[38] C. Watkins, Learning from Delayed Rewards, PhD thesis, University of Cambridge, England, 1989.

[39] H. V. Hasselt, Double Q-learning, Advances in Neural Information Processing Systems 23 (2010) 2613–2621.

[40] L. J. Lin, Self-improving reactive agents based on reinforcement learning, planning and teaching, Machine Learning, 8(3-4) (1992) 293–321.

[41] J. O'Neill, B. Pleydell-Bouverie, D. Dupret, J. Csicsvari, Play it again: Reactivation of waking experiences and memory, Trends in Neurosciences, 33(5) (2010) 220–229.

[42] Z. Zhang, R. Li, Q-value-based experience replay in reinforcement learning, Knowledge-Based Systems 315(2025) 113296.

[43] V. Manih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518(7540) (2015) 529–533.

[44] V. R. Konda, J. N. Tsitsiklis, On actor critic algorithms, SIAM Journal on Control and Optimization 42(4) (2003) 1143–1166.

[45] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, M. Lee, Natural Actor Critic Algorithm, Automatica 45(11) (2009) 2471–2482.

[46] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: Proceedings of the 31st International Conference on Machine Learning, PMLR 32(1), 2014, PP. 387–395.

[47] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning (ICML), 2018, pp. 1582–1591.

[48] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, D. Meger, Deep reinforcement learning that matters, in: AAAI Conference on Artificial Intelligence, 2018, pp. 3207–3214.

[49] A. Zhang, N. Ballas, J. Pineau, A dissection of overfitting and generalization in continuous control, ArXiv Preprint, arXiv:1806.07937 (2018).

[50] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor critic methods, in: Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018, pp. 1587–1596.

[51] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018, pp. 1861–1870.

[52] Y. Gu, Y. Cheng, C. L. P. Chen, X. Wang, Proximal policy optimization with policy feedback, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 52(7) (2022) 4600–4610.

[53] M. Doostmohammadian, M. I. Qureshi, M. H. Khalesi, H. R. Rabiee, U. A. Khan, Log-scale quantization in distributed first-order methods: Gradient-based learning from distributed data, IEEE Transactions on Automation Science and Engineering 22 (2025) 10948–10959.

[54] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, OpenAI Gym, ArXiv Preprint, arXiv: 1606.01540 (2016).

[55] Ch. Lei, Q. Zhu, R. Li, Cascaded robust fixed-time terminal sliding mode control for uncertain cartpole systems with incremental nonlinear dynamic inversion, International Journal of Non-Linear Mechanics 167 (2024) 104900.

[56] R. V. Florian, Correct Equations for the Dynamics of the Cart-Pole System, Center for Cognitive and Neural Studies (Coneural), Romania, 2007.