Journal of Machine Learning and Information Security
https://www.sciltp.com/journals/jmlis

Review

A Survey of Learning in Optimal Control and Differential Game

Yuxuan Zhu, Chuandong Li * and Runtian Zeng

School of Electronic and Information Engineering, Southwest University, Chongqing 400715, China
* Correspondence: cdli@swu.edu.cn

How To Cite: Zhu, Y.; Li, C.; Zeng, R. A Survey of Learning in Optimal Control and Differential Game. *Journal of Machine Learning and Information Security* **2026**, *2*(1), 4. https://doi.org/10.53941/jmlis.2026.100004

Received: 10 December 2025
Revised: 12 February 2026
Accepted: 27 February 2026
Published: 13 March 2026

Abstract: With the widespread applications of large-scale multi-agent systems, optimal control and differential game have become essential components of modern control theory. However, traditional methods often struggle with the inherent challenges posed by high dimensionality when addressing high-dimensional problems. The rapid development of deep learning has provided new solution ideas and methods to address this challenge. This paper reviews the research status and progress of solution methods for optimal control and differential game. First, this review elaborates on the fundamental theoretical frameworks of optimal control theory for continuous-time systems and differential game. Second, this paper introduces in detail two main deep learning methods: Deep Reinforcement Learning (DRL) and Physics-Informed Deep Learning (PIDL). Based on this, this study analyzes the specific applications of these two methods in addressing the aforementioned problems. Finally, this article summarizes the main problems and limitations of existing research and points out future research directions.

Keywords: multi-agent systems; optimal control; differential game; deep reinforcement learning (DRL); physics-informed deep learning (PIDL)

1. Introduction

As two major streams of modern control research, optimal control and differential game methodologies have seen broad deployment across diverse areas. For instance, in autonomous driving systems, optimal control is applied to vehicle path tracking [1]; in intelligent manufacturing, it is utilized to address complex scheduling problems [2]. In the field of UAV target capture and tracking, differential game theory provides a theoretical foundation for achieving optimal cooperative capture strategies among multiple UAVs in dynamic environments [3]. Moreover, in the energy systems domain, differential game theory is applied to assess the dynamic interactions among various agents [4]. These studies indicate that optimal control and differential game theory are not only the basic theoretical pillars of modern engineering, but also a powerful way to optimize complex dynamic systems.

Optimal control is a mathematical approach aimed at determining a control strategy that can optimize the performance index of the system. Game theory refers to the mathematical framework specially used to analyze the strategic decision-making of multiple subjects under mutual influence [5]. Each agent chooses a suitable strategy according to its own objectives and its expectations regarding the actions of other agents, with the goal of maximizing their individual interests. Differential game theory is a dynamic decision-making framework developed on the basis of game theory. Its core is to incorporate the system dynamics described by differential equations into it. Within a continuous-time framework, the system states evolve according to these equations, while agents influence the evolution through their control strategies to achieve the optimization of performance indices.

Traditionally, the methods used to solve optimal control and differential game problems mainly include Pontryagin's Minimum Principle (PMP), the Hamilton–Jacobi–Bellman (HJB) equation, and Dynamic Programming. Traditional methods possess a solid theoretical foundation and typically exhibit high solution accuracy and reliability in solving low-dimensional systems. However, these methods still have certain limitations when applied to complex high-dimensional systems: such techniques are apt to run into the challenges of dimensionality growth — as the system dimensionality increases, the computational cost rises sharply, which in turn results in a substantial drop in solution efficiency. The "curse of dimensionality" arises from two disparate origins: first, high-dimensional state spaces pose a critical challenge: as the scale of state variables expands, a rise in the quantity of state variables per agent results in a substantial surge of computational burden; second, large-scale multi-agent systems, in which an increasing number of agents causes the complexity of policy interactions to grow exponentially. In addition, traditional methods usually rely on accurate mathematical models, which may limit their practical application. Therefore, when addressing large-scale or model-free control problems, traditional techniques often fail to achieve the expected results. In order to overcome the above obstacles, researchers have begun to study the use of deep learning methods to solve differential game and optimal control problems.

In recent years, deep learning, with its powerful generalization ability and excellent scalability, has shown great potential in solving optimal control and differential game problems, effectively alleviating the limitations of traditional methods in high-dimensional system solutions. This review roughly classifies the existing research on deep learning-based optimal control and differential game into two categories: Deep Reinforcement Learning (DRL) methods and Physics-Informed Deep Learning (PIDL) methods. DRL is a method that integrates Reinforcement Learning (RL) and deep learning [6], utilizing deep neural networks to approximate value and policy functions, thereby alleviating the shortcomings of traditional RL when operating in high-dimensional state spaces. PIDL is a method that incorporates physical laws into the deep learning framework, combining with physics-based constraints to optimize the accuracy and reliability of deep learning models [7]. DRL and PIDL offer new methods for optimal control and differential game problems, especially in high-dimensional and complex scenarios where conventional approaches face limitations. In summary, this survey systematically reviews the applications of DRL and PIDL in optimal control and differential game problems, which are analyzed in terms of their theoretical foundations, algorithmic frameworks, and main advantages, as well as the challenges that remain in current research.

This paper's remaining sections are arranged as follows. Section 2 introduces the fundamental concepts, theorems, and algorithms related to optimal control, differential game, DRL, and PIDL. The applications of DRL and PIDL to differential game and optimal control issues are covered in Sections 3 and 4, respectively. The main conclusions of this work are outlined in Section 5, which also suggests possible directions for further investigation.

2. Fundamental Conceptions

2.1. Optimal Control

The optimal control problem is a fundamental concept in the fields of engineering, with its primary objective being to determine the optimal control strategy for a dynamic system by minimizing a predetermined performance index while satisfying all constraints [8]. While performance index offer the standards for assessing control effectiveness, constraints establish the system's operational bounds. They jointly determine the optimal control strategy.

Consider a dynamic system described by:

$$\begin{cases} \frac{dx(t)}{dt} = f(x(t), u(t), t), \\ x(t_0) = x_0, \\ u(t) \in U \subseteq \mathbb{R}^m. \end{cases} \tag{1}$$

where $x : [t_0, t_f] \to \mathbb{R}^n$ denotes the state vector of the system; $u \in U \subset \mathbb{R}^m$ represents the control variable, with U being the admissible control set; $f : [t_0, t_f] \times \mathbb{R}^n \times U \to \mathbb{R}^n$ is the dynamics function that determines how the state changes over time; $\frac{dx(t)}{dt}$ denotes its time derivative; x_0 represents the system's initial state at time t_0.

The performance index is given as follows:

$$J = \phi(x(t_f)) + \int_{t_0}^{t_f} C(x(t), u(t), t)\mathrm{d}t, \tag{2}$$

where $C(t, x(t), u(t))$ is the instantaneous utility function, and $\phi(x(t_f))$ is the terminal utility function.

The main objective is to minimize the performance index J, which is accomplished by determining the optimal control $u^*(t)$, i.e.,

$$J(t, x^*(t), u^*(t)) \leq J(t, x(t), u(t)), \tag{3}$$

$x^*(t)$ indicates the state trajectory associated with the optimal control input $u^*(t)$. The following is an alternate way to formulate the optimal control issues:

$$u^*(t) = \arg \min_{u(t)\in U} J. \tag{4}$$

2.2. Differential Game

A differential game is a continuous-time dynamic game model [9], the core of which resides in the system state evolution being characterized by differential equations. Unlike single-agent optimal control, a differential game involves multiple agents ($M \geq 2$) whose decisions mutually influence each other. The primary distinction lies in the objective function formulation: in optimal control, a single performance index J is minimized; whereas in differential game, each agent i aims to optimize its own cost J_i which is coupled with the strategies u_{-i} of all other agents. Each agent selects a continuous control strategy over time to regulate the system's dynamic progression, thereby maximizing its own payoff.

An M-player differential game can be formulated as follows. The system dynamics are described by:

$$\begin{cases} \frac{dx(t)}{dt} = f\left(x(t), u_1(t), u_2(t), \cdots, u_M(t), t\right), \\ x(t_0) = x_0. \end{cases} \tag{5}$$

where $x(t) \in \mathbb{R}^n$ denotes the state vector and $u_i(t) \in U_i \subseteq \mathbb{R}^{m_i}$ stands for the control variable corresponding to player i.

For each agent i, the cost function J_i is usually formulated as the sum of the integral of the running cost C_i and the terminal cost ϕ_i.

$$J_i(u_i, u_{-i}) = \phi_i(x(t_f)) + \int_{t_0}^{t_f} C_i(u_i, u_{-i})dt, \tag{6}$$

where u_i is the strategy of agent i, while u_{-i} represents each agent's combined strategy set, with the exception of agent i.

The following is the definition of the Nash equilibrium for this differential game: let u_i^* represent the best control approach for each agent, and for all $(i = 1, 2, \ldots, M)$, the following condition is satisfied:

$$J_i(u_i^*, u_{-i}^*) \leq J_i(u_i, u_{-i}^*) \quad \forall i \in \{1, \ldots, M\}, \tag{7}$$

where u_i^* represents the optimal strategy for agent i, and u_{-i}^* is the set of optimal strategies for all agents except i. In other words, if all other agents follow their optimal strategy u_{-i}^*, then no agent can unilaterally stray from u_i^* to attain a lower cost.

2.3. Mean Field Game (MFG)

The core idea of MFG theory is that when the agent count tends to an infinite value, the interactions among individual agents can be described by the average behavior of all agents, which is represented by the probability distribution of the population states [10]. This approach greatly simplifies the evaluation of large-scale multi-agent systems. It substitutes a single mean-field term that characterizes the general population behavior for direct pairwise interactions between agents. Within the framework of MFG, consider the state of any individual in the system as $x(t) \in \mathbb{R}^n$ (where t stands for time and n stands for the dimension of the state). The dynamic evolution of this state is depicted via the subsequent stochastic differential equation:

$$\begin{cases} dx(t) = f\big(x(t), u(t), t\big)dt + g\big(x(t), t\big)dW(t), \\ x(t_0) = x_0 \in \mathbb{R}^n, \\ u(t) \in U \subseteq \mathbb{R}^m. \end{cases} \tag{8}$$

$x(t)$ stands for the individual state vector, $u(t)$ refers to the control input, f signifies the drift coefficient, g corresponds to the diffusion coefficient, and $W(t)$ denotes the standard Brownian motion.

The MFG system comprises the backward Hamilton-Jacobi-Bellman (HJB) equation coupled with the forward Fokker-Planck-Kolmogorov (FPK) equation. These two equations make up a system of coupled Partial Differential Equations (PDEs) [11]. The FPK equation explains the evolution of the population distribution, while the HJB equation solves the optimal control. A concise summary of the MFG system is presented below.

$$-\partial_t V - \frac{g^2}{2}\Delta V + H(x, m, \nabla V) = 0, \tag{9}$$

$$\partial_t m - \frac{g^2}{2}\Delta m - \text{div}\left(\nabla_p H(x, m, \nabla V) m\right) = 0, \tag{10}$$

$$m(t_0) = m_0, \quad V(x, t_f) = \phi\left(x, m(t_f)\right). \tag{11}$$

Equation (9) presents the backward Hamilton–Jacobi–Bellman (HJB) equation that governs the value function $V(x,t)$ over the domain $\mathbb{R}^n \times [t_0, t_f]$, where $x \in \mathbb{R}^n$ denotes the n-dimensional state vector and t represents time. The coefficient $\frac{g^2}{2}$ corresponds to the viscosity term induced by the diffusion coefficient g defined in Equation (8), and Δ denotes the Laplace operator. The Hamiltonian $H(x, m, \nabla V)$ is characterized by the Legendre–Fenchel transform associated with the Lagrangian. In the arguments of the Hamiltonian, x is the spatial state variable, m denotes the population density, and ∇ represents the gradient operator. Equation (10) is a forward PDE defined in the same domain $\mathbb{R}^n \times (t_0, t_f)$, reflecting the evolution of the population distribution $m(x,t)$ over time, as determined by the divergence operator and the gradient of the Hamiltonian with respect to momentum $abla_p H$. Equation (11) specifies the boundary conditions of the system, including the initial value $m(t_0) = m_0$ and the terminal value $V(x, t_f) = \phi(x, m(t_f))$.

2.4. Deep Learning

As an important area of machine learning, deep learning constructs its foundation on artificial neural network structures, which use the backpropagation algorithm to update network parameters so that each layer of neurons can be iteratively optimize according to the output of the preceding layer [12]. Deep learning has been widely adopted across diverse fields, including computer vision [13], natural language processing [14], and large language model [15].

The following mathematical formulas can be used to characterize the input layer, intermediate hidden layers, and final output layer that make up a neural network's architecture.

$$\begin{cases} \mathcal{H}^0(x) = x, \\ \mathcal{H}^l(x) = \sigma(W^l \mathcal{H}^{l-1}(x) + b^l), \\ \hat{y}(x;\theta) = W^l \mathcal{H}^{-1}(x) + b^l. \end{cases} \tag{12}$$

These formulas describe the forward propagation process of data changing in order in each layer.

Let $\mathcal{H}^l(x)$ denote the output of the l-th hidden layer, with W^l and b^l standing for the associated weight matrix and bias vector, respectively. The input to the network is denoted by x, which corresponds to the output of the 0-th layer $\mathcal{H}^0(x)$. $\sigma(\cdot)$ is an activation function, and the final output of the network after l layers is denoted by $\hat{y}(x;\theta)$, where θ is the set of all learnable parameters of the network.

2.5. Reinforcement Learning

RL is a machine learning technique used to solve issues represented by Markov Decision Process (MDP). Each MDP comprises a state space S, an action space A, a state transition probability $P(s'|s,a)$, a reward function $R(s,a)$, and a discount factor $\gamma \in [0,1)$. In this approach, each agent learns an optimal policy $\pi^*(a|s)$ that maximizes the predicted cumulative reward by interacting with the environment through trial and error.

The only used RL algorithms are as follows.

2.5.1. Q-Learning Algorithm

Q-Learning is centered on temporal difference (TD) learning. This algorithm iteratively updates the action-value by computing the TD error, which is the difference between the present prediction of $Q(s,a)$ and the sum of the immediate reward and the optimal future value. Through interaction with the environment, the agent can make $Q(s,a)$ gradually converge to the optimal action-value $Q^*(s,a)$, and ultimately obtain the optimal policy [16]. The updating mechanism for Q-Learning is as follows [17]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)], \tag{13}$$

where α is the learning rate, which is used to control the update magnitude of $Q(s,a)$.

This process is repeated until convergence, ultimately yielding a Q-table $Q(s,a)$ that approximates the optimal policy.

However, when the state or action space grows big, the dimension of the Q-table grows exponentially, making storage and computation intractable. DRL replaces tabular representations with parameterized function approximators, eliminating the need to enumerate all state–action pairs. By using deep neural networks to directly take continuous or high-dimensional states as inputs and output the corresponding action-value estimates, the number of trainable parameters is mainly determined by the network architecture rather than growing exponentially with the state dimension. As a result, efficient learning can be achieved in large-scale or continuous-state environments. To address this issue, the Deep Q-Network (DQN) algorithm extends Q-learning by employing a deep neural network to approximate the action-value function $Q(s, a; \theta)$. Furthermore, it includes experience replay techniques and a target network to stabilize training and allow for effective learning in environments with high-dimensional continuous state spaces but discrete action spaces.

Two neural networks with the same architecture are used in the algorithm: the online network parameterized by θ and the target network parameterized by θ^-. The online network $Q(s, a; \theta)$ is employed to select actions and to perform gradient updates. In contrast, the target network $Q(s, a; \theta^-)$ is employed to select the maximum Q-value of each subsequent state, thereby constructing the TD target.

The mean squared error loss between the estimated Q-value and the TD target is minimized in order to update the network parameters:

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right]. \tag{14}$$

Although DQN exhibits certain advantages in high-dimensional state spaces, its core mechanism relies on maximizing the action-value function, which makes it difficult to apply in continuous action spaces. Actor-critic frameworks and policy gradient approaches, on the other hand, can directly output actions in continuous action spaces, avoiding the computational difficulty of Q-value maximization and making them more appropriate for intricate continuous control problems.

2.5.2. Policy Gradient Methods

Instead of depending on value function estimates, policy gradient methods are a fundamental class of RL algorithms that optimize the policy function directly. Gradient descent is used in policy gradient techniques to minimize a cost function $J(\theta)$ for policy $\pi_\theta(a \mid s)$ with parameter θ. The definition of the cost function is as follows:

$$J(\theta) = -\mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \gamma^t R_t\right], \tag{15}$$

$\tau = (s_0, a_0, R_0, s_1, a_1, R_1, \dots)$ refers to a trajectory consisting of states, actions, and rewards in temporal sequence; $\mathbb{E}_{\tau \sim \pi_\theta}$ denotes the expectation over trajectories τ under policy π_θ; R_t is the immediate reward; $\sum_{t=0}^{T} \gamma^t R_t$ stands for the cumulative discounted reward of a single trajectory across time steps 0 to T. Based on the Policy Gradient Theorem [18], the gradient expression of the cost function $J(\theta)$ is as follows.

$$\nabla_\theta J(\theta) = -\mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) Q^{\pi_\theta}(s_t, a_t)\right]. \tag{16}$$

$\nabla_\theta \log \pi_\theta(a_t|s_t)$ represents the gradient of the log probability of policy π_θ selecting action a_t in state s_t with respect to θ. $Q^{\pi_\theta}(s_t, a_t)$ is the action-value function for state s_t and action a_t under policy π_θ. In accordance with the Policy Gradient Theorem, the policy parameters θ are iteratively updated using gradient descent as follows: $\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta)$, where α is the learning rate. By continuously repeating the process of sampling trajectories and updating parameters until the policy converges, we can approximate the optimal policy.

2.5.3. Actor-Critic Algorithm

The benefits of both value-based and policy-based approaches are combined in Actor-Critic algorithms, a hybrid approach in RL. This dual network structure makes more stable and efficient learning possible, especially in high-dimensional or continuous action space. The Actor-Critic framework consists of two neural networks: The Actor is a policy network, denoted as $\pi_\theta(a|s)$. After mapping the given state to the action distribution, it samples an action from the action distribution. Its goal is to adjust its parameters θ to minimize the cumulative cost, with the instantaneous cost defined as $-R$ where R is the reward from the environment. The Critic is a value network, which evaluates the performance of the current policy and guides the Actor to make improvements.

The two main categories of value-based and policy-based techniques are combined in Actor–Critic. The actor, parameterized by θ, defines the policy $\pi_\theta(a|s)$, while the critic, parameterized by ω, estimates the value function $V_\omega(s)$. In each iteration, the agent samples a state s and then chooses an action a based on the distribution $a \sim \pi_\theta(a|s)$, then interacts with the environment to obtain the immediate cost R and the next state s'. The TD error is computed as $\delta = -R + \gamma V_\omega(s') - V_\omega(s)$. The critic is updated according to $\omega \leftarrow \omega + \beta\,\delta\,\nabla_\omega V_\omega(s)$, and the actor is updated by $\theta \leftarrow \theta - \alpha\,\nabla_\theta \log \pi_\theta(a|s)\,\delta$ [19]. This iterative process continues until convergence.

2.5.4. Integral Reinforcement Learning (IRL)

IRL is a RL method suitable for continuous systems with unknown system model parameters [20]. Its core idea is to derive the integral Bellman equation by performing time integration on the Bellman equation. In the policy evaluation step, this method does not rely on the system's drift term dynamics; it only needs to use the system's state and control data over a time period T to learn the value function. The Bellman equation in integral form is as follows:

$$V(x(t)) = \min_{u(\tau)} \int_t^{t+T} r(x(\tau), u(\tau))d\tau + V(x(t+T)). \tag{17}$$

$r(x(\tau), u(\tau))$ indicates the instantaneous cost incurred at time τ; $u(\tau)$ indicates the control action carried out at time τ; and $V(x(t))$ indicates the optimal value function associated with state $x(t)$ at time t. This formula enables IRL to learn optimal control strategies directly from measurement data, making it suitable for real-world control problems that are difficult to obtain accurate system models.

2.6. Physics-Informed Neural Networks (PINNs)

PINNs are a machine learning approach designed to solve PDE problems. Their core idea is to incorporate physical constraints into the loss function of the neural network [21], so that the network learns to approximate the PDE solution by minimizing the loss during training. The total loss function of PINNs is typically defined as:

$$\mathcal{L}(\theta) = \lambda_{\text{PDE}} \cdot \mathcal{L}_{\text{PDE}}(\theta) + \lambda_{\text{BC}} \cdot \mathcal{L}_{\text{BC}}(\theta) + \lambda_{\text{IC}} \cdot \mathcal{L}_{\text{IC}}(\theta), \tag{18}$$

where λ is a weight, $\mathcal{L}_{\text{PDE}}(\theta)$ penalizes the residual loss when the network solution does not satisfy the PDE. $\mathcal{L}_{\text{BC}}(\theta)$ and $\mathcal{L}_{\text{IC}}(\theta)$ are the constraint loss terms for boundary conditions and initial conditions, respectively. The choice of these weight coefficients is critical, as they directly influence the training stability and the convergence rate of the neural network.

The gradient descent approach is used to iteratively update the model parameters θ in order to minimize the loss function $\mathcal{L}(\theta)$. The update rule is formulated as:

$$\theta_{k+1} = \theta_k - \eta \cdot \nabla_\theta \mathcal{L}(\theta_k). \tag{19}$$

The gradient of the loss function is represented by η and the learning rate is symbolized by $\nabla_\theta \mathcal{L}(\theta_k)$. In the field of computer science and engineering, this gradient-based optimization technology can help PINNs solve problems.

3. Deep Learning Methods for Optimal Control

3.1. DRL Methods

Traditional optimal control solution methods often face the following limitations in practical scenarios: First, the inherent complexity of systems and the uncertainty of external environments make it difficult to establish accurate prediction models [22]; Second, traditional solution methods are less robust and struggle with high-dimensional problems. DRL provides a novel solution for solving optimal control problems [23], with its core advantages lying in three aspects: effectively avoiding the "curse of dimensionality", exhibiting high robustness, and eliminating the need to depend on an accurate system model.

In the previous research, due to the limitations imposed by the "curse of dimensionality" and nonlinear characteristics when solving the HJB equation directly, researchers developed Adaptive Dynamic Programming (ADP) to circumvent the challenge of solving the HJB equation directly. Its main concept is to approximate the value function related to the HJB equation using the Critic network, the control strategy using the Actor network, and then progressively converge to the ideal solution of the equation using an iterative approximation technique. Based on the ADP framework, Heydari [24] proposed adjusting the function approximator parameters offline through a learning algorithm to approximate the value function. When using impulse actuators to solve spaceship rendezvous difficulties, this technique is crucial.

With the developing of research, researchers are conducting further exploration based on ADP. Zhou et al. [25] proposed an Actor-Critic approach for high-dimensional HJB-type elliptic PDEs that is based on neural networks. This method reconstructs the HJB partial differential equation into an optimal control problem and employs the policy gradient method along with the least-squares temporal difference method for solution, successfully addressing high-dimensional problems that are difficult to solve with traditional approaches.

For certain optimal control problems that are partially unknown and subject to input constraints, Modares et al. [26] created an IRL algorithm using the Actor-Critic framework, incorporating experience replay to effectively increase the algorithm's convergence speed and update the Critic network weights. For continuous-time nonlinear systems with input saturation, Xue et al. [27] proposed an event-triggered control method based on IRL. This method does not rely on the system's drift dynamics, designs a single critic neural network, and employs a new weight update rule. By designing the triggering rules of the event-triggered mechanism, the method ensures the stability of the system. Mu et al. [28] addressed the optimal control problem for certain unknown nonlinear systems and proposed an event-sampling IRL algorithm with a novel dynamic event-triggering strategy by introducing dynamic variables. This algorithm, based on policy iteration techniques and the Actor-Critic architecture, not only guarantees system stability but also prevents Zeno behavior. Li et al. [29] combined the IRL algorithm with a Dynamic Event Triggering mechanism to address the distributed optimal control problem of heterogeneous vehicle platoons without requiring accurate knowledge of vehicle dynamics models, thereby improving system efficiency.

For multi-agent systems with communication delays, Liang et al. [30] proposed a hybrid impulsive control strategy. The core of this strategy is to integrate deep learning with the Q-Learning algorithm to construct a Double Deep Q-Network (DDQN), and thus solves the prespecified-time formation tracking challenge of such systems.

Ren et al. [31] combine Contrastive Pretraining Reinforcement Learning with the Lambert algorithm. To overcome the convergence difficulties caused by sparse rewards during training, they first employ the contrastive pretraining approach, and then optimize the policy parameters using the policy gradient method, in order to solve the issues of rendezvous and multi-pulse orbital interception under various limitations. Nasir and Durlofsky [32] proposed a universal control strategy framework based on DRL and employed the Proximal Policy Optimization (PPO) algorithm to address the associated optimization problems. This approach simplifies the repetitive optimization process of traditional CLRM into a single policy that can achieve real-time decision-making after just one training session, thereby reducing computational costs. To further address the challenge of high computational complexity in the pathwise differentiation method under high-dimensional scenarios, Wang et al. [33] proposed an unbiased stochastic gradient estimator called the generator gradient estimator. This estimator can significantly improve efficiency while not affecting the estimation variance. Dixit and Elsheikh [34] proposed a multi-level RL framework, and based on this framework, they introduced a multi-level PPO algorithm. Compared with traditional algorithms, this algorithm can significantly reduce computational load and effectively reduce costs.

3.2. PIDL Methods

Raissi et al. [35] proposed PINNs and designed two different types of algorithms for continuous-time and discrete-time, addressing data-driven forward and inverse problems of partial differential equations, with their effectiveness validated in a series of classical problems.

Subsequently, García-Cervera et al. [36] proposed a novel mesh-free algorithm based on PINNs, effectively addressing the high-dimensional controllability problem by parameterizing the solution of PDEs as a deep neural network and obtaining optimal parameters through network training. Building upon this basis, Furfaro et al. [37] focused the application of PINNs from general PDE controllability problems to closed-loop guidance and control in the aerospace field. They proposed a numerical framework combining PINNs with the Theory of Functional Connections, which learns the solution of the HJB equation to synthesize optimal closed-loop guidance and control strategies, thereby overcoming the "curse of dimensionality" associated with traditional methods. Similarly, to address the conflict between security and performance, Tayal et al. [38] created a machine learning framework based on physics that treats the simultaneous optimization of security and performance as a state-constrained limited optimum control issue. This framework can also be applied to high-dimensional systems, and its central idea is to minimize the performance cost function under the assumption that the system state always stays within the safe range. Mowlavi and Nabi [39] proposed extending the PINNs framework to PDE-constrained optimal control problems. Prior to applying a two-step line search approach to identify the key weights of the loss function, the forward issue serves as a reference for choosing an appropriate PINNs architecture and training parameters. To address the limitations of traditional PINNs in practical control applications, Antonelo et al. [40] proposed the Physics-Informed Neural Network Control framework, which incorporates system initial states and control signals as additional inputs. It achieves variable-length time domain simulation by setting the initial state of the subsequent time domain as the final predicted state of the previous time domain. Demo et al. [41] proposed extending the physics-informed supervised learning strategy to PDEs. The core of this method lies in introducing "additional features" and the "physics-informed architecture" for parametric PDE-constrained optimal control problems. Additionally, in order to solve high-dimensional stochastic control issues built using dynamic programming and physics-informed learning, Jiao et al. [42] also proposed a deep learning-based algorithm. The core of this algorithm lies in introducing path operators related to the HJB equation and designing two numerical methods, namely a single neural network and a dual neural network.

Existing methods of applying PINNs to PDEs and optimal control still face challenges such as insufficient optimization efficiency and loss balancing. To address this, Müller and Zeinhofer [43] proposed adopting the paradigm of "optimize first, then discretize" for scientific machine learning optimization problems. The core of this paradigm lies in selecting an appropriate optimization algorithm in the infinite-dimensional function space and then discretizing it, thereby significantly improving optimization accuracy and efficiency. Song et al. [44] proposed combining the Alternating Direction Method of Multipliers (ADMM) with PINNs, namely the ADMM-PINNs framework, aiming to solve optimization problems constrained by nonsmooth PDEs. However, current approaches face notable challenges in addressing the intricate nonlinear dependencies between the optimization objective and PDE constraints, Hao et al. [45] proposed a Bi-level Physics-Informed Neural Network framework to solve PDE-constrained optimization problems. The inner loop uses PINNs to solve the PDE constraints; the outer loop independently optimizes the objective function based on the optimal states from the inner loop. Wang and Wu [46] proposed a model-free framework called Physics-Informed Reinforcement Learning, it solves the infinite-horizon optimal control issue for general nonlinear systems with input constraints by integrating Lyapunov stability theory into the loss function and using PINNs to estimate the value function and control strategy separately. To further address the challenge of relying on real data in optimal control problems, Kamtue et al. [47] proposed the CalVNet framework. This framework designs deep neural networks based on the fundamental theorem of calculus of variations, incorporates the necessary conditions satisfied by the optimal functions derived from variational calculations into its deep architecture, and can solve optimal control problems in an unsupervised manner without relying on real data.

Due to the fact that PINNs require repeated training when dealing with PDE problems involving multiple scenarios and parameters, the computational cost is relatively high. Operator learning serves as a promising approach to tackle this challenge, since it seeks to learn a direct mapping from input parameters to associated solutions. The fundamental difference between PINNs and operator learning lies in their mapping objectives. Under particular initial and boundary conditions, PINNs usually learn a mapping from spatiotemporal coordinates to state variables in order to estimate the solution of a given PDE. In contrast, operator learning aims to learn the underlying operator that maps function spaces, such as initial conditions or control functions, to the corresponding solution space. Wang et al. [48] proposed a self-supervised framework based on Physics-Informed Deep Operator Networks (DeepONets). It can not only handle continuous infinite-dimensional control functions but also quickly solve PDE-constrained optimization problems without requiring input-output training data. Shukla et al. [49] employed DeepONets neural operators to infer the flow field over airfoils for shape-constrained optimization, utilizing a modular integrated design framework. This framework uses hyperparameterized neural operators as surrogate models with strong generalization capabilities, significantly reducing both generalization error and computational cost compared to high-dimensional CFD solvers. In addition, Song et al. [50] proposed two enhanced primal-dual approaches: one featuring an enlarged step size and the other leveraging operator learning, which tackle PDE-constrained nonsmooth optimal control problems. Subsequently, Hwang et al. [51] proposed a general two-stage framework for solving PDE-constrained optimal control problems. The first stage involves solution operator learning for PDE constraints, while the second stage searches for the optimal control. This framework not only significantly reduces computational costs but also applies to both data-driven and data-free scenarios. Chen and Wu [52] proposed a data-driven modeling framework based on spatiotemporally continuous neural dynamical operators. Through the use of gradient-based and derivative-free approaches, this framework is able to align short-term trajectories and efficiently capture the long-term dynamic behavior of the system.

When accelerating PDE-constrained optimization problems, neural operators are confronted with problems of inefficiency and instability. To address this, Négiar et al. [53] developed a differentiable constraint layer—PDE-Constrained-Layer—that can be integrated into any neural network architecture. By directly imposing PDEs as hard constraints, compared to soft constraint methods, this approach not only requires fewer iterations but also achieves lower PDE residuals and relative errors. Building on this foundation, Cheng et al. [54] proposed a framework integrating optimization-oriented training, enhanced derivative learning, and hybrid optimization. This framework accurately learns operators and their derivatives, thereby significantly enhancing the robustness of gradient-based neural operator optimization. LUO et al. [55] proposed the multi-input reduced basis derivative informed neural operator (MR-DINO) to address optimization under uncertainty constrained by PDEs. MR compresses high-dimensional random parameters and PDE solutions into a low-dimensional space, while DINO can accurately represent the derivatives with respect to the optimization variables and the relationship between random parameters and optimization variables and the PDE state. This method offers the advantages of high accuracy and low computational cost. In addition, Go and Chen [56] designed a precise, scalable, and computationally efficient framework built upon DINO. By reducing the parameter dimension through derivative-informed dimensionality reduction and replacing the parameter-to-observable map and their derivatives with DINO, they solved the Bayesian optimal experimental design problems with infinite-dimensional parameter inputs that were restricted by PDEs.

4. Deep Learning Methods for Differential Game

4.1. DRL Methods

In order to overcome the dependence on precise system models and the "curse of dimensionality" in traditional differential game solutions, researchers have proposed using RL to solve differential game problems. Su et al. [57] proposed an online adaptive event-triggered control framework based on IRL to address multi-player non-zero-sum games in nonlinear continuous-time systems with partially unknown dynamics. This approach ensures system stability while facilitating computational resource conservation. On this basis, Wang and Zhang [58] proposed a hierarchical network structure based on post-experience replay and the reachable domain method. By using a method of centralized training and hierarchical execution, networks at all levels can learn in sparse reward environments, addressing the convergence challenges in RL training under sparse rewards. The proposed method elevates training efficiency while reinforcing its stability.

With the number of players tending toward infinity, the complexity of direct solution grows substantially, and MFG serves as a new method to address this difficulty. Chen et al. [59] proposed a novel framework that, in the context of interbank contacts, formulates the bank interest rate problem as a major-minor MFG. The core of this framework is the use of deep Q-networks (DQN) to solve the major-minor MFG problem, which not only overcomes the limitations of existing algorithms restricted to finite state spaces but also applies to continuous state spaces.

4.2. PIDL Methods

Due to the fact that existing numerical methods easily lead to the "curse of dimensionality" through spatial discretization, to address this issue, Ruthotto et al. [60] presented a machine learning framework that made it possible to solve mean-field control (MFC) and MFG models numerically. This framework combines Lagrangian and Eulerian perspectives and leverages neural network parameterization to solve high-dimensional problems in MFG and MFC.

In general-sum differential games involving two players and state constraints, the safety performance of PINNs is often found to be insufficient. This issue stems from value function discontinuities caused by state or temporal logical constraints. To resolve this issue, Zhang et al. [61] proposed three improved methods based on PINNs: hybrid learning, value-hardening, and epigraphical technique. These methods effectively enhance generalization capability and safety performance. Chen et al. [62] addressed the challenge of solving the mean field equilibrium (MFE) in spatio-temporal MFG by proposing a framework that combines RL with physics-informed deep learning (RL-PIDL), as well as a purely PIDL-based framework. Compared with traditional numerical methods, this approach can significantly enhance spatio-temporal resolution and is applicable to both monotone and non-monotone MFG scenarios. To address the MFG problem in high-dimensional scenarios, Yin et al. [63] proposed an online interactive physics-informed adversarial network, which leverages the variational structure of MFG to transform dynamic games into static optimization problems. The approach solves the value function and density function through two mutually adversarial online physics-informed networks and introduces a self-attention mechanism, significantly improving training efficiency.

Based on the previous findings, Zhang et al. [64] introduced the Pontryagin Neural Operator (PNO) to resolve the convergence problems in differential games with state constraints—problems caused by the value function's large Lipschitz constant. This operator requires no manual adjustment of supervised data or prior knowledge, and features low computational cost and high safety performance.

To address the computational challenges faced by MFG, Liu et al. [65] proposed a scalable learning framework based on the Physics-Informed Neural Operator (PINO). The core of this framework lies in utilizing PINO to solve the forward-backward PDE system generated by MFG. Compared to the traditional PINNs method, this approach significantly enhances both generalization capability and efficiency. Similarly, Zeng et al. [66] proposed a three-stage solution framework based on Physics-Informed Operator Learning (PIOL) and a two-stage solution strategy that integrates PIOL with DRL. Both approaches build operator networks based on DeepONets and Variational Autoencoders, tackling the numerical challenge of solving MFG equilibria in autonomous vehicle congestion mitigation scenarios.

5. Conclusions

This paper systematically reviews the fundamental theoretical frameworks and algorithmic advances in the fields of optimal control and differential game, with a particular focus on the applications of DRL and PIDL in addressing these problems. Through an analysis of existing studies, it can be observed that deep learning methods demonstrate great potential in handling high-dimensional, nonlinear, and uncertain dynamic systems, offering new perspectives for solving optimal control and differential game problems. Due to its model-free nature, DRL is suitable for dynamic scenarios with unknown or partially known system models and real-time decision-making requirements. In contrast, PIDL incorporates known physical laws as prior knowledge into the learning framework and is therefore more suitable for offline optimization problems with explicit physical constraints and high demands on safety and solution accuracy.

However, several limitations still exist. Overall, the theoretical foundation of deep learning methods in these areas requires further refinement, and their algorithmic stability remains to be improved due to the model's numerical instability during training, including convergence difficulties, vanishing gradients and exploding gradients. In addition, the computational efficiency in high-dimensional complex systems continues to pose a challenge.

To address the aforementioned limitations, future research may focus on the following three aspects. In terms of algorithmic stability, adaptive weighting mechanisms can be introduced to improve the design of the loss function, thereby balancing the contributions of the PDE residuals and the initial and boundary condition terms in PINNs, and alleviating numerical instability caused by improper weight settings during training. Neural operator-based techniques can be used to learn the direct mapping from system parameters to optimal control policies in terms of computational efficiency, allowing for real-time decision making in novel settings without retraining. In terms of engineering applications, further work needs to be done to extend the theoretical framework to complex practical scenarios such as multi-agent cooperative pursuit–evasion games and coordinated control of autonomous vehicles.

Figure 1 summarizes the deep learning frameworks employed for solving optimal control and differential game problems, highlighting the roles of DRL and PIDL together with their representative algorithmic architectures.

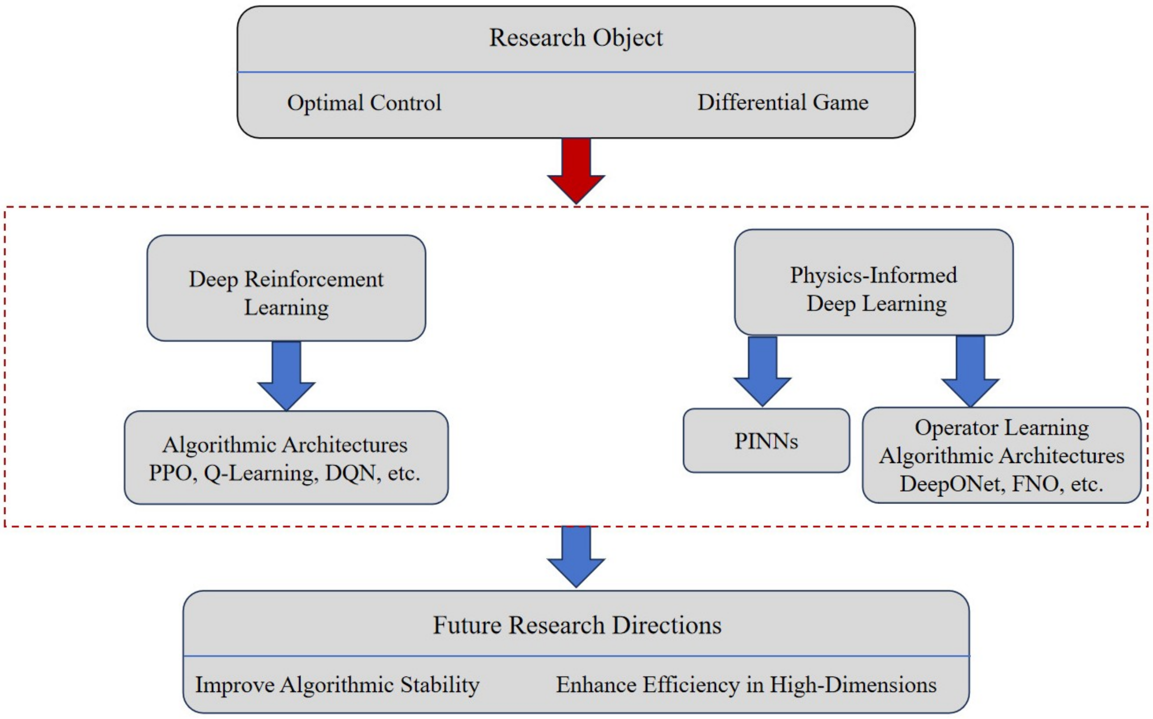
Research Object
Optimal Control
Differential Game
Deep Reinforcement Learning
Physics-Informed Deep Learning
Algorithmic Architectures PPO, Q-Learning, DQN, etc.
PINNs
Operator Learning Algorithmic Architectures DeepONet, FNO, etc.
Future Research Directions
Improve Algorithmic Stability
Enhance Efficiency in High-Dimensions

Figure 1. Deep learning frameworks for optimal control and differential game.

Table 1 summarizes selected authors and their key contributions to the fields of optimal control and differential game. It should be noted that Table 1 lists only representative examples rather than a complete collection, and many other valuable works also contribute significantly to the development of this field [67].

Table 1. Some representative authors and their contributions

Authors/References	Research Interests	Contributions
Zhou, et al. [25]	Reinforcement Learning; Deep Learning; Optimal Control; Mean field control and games	Actor-critic framework for high-dimensional static HJB equations
Modares, et al. [26]	Cyber-physical Systems; Reinforcement Learning; Cooperative Control Systems; Robotics	Integration of experience replay with IRL
Dixit and Elsheikh [34]	Reinforcement learning; AI; Fluid Control; Uncertainty Quantification	Multilevel RL framework for PDE-based control
Raissi, et al. [35]	AI for Science; Uncertainty Quantification; Machine learning (ML)	Proposal of the PINNs framework
Demo, et al. [41]	Scientific Computing; Applied Math; ML; Model Order Reduction	Proposal of the Physics-Informed Architecture (PI-Arch)
SONG, et al. [44]	Optimal Control; Optimization; ML; PDE Constrained Optimization	ADMM-PINNs algorithmic framework
Hao, et al. [45]	ML; AI for Science; physics-informed ML; RL	Bi-level optimization framework (BPN) for PDECO
Wang, et al. [48]	Scientific ML; Deep Learning; Operator Learning	Self-supervised operator learning for PDE constraints
Ruthotto, et al. [60]	Scientific Computing; Inverse Problems; PDE-constrained optimization; ML	Mesh-free ML framework for high-dimensional MFGs

Author Contributions

Y.Z.: methodology, data curation, writing—reviewing and editing; C.L.: conceptualization, methodology; R.Z.: investigation, supervision. All authors have read and agreed to the published version of the manuscript.

Funding

This work was supported in part by the National Natural Science Foundation of China under Grant 62373310.

Conflicts of Interest

The authors declare no conflict of interest.

Use of AI and AI-assisted Technologies

During the preparation of this review, the authors used OpenAI's ChatGPT to refine the language. After employing this tool, the authors carefully reviewed and revised the content as necessary and take full responsibility for the final published version of the article.

References

1. Wu, B.; Liu, Y.; Wang, Q. Path Tracking of Autonomous Vehicle Based on Optimal Control. *World Electr. Veh. J.* **2025**, *16*, 340.
2. Sokolov, B.; Dolgui, A.; Ivanov, D. Optimal Control Algorithms and Their Analysis for Short-Term Scheduling in Manufacturing Systems. *Algorithms* **2018**, *11*, 57.
3. Chi, Y.; Dong, Y.; Zhang, L.; et al. Collaborative Control of UAV Swarms for Target Capture Based on Intelligent Control Theory. *Mathematics* **2025**, *13*, 413.
4. Dong, J.; Bao, A.-R.-H.; Liu, Y.; et al. Dynamic Differential Game Strategy of the Energy Big Data Ecosystem Considering Technological Innovation. *Sustainability* **2022**, *14*, 7158.
5. Owen, G. *Game Theory*; Emerald Group Publishing: Leeds, UK, 2013.
6. Wang, X.; Wang, S.; Liang, X.; et al. Deep Reinforcement Learning: A Survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *35*, 5064–5078.

7. Wang, N.; Chen, Y.; Zhang, D. A Comprehensive Review of Physics-Informed Deep Learning and Its Applications in Geoenergy Development. *Innov. Energy* **2025**, 2, 100087.
8. Kirk, D.E. *Optimal Control Theory: An Introduction*; Courier Corporation: North Chelmsford, MA, USA, 2004.
9. Ba ş ar, T.; Olsder, G.J. *Dynamic Noncooperative Game Theory*; SIAM: Philadelphia, PA, USA 1998.
10. Lasry, J.-M.; Lions, P.-L. Mean Field Games. *Jpn. J. Math.* **2007**, *2*, 229–260.
11. Carmona, R.; Delarue, F. *Probabilistic Theory of Mean Field Games with Applications I–II*; Springer: Berlin/Heidelberg, Germany, 2018; Volume 3.
12. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444.
13. Mienye, I.D.; Swart, T.G. A Comprehensive Review of Deep Learning: Architectures, Recent Advances, and Applications. *Information* **2024**, *15*, 755.
14. Deng, L.; Liu, Y. *Deep Learning in Natural Language Processing*; Springer: Berlin/Heidelberg, Germany, 2018.
15. Li, F.; Zhang, H.; Zhang, Y.-F.; et al. Vision-Language Intelligence: Tasks, Representation Learning, and Large Models. *arXiv* **2022**, arXiv:2203.01922.
16. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 1998; Volume 1.
17. Watkins, C.J.C.H.; Dayan, P. Q-Learning. *Mach. Learn.* **1992**, *8*, 279–292.
18. Sutton, R.S.; McAllester, D.; Singh, S.; et al. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Adv. Neural Inf. Process. Syst.* **1999**, *12*, 1057 - 1063.
19. Konda, V.; Tsitsiklis, J. Actor-Critic Algorithms. *Adv. Neural Inf. Process. Syst.* **1999**, *12*.
20. Vamvoudakis, K.G.; Lewis, F.L. Online Actor–Critic Algorithm to Solve the Continuous-Time Infinite Horizon Optimal Control Problem. *Automatica* **2010**, *46*, 878–888.
21. Karniadakis, G.E.; Kevrekidis, I.G.; Lu, L.; et al. Physics-Informed Machine Learning. *Nat. Rev. Phys.* **2021**, *3*, 422–440.
22. Simpkins, A. System Identification: Theory for the User, (ljung, l.; 1999)[on the Shelf]. *IEEE Robot. Autom. Mag.* **2012**, *19*, 95–96.
23. Bellman, R. Dynamic Programming. *Science* **1966**, *153*, 34–37.
24. Heydari, A. Optimal Impulsive Control Using Adaptive Dynamic Programming and Its Application in Spacecraft Rendezvous. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4544–4552.
25. Zhou, M.; Han, J.; Lu, J. Actor-Critic Method for High Dimensional Static Hamilton–Jacobi–Bellman Partial Differential Equations Based on Neural Networks. *Siam J. Sci. Comput.* **2021**, *43*, A4043–A4066.
26. Modares, H.; Lewis, F.L.; Naghibi-Sistani, M.-B. Integral Reinforcement Learning and Experience Replay for Adaptive Optimal Control of Partially-Unknown Constrained-Input Continuous-Time Systems. *Automatica* **2014**, *50*, 193–202.
27. Xue, S.; Luo, B.; Liu, D. Integral Reinforcement Learning Based Event-Triggered Control with Input Saturation. *Neural Netw.* **2020**, *131*, 144–153.
28. Mu, C.; Wang, K.; Qiu, T. Dynamic Event-Triggering Neural Learning Control for Partially Unknown Nonlinear Systems. *IEEE Trans. Cybern.* **2020**, *52*, 2200–2213.
29. Li, Y.; Xu, Y.; Li, K. Dynamic Event-Triggered Optimal Control for Heterogeneous Vehicle Platoon Based on Integral Reinforcement Learning. *IEEE Trans. Netw. Sci. Eng.* **2025**, *12*, 1885 - 1897.
30. Liang, Z.; Gu, Y.; Li, P.; et al. Prescribed-Time Formation Tracking in Multi-Agent Systems via Reinforcement Learning-Based Hybrid Impulsive Control with Time Delays. *Expert Syst. Appl.* **2025**, *272*, 126723.
31. Ren, H.; Gui, H.; Zhong, R. Efficient Fuel-Optimal Multi-Impulse Orbital Transfer via Contrastive Pre-Trained Reinforcement Learning. *Adv. Space Res.* **2025**, *75*, 7377–7396.
32. Nasir, Y.; Durlofsky, L.J. Deep Reinforcement Learning for Optimal Well Control in Subsurface Systems with Uncertain Geology. *J. Comput. Phys.* **2023**, *477*, 111945.
33. Wang, S.; Blanchet, J.; Glynn, P.W. An Efficient High-Dimensional Gradient Estimator for Stochastic Differential Equations. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 88045–88090.
34. Dixit, A.; Elsheikh, A. A Multilevel Reinforcement Learning Framework for PDE-Based Control. *arXiv* **2022**, arXiv:2210.08400.
35. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *J. Comput. Phys.* **2019**, *378*, 686–707.
36. García-Cervera, C.J.; Kessler, M.; Periago, F. Control of Partial Differential Equations via Physics-Informed Neural Networks. *J. Optim. Theory Appl.* **2023**, *196*, 391–414.
37. Furfaro, R.; D'Ambrosio, A.; Schiassi, E.; et al. Physics-Informed Neural Networks for Closed-Loop Guidance and Control in Aerospace Systems. In Proceedings of the AIAA SCITECH 2022 Forum, San Diego, CA, USA, 3–7 January 2022; p. 0361.
38. Tayal, M.; Singh, A.; Kolathaya, S.; et al. A Physics-Informed Machine Learning Framework for Safe and Optimal Control of Autonomous Systems. *arXiv* **2025**, arXiv:2502.11057.
39. Mowlavi, S.; Nabi, S. Optimal Control of PDEs Using Physics-Informed Neural Networks. *J. Comput. Phys.* **2023**, *473*, 111731.
40. Antonelo, E.A.; Camponogara, E.; Seman, L.O.; et al. Physics-Informed Neural Nets for Control of Dynamical Systems. *Neurocomputing* **2024**, *579*, 127419.
41. Demo, N.; Strazzullo, M.; Rozza, G. An Extended Physics Informed Neural Network for Preliminary Analysis of Parametric

Optimal Control Problems. *Comput. Math. Appl.* **2023**, *143*, 383–396.

42. Jiao, Z.; Luo, X.; Yi, X. Neural Optimal Controller for Stochastic Systems via Pathwise HJB Operator. *arXiv* **2024**, arXiv:2402.15592.
43. M ”u ller, J.; Zeinhofer, M. Position: Optimization in SciML Should Employ the Function Space Geometry. *arXiv* **2024**, arXiv:2402.07318.
44. Song, Y.; Yuan, X.; Yue, H. The ADMM-PINNs Algorithmic Framework for Nonsmooth PDE-Constrained Optimization: A Deep Learning Approach. *Siam J. Sci. Comput.* **2024**, *46*, C659–C687.
45. Hao, Z.; Ying, C.; Su, H.; et al. Bi-Level Physics-Informed Neural Networks for PDE Constrained Optimization Using Broyden's Hypergradients. *arXiv* **2022**, arXiv:2209.07075.
46. Wang, Y.; Wu, Z. Physics-Informed Reinforcement Learning for Optimal Control of Nonlinear Systems. *AIChE J.* **2024**, *70*, e18542.
47. Kamtue, K.; Moura, J.M.F.; Sangpetch, O. Solving Functional Optimization with Deep Networks and Variational Principles. *arXiv* **2024**, arXiv:2410.06277.
48. Wang, S.; Bhouri, M.A.; Perdikaris, P. Fast PDE-Constrained Optimization via Self-Supervised Operator Learning. *arXiv* **2021**, arXiv:2110.13297, .
49. Shukla, K.; Oommen, V.; Peyvan, A.; et al. Deep Neural Operators as Accurate Surrogates for Shape Optimization. *Eng. Appl. Artif. Intell.* **2024**, *129*, 107615.
50. Song, Y.; Yuan, X.; Yue, H. Accelerated Primal-Dual Methods with Enlarged Step Sizes and Operator Learning for Nonsmooth Optimal Control Problems. *arXiv* **2023**, arXiv:2307.00296.
51. Hwang, R.; Lee, J.Y.; Shin, J.Y.; et al. Solving PDE-Constrained Control Problems Using Operator Learning. *Proc. AAAI Conf. Artif. Intell.* **2022**, *36*, 4504–4512.
52. Chen, C.; Wu, J.-L. Neural Dynamical Operator: Continuous Spatial-Temporal Model with Gradient-Based and Derivative-Free Optimization Methods. *J. Comput. Phys.* **2025**, *520*, 113480.
53. Négiar, G.; Mahoney, M.W.; Krishnapriyan, A.S. Learning Differentiable Solvers for Systems with Hard Constraints. *arXiv* **2022**, arXiv:2207.08675.
54. Cheng, Z.; Li, Z.; Wang, X.; et al. Accelerating PDE-Constrained Optimization by the Derivative of Neural Operators. *arXiv* **2025**, arXiv:2506.13120.
55. Luo, D.; O'Leary-Roseberry, T.; Chen, P.; et al. Efficient PDE-Constrained Optimization under High-Dimensional Uncertainty Using Derivative-Informed Neural Operators. *Siam J. Sci. Comput.* **2025**, *47*, C899–C931.
56. Go, J.; Chen, P. Accurate, Scalable, and Efficient Bayesian Optimal Experimental Design with Derivative-Informed Neural Operators. *Comput. Methods Appl. Mech. Eng.* **2025**, *438*, 117845.
57. Su, H.; Zhang, H.; Sun, S.; et al. Integral Reinforcement Learning-Based Online Adaptive Event-Triggered Control for Non-Zero-Sum Games of Partially Unknown Nonlinear Systems. *Neurocomputing* **2020**, *377*, 243–255.
58. Wang, H.; Zhang, Y. Impulsive Maneuver Strategy for Multi-Agent Orbital Pursuit-Evasion Game under Sparse Rewards. *Aerosp. Sci. Technol.* **2024**, *155*, 109618.
59. Chen, F.; Martin, N.; Chen, P.-Y.; et al. Deciding Bank Interest Rates—A Major-Minor Impulse Control Mean-Field Game Perspective. *arXiv* **2024**, arXiv:2411.14481.
60. Ruthotto, L.; Osher, S.J.; Li, W.; et al. A Machine Learning Framework for Solving High-Dimensional Mean Field Game and Mean Field Control Problems. *Proc. Natl. Acad. Sci. USA* **2020**, *117*, 9183–9193.
61. Zhang, L.; Ghimire, M.; Zhang, W.; et al. Value Approximation for Two-Player General-Sum Differential Games with State Constraints. *IEEE Trans. Robot.* **2024**, *40*, 4631–4649.
62. Chen, X.; Liu, S.; Di, X. A Hybrid Framework of Reinforcement Learning and Physics-Informed Deep Learning for Spatiotemporal Mean Field Games. In Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems, London, UK, 29 May–2 June 2023.
63. Yin, W.; Shen, Z.; Meng, P.; et al. An Online Interactive Physics-Informed Adversarial Network for Solving Mean Field Games. *Eng. Anal. Bound. Elem.* **2024**, *169*, 106002.
64. Zhang, L.; Ghimire, M.; Xu, Z.; et al. Pontryagin Neural Operator for Solving General-Sum Differential Games with Parametric State Constraints. In Proceedings of the 6th Annual Learning for Dynamics & Control Conference, Oxford, UK, 15–17 July 2024; pp. 1728–1740.
65. Liu, S.; Chen, X.; Di, X. Scalable Learning for Spatiotemporal Mean Field Games Using Physics-Informed Neural Operator. *Mathematics* **2024**, *12*, 803.
66. Zeng, R.; Li, C.; Li, L.; et al. Solving Mean Field Game Based on Physics-Informed Operator Learning and Deep Reinforcement Learning. *J. Comput. Phys.* **2025**, *545*, 114457.
67. Li, C.; Wang, W. Optimal Impulse Control and Impulse Game for Continuous-Time Deterministic Systems: A Review. *Artif. Intell. Sci. Eng.* **2025**, *1*, 208–219.