Scilight

Journal of Artificial Intelligence for Automation
https://www.sciltp.com/journals/jaia

Review

A Survey on Neural Dynamics for Computing and Control: Theories, Models, and Applications

Long Jin

School of Information Science and Engineering, Lanzhou University, Lanzhou 730000, China; jinlongsysu@foxmail.com

How To Cite: Jin, L. A Survey on Neural Dynamics for Computing and Control: Theories, Models, and Applications. *Journal of Artificial Intelligence for Automation* **2026**, *1*(1), 1.

Received: 16 November 2025
Revised: 25 December 2025
Accepted: 6 January 2026
Published: 13 January 2026

Abstract: Neural dynamics provides a powerful and unifying tool for understanding systems that learn, adapt, and interact. This survey provides a comprehensive overview of the theories, models, and applications of neural dynamics at the intersection of computing and control. We first articulate the core concept of neural dynamics, explaining the close connection between this concept and dynamical system theory. We then demonstrate the broad applicability of neural dynamics by reviewing a wide range of models across various key domains. In the field of control, we survey neural dynamics approaches to classical problems of stability and optimality, especially within control systems and multi-agent systems (MASs). In the field of computing, we focus on deep learning, analyzing both model architectures and optimizers as different dynamical systems. The principal contribution of this work is to bridge these domains, revealing the computation and control topics governed by neural dynamics theories. This integrated viewpoint illuminates numerous applications and inspires future research directions focusing on advanced models in terms of computation and control.

Keywords: control systems; deep learning; dynamical systems; multi-agent systems; neural dynamics

1. Introduction

With the rapid advancement of artificial intelligence (AI), modern intelligent systems, such as automatic control systems [1–3], embodied intelligence systems [4–6], multi-agent systems (MASs) [7–9], and deep learning models [10], increasingly exhibit a deep and inextricable integration of computation and control. On one hand, these systems must perform complex computations to process vast amounts of information. On the other hand, they must interact with the physical world through precise control to maintain stability and accomplish tasks. At the theoretical level, however, the disciplinary foundations for these domains—computational science, centered on deep learning, and control science, centered on control systems—have evolved mainly along separate trajectories. This theoretical divide not only hinders a unified understanding of complex intelligent systems but also poses a fundamental challenge to designing the next generation of high-performance systems.

To bridge this theoretical divide, we propose neural dynamics as a unifying analytical language and theoretical framework. As an interdisciplinary field studying the temporal evolution of neural states using mathematical tools from dynamical system theory [11,12], neural dynamics is intrinsically rooted in neuroscience [13]. However, the idea of utilizing dynamical system theory to analyze complex systems extends far beyond its biological origins. The dynamical system theory offers a powerful toolkit, including stability analysis [14], attractor theory [15], and bifurcation theory [16], to describe and predict the behavior of time-evolving systems. This makes neural dynamics an ideal bridge to connect knowledge across the disparate fields of computation and control.

As shown in Figure 1, this survey systematically reviews the theoretical foundations, core models, and wide-ranging applications of neural dynamics at the intersection of computation and control. In the field of computation, we focus on deep learning, analyzing how both the forward propagation in model architectures and the parameter update in optimization algorithms can be characterized as distinct dynamical systems. In the field of control, we explore how neural dynamics models address classical control problems of stability and optimality, as well as their

application in modeling distributed coordination and emergent collective behavior in MASs. Through this comprehensive review, the principal contribution of this work is to unveil the dynamical mechanisms and underlying principles governing both computation and control. We argue that this integrative perspective on neural dynamics not only offers profound insights into macroscopic properties of existing systems, such as generalization and robustness, but more importantly, it illuminates a path toward building future hybrid intelligent systems that seamlessly integrate perception, learning, reasoning, and decision-making, providing a solid theoretical foundation for this endeavor.

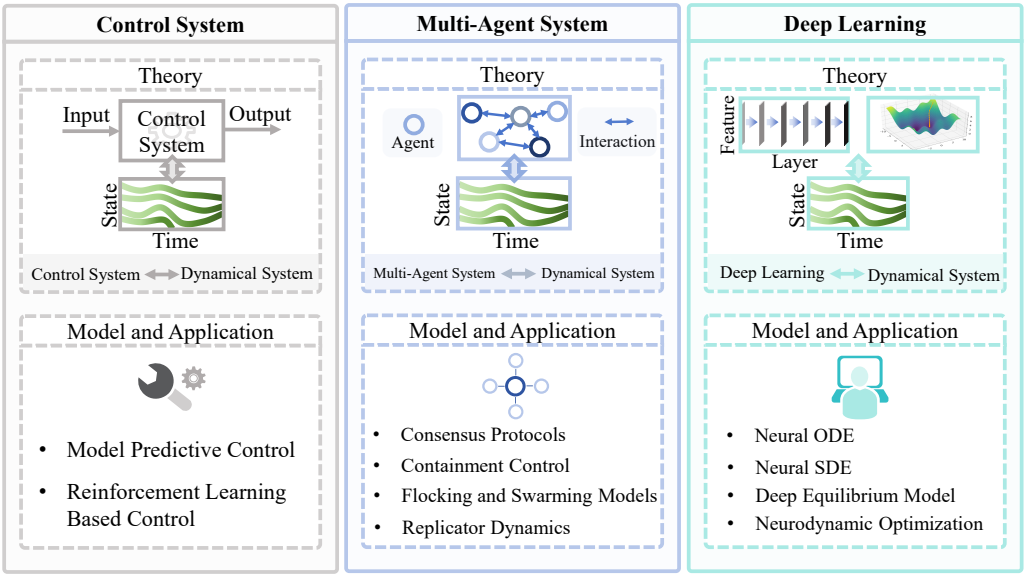


Figure 1. Framework of this paper.

2. Neural Dynamics

Neural dynamics is fundamentally grounded in dynamical system theory, which provides a mathematical framework to describe how the state of a system evolves over time [11,12]. This section introduces the key concepts and tools from dynamical system theory that are essential for understanding neural dynamics.

2.1. Neural Dynamics and Dynamical Systems

Neural dynamics is an interdisciplinary theoretical framework dedicated to investigating the dynamical systems that govern neural activity [13]. It integrates principles from neuroscience, mathematics, physics, and computer science to understand how the states of a neural system evolve over time and how this temporal evolution gives rise to complex behaviors. The early research on neurodynamics can be traced back to the Hopfield network [17] in the 1980s. In recent years, this framework has been increasingly applied beyond its biological origins to investigate problems at the intersection of control and computation, yielding promising theoretical insights [18,19].

Despite its biological origin, in the context of this paper, neural dynamics can be viewed as a specialized branch of dynamical system theory. A dynamical system is formally described by a state space and a rule that dictates the evolution of its state variables [20]. The power of this mathematical abstraction lies in its universality: The state variables and the evolution rule are defined by the specific properties of the system under investigation. This adaptability allows the theoretical tools from dynamical systems to serve as a unifying framework for computation and control. Specifically, in control theory, the state evolution of a control system under feedback can be naturally modeled as a dynamical system [21], where the objective is to guide the system towards a stable equilibrium or a desired trajectory. In MAS, the emergent collective behavior of a group is described as the evolution of the agents' combined state, governed by local interaction rules and communication topologies [22]. In deep learning, a dual dynamical system perspective can be adopted. The forward propagation of hidden states through a deep neural network can be modeled as a dynamical system that maps input data to an output through a sequence of non-linear transformations [23]. Concurrently, the learning process itself, driven by an optimization algorithm like gradient descent, can be modeled as another dynamical system evolving in the high-dimensional parameter space [18].

Neural dynamics frames these seemingly disparate problems in a common language of dynamical systems, allowing us to leverage a powerful analytical toolkit to analyze their stability, convergence, and robustness, thereby revealing the fundamental principles that connect them.

2.2. Differential Equations and Dynamical Systems

The relationship between differential equations and dynamical systems is foundational and deeply connected [24]. They represent two complementary perspectives for analyzing the evolution of a system over time. A differential equation provides a local description of change. Specifically, an ordinary differential equation (ODE) of the form $\mathrm{d}x/\mathrm{d}t = f(x,t)$ defines the instantaneous rate of change of a system's state at any given time t [25]. It is the mathematical expression of the underlying mechanism governing the system from one moment to the next. On the other hand, the theory of a dynamical system offers a global, geometric framework [26]. It encompasses not only the evolution rule given by the differential equation but also the entire state space in which the system's evolution unfolds. The primary object of study in dynamical system theory is the qualitative behavior of the system's trajectories—the paths traced by the state $x(t)$ as it evolves from various initial conditions. Therefore, a differential equation can be understood as the infinitesimal generator of the dynamics, defining the vector field that directs the flow within the state space. The solution to an initial value problem for the differential equation corresponds to a single, unique trajectory within this global state space. This dual perspective allows us to translate the specific, mechanistic rules encoded in a differential equation into a macroscopic, qualitative understanding of system-level behaviors such as stability, periodicity, and chaos [27,28]. For instance, in deep learning, the models known as neural ODE [29] explicitly define the forward pass as the solution to a differential equation whose dynamics are parameterized by a neural network. Analyzing this system from a dynamical system perspective allows researchers to investigate properties like invertibility and computational efficiency by studying the stability and flow of its learned vector field [30,31].

3. Application of Neural Dynamics in Control Systems

In this section, we explore the application of neural dynamics in control systems, focusing on stability analysis and optimal control. Neural dynamics provides a framework for modeling and analyzing complex control systems, enabling the design of robust and adaptive controllers.

3.1. Theoretical Basis: Modeling Control Systems as Dynamical Systems

Control theory is an interdisciplinary field of mathematics, physics, and computer science that deals with the analysis of control systems [32]. As shown in Figure 2, control systems can be broadly classified into two categories: Open-loop and closed-loop systems [33–35]. Open-loop systems do not use feedback to control the output [36]. The input is predetermined, and the system operates based only on this input without monitoring the output. While closed-loop systems use feedback to control the output [37]. The output is monitored, and the input is adjusted accordingly to achieve the desired output. In both types of systems, the state of the system can be represented as a vector in a high-dimensional space, and the evolution of the state over time can be described by a set of differential equations [38]. Therefore, control systems can be effectively modeled as dynamical systems, which aligns with the core idea of neural dynamics. In 1993, Colonius et al. [21] formally models control systems as dynamical systems, providing a theoretical foundation for applying neural dynamics to control theory. It argues that control systems are projections of specific dynamical systems. Specifically, consider a control-affine system of the form:

$$\dot{\boldsymbol{x}}(t) = X_0(\boldsymbol{x}(t)) + \sum_{i=1}^{m} u_i(t) X_i(\boldsymbol{x}(t)), \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the state vector which evolves on a smooth manifold M; $\dot{\boldsymbol{x}}(t)$ is time derivative of the state which represents the velocity vector of the state's trajectory in the manifold M, indicating the instantaneous direction and rate of change; $\boldsymbol{u}(t) = [u_1(t), u_2(t), \ldots, u_m(t)]$ is the control input, a time-varying vector function that can be manipulated to steer the system's behavior, with each component $u_i(t)$ being a scalar control function; $U \in \mathbb{R}^m$ is the control space, a compact set from which the instantaneous control vectors $\boldsymbol{u}(t)$ must be chosen; $X_0, X_1, \ldots, X_m$ are vector fields on $\mathbb{R}^n$, with each vector field X_i determines how the corresponding scalar control input $u_i(t)$ influences the system's dynamics at state $\boldsymbol{x}(t)$. The key insight is to associate this control system (1) with a corresponding dynamical system that evolves on an extended state space $U \times M$. The state of this new system at any time is a pair $(\boldsymbol{u}, \boldsymbol{x})$, consisting of both the entire control function and the system's physical state. The evolution of this pair over time t is defined as

$$\phi(t, \boldsymbol{u}, \boldsymbol{x}) = (\theta_t \boldsymbol{u}, \varphi(t, \boldsymbol{x}, \boldsymbol{u})), \tag{2}$$

where $\phi : \mathbb{R} \times U \times M \to U \times M$ is the flow map of the dynamical system which takes an initial state $(\boldsymbol{u}, \boldsymbol{x})$ and an evolution time t and returns the state to which the system has evolved after that time; θ_t is the time-shift operator which acts on a control function $\boldsymbol{u}(t_0)$ to produce a new function that is shifted in time. Specifically, $\theta_t \boldsymbol{u}(t_0) = \boldsymbol{u}(t_0 + t)$; $\varphi(t, \boldsymbol{x}, \boldsymbol{u})$ is the solution to the control system (1) at time t, starting from the initial state $\boldsymbol{x}$ and under the influence of the control function $\boldsymbol{u}$. This construction establishes a profound correspondence between the topological properties of the control system and the dynamical properties of ϕ. Specifically, it has been shown that a subset $D \subset M$ is a control set of the control system (1) if and only if there exists a compact invariant set $\mathcal{D} \subset U \times M$ of the dynamical system ϕ such that $D = \pi_M(\mathcal{D})$, where $\pi_M : U \times M \to M$ is the projection onto the second component. This result provides a rigorous mathematical foundation for analyzing control systems using the tools and techniques from dynamical system theory. In 1998, the study in [39] investigates the relation between discounted and average deterministic optimal control problems for non-linear control systems. It uses the mathematical tools from dynamical system theory to analyze the controllability properties of the system. In 2007, ref. [40] extends the classical Grobman-Hartman theorem to control systems, providing conditions under which a non-linear control system can be locally topologically conjugate to its linearization around a hyperbolic equilibrium point. This extension further solidifies the connection between control systems and dynamical systems, enabling the application of linearization techniques and stability analysis from dynamical systems theory to control problems. In 2021, ref. [41] introduces six types of equi-invariability, which are the analogies to equi-continuity, equi-continuity in the mean, and mean equi-continuity in topological dynamical systems. Then, it utilizes three versions of equi-invariability to characterize bounded invariance complexity, bounded invariance complexity in the mean, and mean L-stability for control systems, respectively. Moreover, it obtains two new dichotomy theorems for a control set with dense interior. Based on dynamical system theory, in 2025, ref. [42] establishes Bowen's equations for the upper capacity invariance pressure and Pesin-Pitskel invariance pressure of discrete-time control systems. It introduces a new invariance pressure, called induced invariance pressure on partitions, that specializes the upper capacity invariance pressure on partitions and then shows that the two types of invariance pressures are related by Bowen's equation. In summary, these works demonstrate the theoretical basis for applying neural dynamics to control theory by modeling control systems as dynamical systems. This modeling approach enables the use of dynamical system theory to analyze and design control systems, providing a powerful framework for addressing complex control problems, such as stability analysis and optimal control.

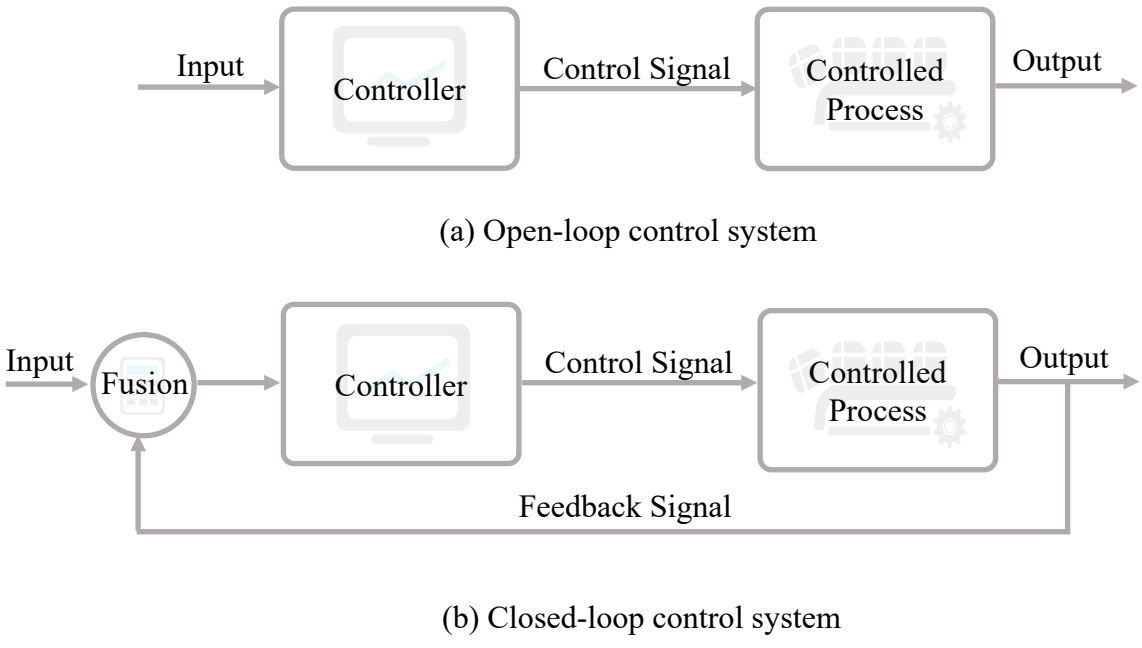


Figure 2. Diagram of control systems.

While the aforementioned theoretical foundations are primarily established in the continuous-time domain, practical implementations on modern digital hardware require discretization. The translation from continuous neural dynamics to digital controllers typically involves numerical integration methods, such as Euler or Runge-Kutta schemes. It is crucial to note that the stability guarantees derived for the continuous system hold for the discrete implementation provided that the sampling rate is sufficiently high to capture the system's dynamics. If the sampling interval is too large relative to the system's time constants, discretization errors may accumulate, potentially leading

to numerical instability or aliasing effects that compromise the theoretical convergence properties.

3.2. Core Models and Their Applications in Control Theory

Building upon the theoretical foundation that treats control systems as dynamical systems, the central task becomes identifying the specific mathematical form of the governing equations [43–45], typically expressed as $\dot{x} = f(x, u)$. The methods for defining this function f have evolved significantly, transitioning from classical, first-principle models like model predictive control (MPC) [46–48] to modern, data-driven paradigms such as reinforcement learning (RL) [49–51], which utilizes neural network architectures. This section outlines these core modeling paradigms and their primary applications in control theory.

MPC: MPC [46–48] is a widely used control strategy that relies on an explicit model of the system to predict future behavior and optimize control actions. In MPC, the control problem is formulated as an optimization problem that minimizes a cost function over a finite prediction horizon, subject to the system dynamics and constraints. The system dynamics are typically represented by a set of differential equations, which can be derived from first principles or identified through data analysis. The optimization problem is solved at each time step, and the first control action is applied to the system. This process is repeated in a receding horizon fashion, allowing the controller to adapt to changes in the system and environment. Specifically, the operation of MPC can be summarized by three fundamental principles:

(1) *Prediction:* At each time step k, given the current measured state x_k, a model of the system dynamics, typically in the form $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$, is used to predict the future state trajectory over a predefined prediction horizon N. This prediction is contingent upon a candidate sequence of future control inputs, $U_k = \{\boldsymbol{u}_k, \boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_{k+N-1}\}$.

(2) *Optimization:* An optimal control problem is solved online to find the control sequence U_k that minimizes a cost function J. This cost function typically penalizes deviations from a desired reference trajectory and the magnitude of control effort. The problem is formally stated as:

$$\begin{aligned} \min_{U_k} J(\boldsymbol{x}_k, U_k) = & \sum_{i=0}^{N-1} \left(\|\boldsymbol{x}_{k+i|k} - \boldsymbol{x}_{\text{ref},k+i}\|_Q^2 + \|\boldsymbol{u}_{k+i}\|_R^2 \right) \\ & + \|\boldsymbol{x}_{k+N|k} - \boldsymbol{x}_{\text{ref},k+N}\|_P^2, \\ \text{s.t.} \quad & \boldsymbol{x}^- \leq \boldsymbol{x}_{k+i|k} \leq \boldsymbol{x}^+, \\ & \boldsymbol{u}^- \leq \boldsymbol{u}_{k+i} \leq \boldsymbol{u}^+, \end{aligned} \tag{3}$$

where the matrices Q, R, and P are positive semi-definite weighting matrices, and $||\boldsymbol{x}||_Q^2 = \boldsymbol{x}^T Q \boldsymbol{x}$; $\boldsymbol{x}_{k+i|k}$ is the predicted system state for the future time step $k+i$ at the current time step k; $\boldsymbol{x}_{\text{ref},k+i}$ is the reference state we set for the system at the future time step $k+i$. This optimization is performed subject to constraints on system dynamics, states, and control inputs.

(3) *Receding horizon implementation:* Upon solving the optimization problem, an optimal control sequence $U_k^* = \{\boldsymbol{u}_{k|k}^*, \boldsymbol{u}_{k+1|k}^*, \ldots, \boldsymbol{u}_{k+N-1|k}^*\}$ is obtained. However, only the first element of this sequence, $\boldsymbol{u}_{k|k}^*$, is applied to the system. The remainder of the sequence is discarded. At the next time step, $k+1$, a new state measurement is taken, and the entire process—prediction and optimization—is repeated to compute a new optimal control sequence.

This receding horizon strategy is the core characteristic of MPC. By continuously re-evaluating the optimal control plan based on the most recent state information, it establishes a powerful feedback mechanism. This allows the controller to inherently handle system constraints and effectively compensate for external disturbances and model-plant mismatch, resulting in a robust and high-performance closed-loop control system. MPC is successfully applied in various domains, including process control [52–54] and robotics [55–57]. In terms of robust control for robotics, ref. [58] proposes a modified primal-dual neural network designed for the motion control of redundant manipulators. By incorporating a dynamic noise-rejection mechanism, this approach effectively suppresses harmonic noises during operation, ensuring precise trajectory tracking even in the presence of periodic disturbances. Complementing this, Cao et al. [59] address the challenges of underactuated systems by presenting a robust neuro-optimal control strategy for snake robots, specifically employing experience replay to enhance adaptability and robustness. In terms of process control, ref. [52] proposes a distributed economic model predictive control (EMPC) framework for a non-linear chemical process network. The proposed method is based on a sequential distributed MPC design and guarantees closed-loop stability. The effectiveness of the proposed method is demonstrated through extensive simulations on a catalytic alkylation of benzene process network. Moreover, literature [53] presents a process control framework for deep drawing based on MPC. The proposed framework represents the deep drawing process

as a single-input multiple-output model that relates the blank holder force to the draw-in of n different critical points around the die, allowing for the avoidance of workpiece defects caused by the abnormal sliding of the metal sheet during the forming phase. Additionally, Huang et al. [54] proposes a long short-term memory (LSTM) neural network-based MPC approach for process control. The LSTM is applied to predict the behaviors of the controlled process, which can automatically match different operation modes without requiring a switching strategy. Then, combined with the MPC framework, an adaptive gradient descent method is introduced to handle the optimization problem and its constraints. In terms of robotics, literature [60] presents a data-driven neural dynamics-based MPC algorithm, which consists of an MPC scheme, a neural dynamics solver, and a discrete-time Jacobian matrix updating law. With the help of the updating law, the future output of the model-unknown redundant manipulator is predicted, and the MPC scheme for trajectory tracking is constructed. The neural dynamics solver is designed to solve the MPC scheme to generate control input driving the redundant manipulator. Moreover, ref.[56] designs a varying-parameter complementary neural network and combines it with MPC to solve the multi-robot tracking and formation problems via a leader-follower strategy. Additionally, ref. [57] presents a novel snap-layer minimum motion scheme, otherwise known as the minimum motion planning and control scheme for redundant robot arms, to obtain smoother kinematic control of minimum motion. The proposed scheme is based on the neural dynamics equivalency and solver, which can be used to solve the multi-layer physical limits of redundant robot arms.

RL-based control: Another core model in control theory is RL [49–51], which is a data-driven approach that learns optimal control strategies through interaction with the environment. In RL, an agent learns to make decisions by taking actions in an environment to maximize a cumulative reward signal. The behavior of the agent is typically modeled as a Markov decision process (MDP) [61,62], where the state of the system evolves according to a transition function that depends on the current state and action taken by the agent. To tighten the connection with dynamical system theory, it is crucial to recognize that this state transition probability, denoted as $P(s'|s,a)$, serves as the stochastic, discrete-time equivalent of the differential equation $\dot{x} = f(x,u)$ used in classical control. This mapping reveals that RL and MPC are mathematically isomorphic in their objectives: Both aim to optimize a system's trajectory. The primary distinction lies in their underlying dynamics—MPC typically assumes a deterministic, continuous evolution, whereas RL operates within a stochastic, discrete framework. The agent's goal is to learn a policy that maps states to actions in a way that maximizes the expected cumulative reward over time. RL algorithms can be broadly classified into two categories: Model-based and model-free methods [63]. Model-based methods learn a model of the system dynamics and use it to plan optimal actions, while model-free methods directly learn a strategy or value function from experience without explicitly modeling the dynamics. RL has been successfully applied to various control problems, including robotics [64,65] and autonomous driving [8,66]. In terms of robotics, literature [64] presents a reinforcement learning approach for acquiring motor skills in a robotic system. The proposed method utilizes a stochastic real-valued reinforcement learning algorithm to represent the policy, which learns the optimal policy through trial-and-error interactions with the environment. The effectiveness of the proposed method is evaluated on two classic tasks with a robotic arm. Moreover, Raffin et al. [65] address the poor exploration issues in RL by adapting state-dependent exploration to deep RL algorithms. It presents two extensions to the original state-dependent exploration, utilizing more general features and resampling the noises periodically, which leads to a new exploration method: generalized state-dependent exploration. In terms of autonomous driving, Sallab et al. [66] first present a deep reinforcement learning approach for autonomous driving in complex urban environments. The presented method incorporates recurrent neural networks [67,68] for information integration, enabling the car to handle partially observable scenarios. It also integrates the attention models to focus on relevant information, thereby reducing the computational complexity for deployment on embedded hardware. The framework was tested in an open-source 3D car racing simulator. Additionally, ref. [8] presents a decentralized solution based on the attention mechanism and recurrent neural networks, utilizing a multi-agent distributed deep deterministic policy gradient, to solve the real-time task offloading and heterogeneous resource allocation problem in vehicular edge computing systems. The proposed method enables each vehicle to make optimal offloading decisions and allocate computational resources based on its own observations and the historical information of other vehicles.

4. Application of Neural Dynamics in Multi-Agent Systems

MASs consist of multiple interacting agents that work together to achieve a common goal or perform complex tasks [69–71]. The study of MAS gains significant attention in recent years due to its wide range of applications, including robotics [72,73], sensor networks [74], and social systems [75]. Neural dynamics provides a powerful framework for modeling and analyzing the behavior of MAS, enabling the design of robust and adaptive coordination strategies. In this section, we first introduce the theoretical basis for applying neural dynamics to MAS, and then review the core models and their applications in MAS.

4.1. Theoretical Basis: Modeling Multi-Agent Systems as Dynamical Systems

As illustrated in Figure 3, MASs can be characterized by a variety of organizational structures. An organizational structure defines the fundamental relationships between agents, governing their roles, privileges, patterns of information flow, and coordination protocols [76]. These structures typically fall into several key paradigms, including centralized, decentralized, and hierarchical, with each paradigm determining the system's underlying communication topology among the agents. MASs with different organizational structures can be effectively modeled as different kinds of dynamical systems, where the state of the system is represented by the states of all agents, and the evolution of the state over time is governed by a set of differential equations. This modeling provides a promising framework for analyzing the decentralized intelligence and emergent collective behaviors of MAS, especially in large-scale applications such as smart grids [76], autonomous vehicle fleets [77], and sensor swarms [78]. In these systems, coordination must be achieved with limited global information and under stringent real-time constraints [79]. The framework enables each agent to evolve its state based on local observations and interactions, thereby achieving system-level objectives through local computation. The interactions between agents can be captured by coupling terms in the differential equations, which model the influence of neighboring agents on each agent's state. This approach allows for the analysis of stability, convergence, and robustness of the MAS using tools from dynamical system theory. The work in [80] presents novel distributed near-optimal consensus protocols for input-constrained double-integrator MASs by modeling one-dimensional MASs as dynamical systems. Specifically, consider an MAS consisting of N agents with a scalar state x_i. The communication topology of the agents can be described by an undirected connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ is the set of nodes representing the agents, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges representing the communication links between agents. The degree of a node v_i, i.e., the number of nodes connecting with the node v_i, is denoted by $\deg(v_i)$. The adjacency matrix of the graph $\mathcal{G}$ is denoted by $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{N\times N}$, where $a_{ij} = 1$ if v_i and v_j is connected. The degree matrix of the graph is denoted by $\mathcal{D} = \mathrm{diag}(d_i) \in \mathbb{R}^{N\times N}$, where $d_i = \deg(v_i)$. The Laplacian matrix of the undirected connected graph is defined as $L = \mathcal{D} - \mathcal{A}$. Building on this, the dynamics of each agent is given by:

$$\begin{aligned} \frac{\mathrm{d}x_i(t)}{\mathrm{d}t} &= v_i(t), \\ \frac{\mathrm{d}v_i(t)}{\mathrm{d}t} &= u_i(t), \end{aligned} \tag{4}$$

where t is the time variable, $x_i \in \mathbb{R}^n$ is the position of agent i, $v_i \in \mathbb{R}^n$ is the velocity of agent i, and $u_i \in \mathbb{R}^n$ is the control input of agent i. The objective of the MAS is to design the control inputs u_i such that the agents achieve a desired collective behavior, such as the state consensus with a common state $x^* \in \mathbb{R}^n$ and a common velocity $v^* \in \mathbb{R}^n$, i.e., $\lim_{t\to\infty} x_i(t) = x^*$ and $\lim_{t\to\infty} v_i(t) = v^*$ for all $i = 1, 2, \ldots, N$. The agents are with limited actuation capabilities, modeled by the constraint $u_i^- \le u_i \le u_i^+$, where u_i^- and u_i^+ are the lower and upper bounds of the input of the i-th agent, respectively. The collective dynamics of the MAS can be written in a compact form as

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} &= \boldsymbol{v}(t), \\ \frac{\mathrm{d}\boldsymbol{v}(t)}{\mathrm{d}t} &= \boldsymbol{u}(t), \end{aligned} \tag{5}$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^N$, $\boldsymbol{v} = [v_1, v_2, \ldots, v_N] \in \mathbb{R}^N$, and $\boldsymbol{u} = [u_1, u_2, \ldots, u_N] \in \mathbb{R}^N$. The input constraints of the agents can be written as $\boldsymbol{u}^- \le \boldsymbol{u} \le \boldsymbol{u}^+$, where $\boldsymbol{u}^- = [u_1^-, u_2^-, \ldots, u_N^-]$ and $\boldsymbol{u}^+ = [u_1^+, u_2^+, \ldots, u_N^+]$. For achieving the consensus of the double-integrator multi-agent system, a receding-horizon performance index is defined as:

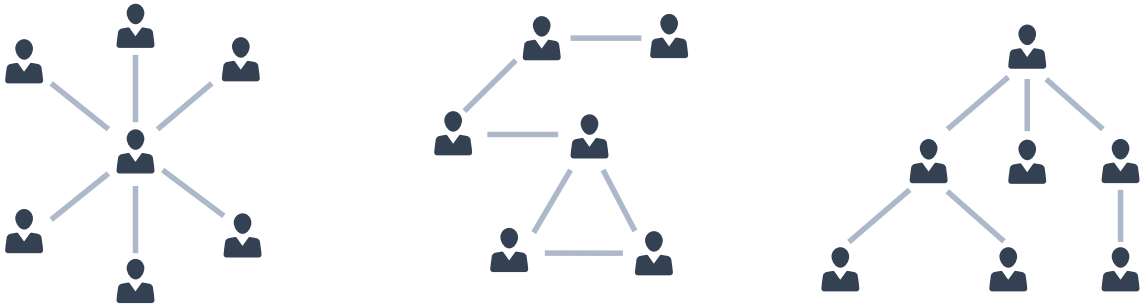
$$J(t) = \int_0^T \boldsymbol{x}(t+\tau)^\top L \boldsymbol{x}(t+\tau)\mathrm{d}\tau, \tag{6}$$

where $T > 0 \in \mathbb{R}$ is the predictive period, and L is the Laplacian matrix of the undirected connected communication graph of the MAS. The objective is to design the control input $\boldsymbol{u}(t)$ to minimize the performance index $J(t)$ at each time instant t with input constraints of the agents. This performance index ensures that the velocity of the agents is zero when consensus is achieved, which means that the position of the agents is eventually static. Therefore, the consensus problem is converted into an optimization problem as

$$\begin{aligned} &\min_{\boldsymbol{u}(t)} J(t), \\ \text{s.t.} \quad &\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{v}(t), \\ &\frac{\mathrm{d}\boldsymbol{v}(t)}{\mathrm{d}t} = \boldsymbol{u}(t), \\ &\boldsymbol{u}^- \leq \boldsymbol{u}(t) \leq \boldsymbol{u}^+, \end{aligned} \tag{7}$$

where $\boldsymbol{u}(t)$ is the control input of the MAS at time t. Applying the Taylor expansion to the system dynamics (5) yields:

$$\begin{aligned} \boldsymbol{x}(t+\Delta t) &\approx \boldsymbol{x}(t) + \Delta t \boldsymbol{v}(t) + \frac{\Delta t^2}{2}\ddot{\boldsymbol{x}}(t), \\ &= \boldsymbol{x}(t) + \Delta t \boldsymbol{v}(t) + \frac{\Delta t^2}{2}\boldsymbol{u}(t). \end{aligned} \tag{8}$$



(a) Centralized (b) Decentralized (c) Hierarchical

Figure 3. Examples of different MAS structures. (**a**) Centralized structure, (**b**) Decentralized structure, (**c**) Hierarchical structure.

This equation describes how the positions of all agents in the MAS evolve over a small time interval Δt based on their current positions, velocities, and control inputs. The term $\Delta t \boldsymbol{v}(t)$ represents the contribution of the current velocities to the change in position, while the term $(\Delta t^2/2)\boldsymbol{u}(t)$ captures the effect of the control inputs on the agents. Based on this, the optimization problem (7) can be relaxed as

$$\begin{aligned} J(t) \approx &\int_0^T \left(\boldsymbol{x}(t) + \tau \boldsymbol{v}(t) + \frac{\tau^2}{2}\boldsymbol{u}(t)\right)^\top L \\ &\left(\boldsymbol{x}(t) + \tau \boldsymbol{v}(t) + \frac{\tau^2}{2}\boldsymbol{u}(t)\right) \mathrm{d}\tau \\ = &\frac{T^5}{20}\boldsymbol{u}(t)^\top L \boldsymbol{u}(t) + \frac{T^3}{3}\boldsymbol{x}(t)^\top L \boldsymbol{u}(t) \\ &+ \frac{T^4}{4}\boldsymbol{v}(t)^\top L \boldsymbol{u}(t) + * \\ = &\hat{J}(t) + *, \end{aligned} \tag{9}$$

where $*$ is the sum of the terms that are independent of the control input $\boldsymbol{u}(t)$. Therefore, the optimization problem (7) can be relaxed as

$$\begin{aligned} &\min_{\boldsymbol{u}(t)} \hat{J}(t), \\ \text{s.t.} \quad &\boldsymbol{u}^- \leq \boldsymbol{u}(t) \leq \boldsymbol{u}^+, \end{aligned} \tag{10}$$

where $\hat{J}(t) = (T^5/20)\boldsymbol{u}(t)^\top L \boldsymbol{u}(t) + (T^3/3)\boldsymbol{x}(t)^\top L \boldsymbol{u}(t) + (T^4/4)\boldsymbol{v}(t)^\top L \boldsymbol{u}(t)$. Building on this, literature [80] obtain the optimal control input $\boldsymbol{u}^*(t)$ by modeling the optimization problem as a dynamical system:

$$\lambda \frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} = -\boldsymbol{u}(t) + P(\boldsymbol{u}(t) - \frac{\partial \hat{J}}{\partial \boldsymbol{u}(t)}), \tag{11}$$

where $\lambda > 0 \in \mathbb{R}$ is a constant used for scaling the convergence rate, $P(\cdot)$ is a projection operator that ensures the input constraints of the agents are satisfied, which can be defined as $P(x_i) = \min(\max(u_i^+, x_i), u_i^-)$, and

$\partial\hat{J}/\partial\boldsymbol{u}(t) = (T^5/10)L\boldsymbol{u}(t) + (T^3/3)L\boldsymbol{x}(t) + (T^4/4)L\boldsymbol{v}(t)$ is the gradient of the performance index $\hat{J}(t)$ with respect to the control input $\boldsymbol{u}(t)$. This dynamical system describes how the control input $\boldsymbol{u}(t)$ evolves over time to minimize the performance index $\hat{J}(t)$ while satisfying the input constraints of the agents. The term $-\boldsymbol{u}(t)$ represents a damping effect that prevents the control input from growing unbounded, while the term $P(\boldsymbol{u}(t) - \partial\hat{J}/\partial\boldsymbol{u}(t))$ represents a gradient descent step that drives the control input towards the optimal solution. The optimal control input $\boldsymbol{u}^*(t)$ is obtained when the system reaches equilibrium, i.e., $\mathrm{d}\boldsymbol{u}(t)/\mathrm{d}t = 0$. The proposed method guarantees that the MAS achieves consensus while satisfying the input constraints of the agents. Thereafter, ref. [81] demonstrates that the method presented in [80] can be extended to cases where the privacy protection of agents' initial positions is necessary. Different from the one-dimensional MAS studied in [80], the multi-dimensional MAS is a more general case, where the state of each agent is represented by a vector $\boldsymbol{x}_i \in \mathbb{R}^n$ instead of a scalar. For achieving the consensus of a multi-dimensional MAS, consider the following multi-objective optimization problem given in [82]:

$$\begin{aligned} &\min_{\boldsymbol{x}} \sum_{i=1}^{N} \omega_i f_i(\boldsymbol{x}_i), \\ &\text{s.t.} \quad \boldsymbol{L}\tilde{\boldsymbol{x}} = 0, \boldsymbol{x}_i \in \Gamma, \end{aligned} \tag{12}$$

where $\omega_i > 0 \in \mathbb{R}$ is a fixed weight coeffcient, $\tilde{\boldsymbol{x}} = \mathrm{col}\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\} \in \mathbb{R}^{n\times N}$, $\boldsymbol{L} = L \otimes I_n \in \mathbb{R}^{nN\times nN}$ with I_n being the identity matrix of dimension n, and $\Gamma = \{\boldsymbol{x} | A\boldsymbol{x} < b, g(\boldsymbol{x}) < 0, \boldsymbol{x} \in \Omega\}$, where Ω is a convex set. The objective of the problem (12) is to find the common state $\boldsymbol{x}^* \in \mathbb{R}^n$ such that the agents achieve a desired collective behavior, i.e., $\lim_{t\to\infty} \boldsymbol{x}_i(t) = \boldsymbol{x}^*$ for all $i = 1, 2, \ldots, N$. Building on this, ref. [82] solves the optimization problem (12) by modeling the consensus optimization problem as a dynamical system:

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{x}_i(t)}{\mathrm{d}t} &= -\boldsymbol{x}_i(t) + P(\boldsymbol{x}_i(t) - \omega_i \frac{\partial f_i(\boldsymbol{x}_i(t))}{\boldsymbol{x}_i(t)} \\ &\quad - \frac{\partial g(\boldsymbol{x}_i(t))}{\boldsymbol{x}_i(t)}\boldsymbol{y}_i(t) - A^\top \boldsymbol{z}_i(t)) \\ &\quad + \sum_{j=1}^{N} \omega_{ij}(\boldsymbol{x}_j(t) + \boldsymbol{\eta}_j(t) - \boldsymbol{x}_i(t) - \boldsymbol{\eta}_i(t)), \\ \frac{\mathrm{d}\boldsymbol{y}_i(t)}{\mathrm{d}t} &= -\boldsymbol{y}_i(t) + (\boldsymbol{y}_i(t) + g(\boldsymbol{x}_i))^+, \\ \frac{\mathrm{d}\boldsymbol{z}_i(t)}{\mathrm{d}t} &= A\boldsymbol{x}_i - \boldsymbol{b}, \\ \frac{\mathrm{d}\boldsymbol{\eta}_i(t)}{\mathrm{d}t} &= \sum_{j=1}^{N} \omega_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j), \end{aligned} \tag{13}$$

where $(\cdot)^+ = \max\{\cdot, \boldsymbol{0}\}$, and $\boldsymbol{\eta}_i(t)$ is an auxiliary variable used to estimate the global information of the MAS. Moreover, Yang et al. [82] prove that the dynamical system (13) is globally asymptotically stable and can find an accurate solution to the consensus optimization problem. This modeling approach allows us to analyze the collective behavior of MAS using tools from dynamical system theory, such as stability analysis, bifurcation theory, and attractor theory [83]. In 2004, ref. [84] presents a consensus algorithm for MAS with switching topology and time-delays. It uses graph theory to model the communication network among agents and analyzes the convergence properties of the consensus algorithm using Lyapunov functions and matrix theory. Thereafter, literature [85] presents a distributed neurodynamic approach for solving constrained optimization problems in MAS. It models the optimization problem as a dynamical system and employs this approach to design distributed algorithms, where each local objective function is minimized individually. In 2024, ref. [86] presents a distributed neurodynamic approach for solving a class of time-dependent non-linear equation systems, over multi-agent networks from a distributed optimization perspective. It models the non-linear equation system as a dynamical system to solve.

In summary, these works demonstrate the theoretical basis for applying neural dynamics to MAS by modeling them as dynamical systems. This modeling approach enables us to analyze and design coordination strategies for MAS using tools from dynamical system theory.

4.2. Core Models and Their Applications in Multi-Agent Systems

In the study of MAS through the lens of neural dynamics, several core models have emerged as foundational paradigms. These models, which conceptualize collective behaviors as the emergent results of interconnected dynam-

ical systems, are instrumental for both theoretical analysis and practical applications. Key among them are consensus protocols [87], containment control [88], flocking and swarming models [89], and replicator dynamics [90], each addressing a different facet of coordination and collective action.

Consensus protocols: Consensus is a fundamental problem in cooperative control, representing the challenge of getting all agents in a network to agree on a certain quantity of interest [91]. This agreement is achieved through local communication, where each agent updates its state based on the information received from its neighbors. The study of consensus in MAS is a benchmark for understanding multi-agent coordination, revealing the crucial role of the system's communication topology, which is often represented by a graph [91]. From a neural dynamics perspective, the consensus process is modeled as a system of coupled differential or difference equations, and its convergence is analyzed using tools from stability theory and algebraic graph theory [92]. A typical formulation is the leader-follower consensus, where one or more leader agents have a desired state, and the follower agents must converge to this state [93]. This framework is particularly relevant for guiding a group of autonomous agents. The applications of consensus models are extensive and varied, including attitude synchronization [94], sensor networks [95], and robot coordination [96]. Attitude synchronization [94] ensures that a fleet of satellites or spacecraft achieves and maintains a common orientation. Sensor networks [95] average measurements across a distributed network of sensors to obtain a more accurate global reading. Robot coordination [96] coordinates the velocity or position of multiple robots for tasks such as formation control and cooperative transport.

Containment control: Containment control is a sophisticated extension of consensus and formation control, designed for scenarios involving leaders and followers [97]. The primary objective is to drive a group of follower agents into the convex hull formed by a group of leader agents [98]. This ensures that the followers are contained within a safe or desired operational area defined by the leaders. The leaders can be static or dynamic, allowing for both fixed and time-varying containment regions [99,100]. The analysis of containment control protocols heavily relies on Lyapunov stability theory to mathematically guarantee that the followers will converge to and remain within the leaders' convex hull [97]. Moreover, neural networks are often employed to handle unknown non-linear dynamics within the agents, enhancing the model's applicability to real-world systems with uncertainties. Key applications include autonomous vehicle platooning [101] and search and rescue operations [102]. Autonomous vehicle platooning aims to guide a platoon of autonomous vehicles, where the lead vehicles define the path and boundaries for the followers [101]. Search and rescue operations aim at deploying a team of robots where leaders can identify and surround a hazardous area, ensuring follower robots operate within this safe zone [102].

Flocking and swarming models: Inspired by the collective motion observed in nature, such as flocks of birds and schools of fish, flocking and swarming models aim to produce cohesive, coordinated movement from simple, decentralized rules [103,104]. These models typically involve three fundamental rules for each agent, including cohesion [105], separation [106], and alignment [107]. Cohesion steers to move toward the average position of local flockmates [105]. Separation steers to avoid crowding local flockmates [106]. Moreover, alignment [107] steers toward the average heading of local flockmates. The resulting emergent behavior is a characteristic of complex dynamical systems, where organized global patterns arise from local interactions without a central coordinator. This self-organization is a powerful principle for engineering robust and scalable multi-agent systems. Applications are particularly prevalent in robotics and autonomous systems, such as unmanned aerial vehicle (UAV) swarms [108], autonomous underwater vehicles (AUVs) [108], and financial modeling [109]. UAV swarms aim at coordinating large groups of drones for surveillance, environmental monitoring, or entertainment [108]. The study of AUVs aims at enabling teams of AUVs to collaboratively map the seabed or track marine life [108]. As for the financial modeling, it focuses on simulating the collective behavior of traders in financial markets to understand market dynamics and return predictability [109].

Replicator dynamics: In the domain of multi-agent reinforcement learning (MARL) [110,111], a significant challenge is the non-stationarity of the environment; as one agent learns and changes its policy, the environment effectively changes for all other agents. This can lead to instability in standard learning algorithms like policy gradient methods [112]. Replicator dynamics [113], a model originating from evolutionary game theory, offers a robust framework for learning in such environments. It describes how the prevalence of different strategies in a population changes over time based on their relative success. The recently developed neural replicator dynamics (NeuRD) algorithm [114] integrates this classical dynamical model into modern deep reinforcement learning. NeuRD modifies the standard policy gradient update to mimic the replicator dynamics, allowing agents to adapt more dynamically and stably to the changing policies of others. This approach has proven effective in complex, competitive, and cooperative scenarios, with applications including game playing [115], resource allocation [116], and economic modeling [117]. Game playing focuses on training AI agents to play imperfect information games like poker and Goofspiel, where they must adapt to the strategies of their opponents [115]. Resource allocation aims

at managing the allocation of network resources or computational power among competing users or processes [116]. Moreover, economic modeling aims to simulate how competing companies adjust their strategies in a market environment [117].

In summary, these core models of neural dynamics in multi-agent systems provide a rich theoretical framework for understanding and designing collective behaviors. Their applications span a wide range of fields, demonstrating the versatility and power of this approach in addressing complex coordination challenges in MASs.

5. The Application of Neural Dynamics in Deep Learning

Deep learning has revolutionized various fields, including computer vision [118], natural language processing [119], and speech recognition [120]. However, the theoretical understanding of deep learning models remains limited. Neural dynamics provides a powerful framework for analyzing and understanding the behavior of deep learning models, enabling the design of effective deep neural network (DNN) architectures and optimizers. In this section, we first introduce the theoretical basis for applying neural dynamics to deep learning, and then review the core models and their applications in deep learning.

5.1. Theoretical Basis: Modeling DNN Architectures and Optimizers as Dynamical Systems

As shown in Figure 4a, DNNs can be effectively analyzed from the perspective of dynamical systems, where the hidden states through layers of a DNN are treated as the states over time in a dynamical system. Concurrently, as shown in Figure 4b, the optimization process of DNNs' parameters can also be modeled as a dynamical system, where the parameters of the DNN evolve over time based on the gradients of a loss function. This modeling provides a promising framework for understanding the behavior of DNNs, including their stability, convergence, and generalization properties [18,121].

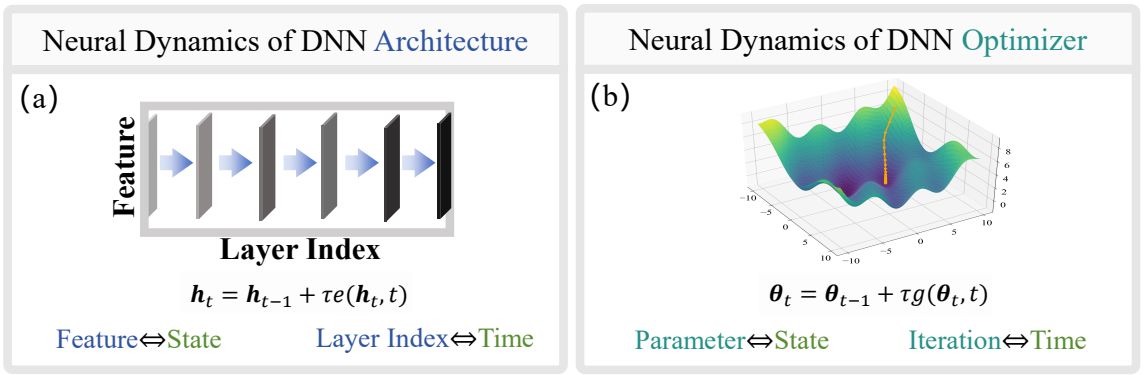


Figure 4. Relationship between deep learning and dynamical systems. (**a**) The DNN architecture defines a dynamical system that governs the evolution of hidden states $\boldsymbol{h}_t$ through layers, with the vector field e determining the evolution rule. (**b**) The optimizer defines a dynamical system that governs the evolution of the model parameter $\boldsymbol{\theta}_t$ during training, with the vector field g determining the evolution rule.

For the architectures of DNNs, ref. [122] discusses the connection between DNN architectures and dynamical systems in 2017. Specifically, consider a residual neural network (ResNet) [123] with L layers, where the input to the network is denoted by $\boldsymbol{x}_0 \in \mathbb{R}^n$, and the output of the l-th layer is denoted by $\boldsymbol{x}_l \in \mathbb{R}^n$ for $l = 1, 2, \ldots, L$. The ResNet architecture can be described by the following equation:

$$\boldsymbol{x}_l = \boldsymbol{x}_{l-1} + f(\boldsymbol{x}_{l-1}, \boldsymbol{\theta}_l), \tag{14}$$

where f is a nonlinear function representing the transformation applied by the l-th layer, and $\boldsymbol{\theta}_l$ is the parameter of the l-th layer. By interpreting the layer index l as a discrete time variable, the evolution of the hidden states through the layers of the ResNet can be viewed as a dynamical system. Specifically, the continuous-time limit of the ResNet can be described by the following ODE:

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = f(\boldsymbol{x}(t), t, \boldsymbol{\theta}), \tag{15}$$

where $\boldsymbol{x}(t)$ is the state of the system at time t, and $\boldsymbol{\theta}(t)$ is the parameter of the network at time t. This ODE describes how the hidden states evolve over time as they pass through the layers of the ResNet. The study in [122] further discusses how this dynamical system perspective enables the analysis of the stability properties of the DNN architecture using tools from dynamical system theory. Furthermore, the study in [124] presents new

forward propagation techniques inspired by the dynamical system theory, which alleviates exploding and vanishing gradient problems and leads to well-posed learning problems for arbitrarily deep networks. The backbone of this approach is interpreting deep learning as a parameter estimation problem of non-linear dynamical systems. Given this formulation, Haber et al. [124] analyze the stability and well-posedness of deep learning and use this new understanding to develop new network architectures. Moreover, the study in [125] presents that many architectures of DNNs can be associated with the discretization of differential equations. Specifically, pre-activation ResNet (PreResNet) [126] corresponds to the forward Euler discretization. PolyNet [127] can be regarded as an approximation of the backward Euler discretization. FractalNet [128] is similar to the second-order Runge-Kutta method [129].

As for the optimizers of DNNs, modern deep learning models are typically trained using variants of stochastic gradient descent (SGD) [130], which uses a randomly selected mini-batch of data to estimate the gradient and updates the parameter of the DNN iteratively. The update rule of SGD can be written as:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \nabla \mathcal{L}_{\mathcal{B}_k}(\boldsymbol{\theta}_k), \tag{16}$$

where $\boldsymbol{\theta}_k$ is the parameter of the DNN at iteration k, η is the learning rate, $\nabla \mathcal{L}_{\mathcal{B}_k}(\boldsymbol{\theta}_k)$ is the gradient of the loss function estimated using the mini-batch $\mathcal{B}_k$. Ref. [131] presents that the optimization process of SGD can be modeled as a stochastic differential equation (SDE). Specifically, the update rule of SGD (16) can be reformulated as

$$\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t = -\eta \nabla \mathcal{L}(\boldsymbol{\theta}_t) + \sqrt{\eta}\boldsymbol{\xi}_t, \tag{17}$$

where $\mathcal{L}(\boldsymbol{\theta}_t)$ is the loss function on the entire dataset at step t, and $\boldsymbol{\xi}_t = \sqrt{\eta}(\nabla \mathcal{L}(\boldsymbol{\theta}_t) - \nabla \mathcal{L}_{\mathcal{B}_k}(\boldsymbol{\theta}_t))$ is a random variable representing the noise introduced by the mini-batch sampling. In the continuous-time limit, this update rule can be described by the following SDE:

$$\mathrm{d}\boldsymbol{\theta}(t) = -\nabla \mathcal{L}(\boldsymbol{\theta}(t))\mathrm{d}t + \sqrt{\eta \Sigma(\boldsymbol{\theta}(t))}\mathrm{d}\boldsymbol{W}(t), \tag{18}$$

where $\Sigma(\boldsymbol{\theta}(t))$ is the covariance matrix of the noise, and $\boldsymbol{W}(t)$ is a standard Wiener process. This SDE describes how the parameters of the DNN evolve over time under the influence of both the deterministic gradient descent and the stochastic noise from the mini-batch sampling. Literature [131] further discusses how this SDE perspective enables the analysis of the convergence properties of SGD using tools from dynamical system theory. From this perspective, Chaudhari et al. [132] demonstrate that SGD minimizes an average potential over the posterior distribution of weights, subject to an entropic regularization term. This potential, however, is generally not the original loss function used to compute the gradients. Therefore, while SGD can be viewed as performing variational inference, it does so for a different objective than the one explicitly defined. Specifically, literature [132] shows that the continuous-time limit of SGD can be described by the following SDE:

$$\mathrm{d}\boldsymbol{\theta}(t) = -\nabla \mathcal{L}(\boldsymbol{\theta}(t))\mathrm{d}t + \sqrt{2\beta^{-1} D(\boldsymbol{\theta}(t))}\mathrm{d}\boldsymbol{W}(t), \tag{19}$$

where $D(\boldsymbol{\theta}(t)) = (1/N \sum_{k=1}^{N} \nabla \mathcal{L}_{\mathcal{B}_k}(\boldsymbol{\theta}_t) \nabla \mathcal{L}_{\mathcal{B}_k}(\boldsymbol{\theta}_t)^\top) - \nabla \mathcal{L}(\boldsymbol{\theta}_t) \nabla \mathcal{L}(\boldsymbol{\theta}_t)$ is the diffusion matrix representing the covariance of the gradient noise, and β is the inverse temperature that controls the strength of the gradient noise. This SDE describes how the parameter of a DNN evolve over time under the influence of both the deterministic gradient descent and the stochastic gradient noise. Ref. [132] further discusses how this SDE perspective enables the analysis of the generalization properties of SGD using tools from dynamical systems.

In summary, modeling DNN architectures and optimizers as dynamical systems provides a powerful framework for analyzing and understanding the behavior of deep learning models. This perspective enables the design of efficient and effective architectures and optimizers, ultimately advancing the field of deep learning.

5.2. Core Models and Their Applications in Deep Learning

Based on the theoretical basis introduced above, several core models have emerged as foundational paradigms for applying neural dynamics to deep learning. For the DNN architectures, the most representative models include the neural ODE [29], neural stochastic differential equation (SDE) [133], and deep equilibrium model (DEQ) [134]. For the optimizers of DNNs, SGD [131] and its variants, such as SGD with momentum [135] and Adam [136], are widely used in practice. While these gradient-based algorithms offer rapid convergence and computational efficiency, they are prone to getting trapped in local minima. Meta-heuristic optimization algorithms are known for their global search capabilities, which can help escape poor local minima; however, they often suffer from high computational costs and may converge slowly compared to gradient-based algorithms [137,138]. Therefore, inspired

by the perspective of neural dynamics, gradient-based algorithms are combined with meta-heuristic algorithms to train DNNs, a technique commonly referred to as collaborative neurodynamic optimization [82,139,140].

Neural ODE: In 2018, literature [29] presents the neural ODE [29], which generalizes the DNN architecture by allowing for continuous-depth models. In neural ODE, the transformation applied by each layer is replaced by a continuous-time ODE, enabling the model to adaptively choose the number of layers based on the complexity of the input data. This approach achieves competitive performance on various tasks while reducing the computational cost. Specifically, the forward propagation of the neural ODE can be described by

$$\begin{aligned} \boldsymbol{x}(T) &= \boldsymbol{x}(0) + \int_0^T f(\boldsymbol{x}(t), t, \boldsymbol{\theta}) \mathrm{d}t \\ &= \boldsymbol{x}(0) + \mathrm{ODESolver}(f, \boldsymbol{x}(0), 0, T, \boldsymbol{\theta}), \end{aligned} \tag{20}$$

where $\boldsymbol{x}(0)$ is the input to the neural ODE, $\boldsymbol{x}(T)$ is the output of the neural ODE at time T, and ODESolver is a numerical solver that computes the solution of the ODE defined by f from time 0 to time T. It is worth noting that while the continuous-time formulation offers theoretical elegance, the practical performance is intrinsically linked to the discretization scheme of this underlying solver. The choice of numerical method (e.g., Euler methods versus Runge-Kutta methods) directly impacts the computational error and the model's ability to handle stiff dynamics effectively. For the backpropagation of the neural ODEs, literature [29] uses the adjoint sensitivity method to compute the gradients of the loss function with respect to the parameters $\boldsymbol{\theta}$. This method involves solving a second ODE backward in time, allowing for the efficient computation of gradients without storing intermediate states. The adjoint state $\boldsymbol{a}(t)$ is defined as:

$$\boldsymbol{a}(t) = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}(t)}, \tag{21}$$

where $\mathcal{L}$ is the loss function. The dynamics of the adjoint state can be described by the following ODE:

$$\frac{\mathrm{d}\boldsymbol{a}(t)}{\mathrm{d}t} = -\boldsymbol{a}^\top(t) \frac{\partial f(\boldsymbol{x}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{x}(t)}. \tag{22}$$

The gradients of the loss function with respect to the parameters $\boldsymbol{\theta}$ can be computed as:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = -\int_0^T \boldsymbol{a}^\top(t) \frac{\partial f(\boldsymbol{x}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \mathrm{d}t. \tag{23}$$

This approach enables the efficient training of neural ODEs using standard optimization algorithms, such as SGD.

Neural SDE: In 2019, ref. [133] presents the neural SDE, which extends the neural ODE by incorporating stochasticity into the model. Neural ODE exhibits several advantages over traditional discrete DNNs in terms of memory efficiency, parameter efficiency, and explicit control of the numerical error. The neural ODE model lacks some regularization mechanisms commonly employed in discrete DNNs, which has been demonstrated to be crucial in reducing generalization errors. For example, dropout [141] is a widely used mechanism for preventing overfitting, which injects Gaussian random noise during the forward propagation [142]. However, due to their deterministic nature, neural ODEs preclude the direct application of these regularization mechanisms commonly used in discrete DNNs. Therefore, neural SDE incorporates stochastic noise injection-based regularization mechanisms into neural ODE to improve generalization performance. Neural SDE models the stochastic noise in the forward propagation using an SDE, providing theoretical insights into understanding why introducing stochasticity during DNN training and testing leads to improved generalization performance. Moreover, experiments in [133] show that the stochastic noise in neural SDE can enhance the robustness of the model against adversarial attacks. Specifically, the forward propagation of the neural SDE can be described by

$$\mathrm{d}\boldsymbol{x}(t) = f(\boldsymbol{x}(t), t; \boldsymbol{w})\mathrm{d}t + g(\boldsymbol{x}(t), t; \boldsymbol{v})\mathrm{d}\boldsymbol{W}(t), \tag{24}$$

where $\boldsymbol{x}(t)$ is the state of the system at time t, f is a drift function representing the deterministic part of the dynamics parameterized by $\boldsymbol{w}$, g is a diffusion function representing the stochastic part of the dynamics parameterized by $\boldsymbol{v}$, and $\boldsymbol{W}(t)$ is a standard Wiener process which is a continuous time stochastic process such that $\boldsymbol{W}(t+s) - \boldsymbol{W}(s)$ follows Gaussian distribution with mean zero and variance t. Equation (24) describes how the hidden states evolve over time under the influence of both deterministic and stochastic components. It can include many existing noise injection mechanisms with residual connections under different forms of g. For example, when $g(\boldsymbol{x}(t), t; \boldsymbol{v}) = \sigma \boldsymbol{x}(t)$ with $\sigma > 0$ being a constant, the neural SDE reduces to the case of multiplicative noise injection, which is commonly

used in dropout. For the backpropagation of the neural SDE, the study in [133] extends the adjoint sensitivity method used in neural ODE to compute the gradients of the loss function with respect to the parameters $\boldsymbol{w}$ and $\boldsymbol{v}$.

DEQ: In 2019, literature [134] presents the deep equilibrium model (DEQ), which is an implicit deep learning model that defines the hidden states of the network as the fixed points of a non-linear dynamical system. Historically, this concept traces its lineage back to Hopfield networks [17], which characterized memory retrieval as the system's convergence to an energy minimum. DEQs generalize this foundational principle by treating the entire depth of a neural network not as a fixed sequence of layers, but as a continuous fixed-point iteration problem. The motivation of DEQ is based on an observation that the hidden states of DNNs converge towards some fixed points, and DEQ is presented to find these equilibrium points via root-finding. DEQ can be viewed as an extension of the traditional DNN, where the depth of the network is effectively infinite, and the hidden states are obtained by solving a fixed-point equation. This approach can significantly improve memory efficiency, maintaining constant memory usage during training of large-scale sequence models. Experiments demonstrate that DEQ allows for efficient training and inference while maintaining competitive performance on various sequence modeling tasks. Specifically, DEQ directly computes the fixed point $\boldsymbol{z}^*_{1:T}$ of the following non-linear dynamical system:

$$\boldsymbol{z}^*_{1:T} = f_{\boldsymbol{\theta}}(\boldsymbol{z}^*_{1:T}, \boldsymbol{x}_{1:T}), \tag{25}$$

where $\boldsymbol{z}^*_{1:T}$ is the fixed point of the dynamical system defined by a non-linear transformation $f_{\boldsymbol{\theta}}$, $\boldsymbol{x}_{1:T}$ is the input sequence to the DEQ, T is the sequence length, and $\boldsymbol{\theta}$ is the parameter of the DEQ. The fixed point $\boldsymbol{z}^*$ can be computed using root-finding algorithms such as Newton's method [143] and Broyden's method [144]. For the backpropagation of DEQ, ref. [134] uses the implicit function theorem to compute the gradients of the loss function with respect to the parameters $\boldsymbol{\theta}$. This method involves solving a linear system, enabling the efficient computation of gradients without storing intermediate states. The computation method of the gradients is independent of the root-finding algorithm used to compute the fixed point or the internal structure of the transformation $f_{\boldsymbol{\theta}}$, and thus does not require any storage of the intermediate hidden states, which is necessary for backpropagation in conventional deep networks. Therefore, DEQ can achieve constant memory usage during training and inference.

Collaborative neurodynamic optimization: The concept of collaborative neurodynamic optimization is notably advanced by literature [82] for multi-objective distributed optimization. The framework presented by [82] employs a system of collaborative neural networks, with each network dedicated to a specific objective function and its associated constraints, to identify Pareto optimal solutions. Building upon this collaborative approach, ref. [139] introduces a neural solution tailored for time-varying nonconvex optimization with noise rejection. Their algorithm uniquely integrates evolutionary computation with neurodynamic methods, utilizing a meta-heuristic rule alongside a robust, gradient-based neural solution to effectively manage various forms of noise. More recently, the application of this framework is further expanded by literature [140] to the domain of index tracking and enhanced index tracking. To overcome the inherent nonconvexity of the objective function in these financial problems, they implemented a sparse Bayesian regression algorithm using multiple RNNs within the collaborative neurodynamic optimization structure.

In summary, these core models of neural dynamics in deep learning provide a rich theoretical framework for understanding and designing deep learning architectures and optimizers. Their applications span a wide range of fields, demonstrating the versatility of this approach in advancing the field of deep learning.

6. Future Directions

The application of neural dynamics in control systems, MASs, and deep learning is a rapidly evolving field with numerous promising directions for future research. Here, we outline several potential avenues for further exploration:

Integration of neural dynamics with reinforcement learning: The integration of neural dynamics with RL presents a promising avenue for future research. By combining the strengths of both approaches, it is possible to develop more robust and adaptive control strategies for complex systems. Future work could explore the use of neural dynamics to model the environment and the agent's behavior, enabling more efficient exploration and exploitation in RL algorithms [145]. Crucially, standard RL often lacks rigorous safety assurances. Future research should focus on integrating control-theoretic tools intrinsic to neural dynamics, such as Lyapunov stability analysis and barrier functions, into RL frameworks. This integration paves the way for "safe RL", where system constraints and stability are mathematically certified even during the learning process, addressing a fundamental limitation of current data-driven approaches.

Hybrid models combining neural dynamics with traditional control methods: The development of hybrid models that combine neural dynamics with traditional control methods, such as MPC and adaptive control systems [146,147], could lead to more effective and efficient control strategies. Future research could explore how to integrate these approaches to leverage the strengths of both, potentially leading to improved performance in a

wide range of applications.

Scalability and robustness in multi-agent systems: As MASs become increasingly complex and large-scale, ensuring scalability and robustness remains a significant challenge [148, 149]. Future research could focus on developing neural dynamic models that can effectively handle large numbers of agents while maintaining stability and performance. This may involve exploring new architectures, communication protocols, and learning algorithms that are specifically designed for large-scale MASs.

Neural dynamics for explainable AI: As deep learning models become more complex, there is a growing need for explainable AI techniques [150–152] that can provide insights into the decision-making processes of these models. Future research could investigate how neural dynamics can be leveraged to enhance the interpretability and transparency of deep learning models, enabling users to understand and trust their predictions.

Neural dynamics on hardware and edge computing: While current research focuses heavily on theoretical frameworks and software simulations, the physical realization of neural dynamics is critical for real-world deployment. Future research should investigate hardware-software co-design strategies to implement continuous-time neural dynamics on resource-constrained edge devices. A promising direction is the integration of neural dynamics with neuromorphic computing architectures, which can inherently support the parallel and event-driven nature of these models. This would significantly reduce latency and power consumption, enabling efficient embedded AI solutions for time-sensitive applications such as autonomous robotics and distributed sensor networks.

In summary, the future directions outlined above highlight the potential for continued advancements in the application of neural dynamics in control systems, MASs, and deep learning. By exploring these avenues, researchers can contribute to the development of more effective, efficient, and interpretable models that can address a wide range of challenges in these fields.

7. Conclusions

In this survey, we have examined how neural dynamics serves as a unifying bridge between the fields of control, represented by control systems and MASs, and computation, represented by deep learning. We have begun by introducing the theoretical basis for applying neural dynamics to these fields, highlighting how control systems, MASs, and deep learning models can be modeled as dynamical systems. Moreover, we have reviewed the core models and their applications in control systems, including MPC and RL-based control. For MASs, we have discussed models such as consensus protocols, containment control, flocking and swarming models, and replicator dynamics. In the context of deep learning, we have discussed models such as neural ODEs, neural SDEs, DEQs, and collaborative neurodynamic optimization. Finally, we have outlined several promising directions for future research in these rapidly evolving fields. By leveraging the strengths of neural dynamics, researchers can continue to advance the state-of-the-art models in control systems, MASs, and deep learning, ultimately leading to effective and efficient models that can address a wide range of challenges.

Funding

This research received no external funding.

Data Availability Statement

Not applicable.

Conflicts of Interest

The author declares no conflict of interest.

Use of AI and AI-Assisted Technologies

During the preparation of this work, the author used Gemini 3.0 to improve the readability. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the published article.

References

1. Awan, A.U.; Zamani, M. Reduced-Order Gaussian Processes for Partially Unknown Nonlinear Control Systems. *IEEE Trans. Autom. Control.* **2025**,*70*, 6893–6900.
2. Xie, M.; An, B.; Jia, X. Simultaneous Update of Sensing and Control DATA Using Free-Ride Codes in Vehicular Networks: An Age and Energy Perspective. *Comput. Netw.* **2024**, *252*, 110667.

3. Xiang, Z.; Guo, Y. Controlling Melody Structures in Automatic Game Soundtrack Compositions With Adversarial Learning Guided Gaussian Mixture Models. *IEEE Trans. Games* **2021**, *13*, 193–204.
4. Liu, H.; Guo, D.; Cangelosi, A. Embodied Intelligence: A Synergy of Morphology, Action, Perception and Learning. *ACM Comput. Surv.* **2025**,*57*, 1–36.
5. Ren, L.; Dong, J.; Liu, S.; et al. Embodied Intelligence Toward Future Smart Manufacturing in the Era of AI Foundation Model. *IEEE/ASME Trans. Mechatron.* **2025**, *30*, 2632–2642.
6. Shen, T.; Sun, J.; Kong, S.; et al. The Journey/DAO/TAO of Embodied Intelligence: From Large Models to Foundation Intelligence and Parallel Intelligence. *IEEE/CAA J. Autom. Sin.* **2024**, *11*, 1313–1316.
7. Li, J.; Guan, Y.; Deng, T.; et al. Periodic-Noise-Tolerant Neurodynamic Approach for kWTA Operation Applied to Opinions Evolution. *Neural Netw.* **2025**, *191*, 107839.
8. Wang, C.; Wang, Y.; Yuan, Y.; et al. Joint Computation Offloading and Resource Allocation for End-Edge Collaboration in Internet of Vehicles via Multi-Agent Reinforcement Learning. *Neural Netw.* **2024**, *179*, 106621.
9. He, J.; Treude, C.; Lo, D. LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision, and the Road Ahead. *ACM Trans. Softw. Eng. Methodol.* **2025**,*34*, 1–30.
10. Kumar, P. Large Language Models (LLMs): Survey, Technical Frameworks, and Future Challenges. *Artif. Intell. Rev.* **2024**,*57*, 260.
11. Mudrik, N.; Chen, Y.; Yezerets, E.; et al. Decomposed Linear Dynamical Systems (dLDS) for Learning the Latent Components of Neural Dynamics. *J. Mach. Learn. Res.* **2024**,*25*, 1–44.
12. Tsuda, I. Toward an Interpretation of Dynamic Neural Activity in Terms of Chaotic Dynamical Systems. *Behav. Brain Sci.* **2001**,*24*, 793–810.
13. Chialvo, D.R. Emergent Complex neural Dynamics. *Nat. Phys.* **2010**,*6*, 744–750.
14. Mondié, S.; Egorov, A.; Ortiz, R. Lyapunov Stability Tests for Integral Delay Systems. *Annu. Rev. Control.* **2025**,*59*, 100985.
15. Hu, J.; Hu, Y.; Chen, W.; et al. Attractor Memory for Long-Term Time Series Forecasting: A Chaos Perspective. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 20786–20818.
16. Dueñas, J.; Núñez, C.; Obaya, R. Bifurcation Theory of Attractors and Minimal Sets in d-Concave Nonautonomous Scalar Ordinary Differential Equations. *J. Differ. Equ.* **2023**, *361*, 138–182.
17. Zhang, S.; Chen, C.; Zhang, Y.; et al. Multidirectional Multidouble-Scroll Hopfield Neural Network With Application to Image Encryption. *IEEE Trans. Syst. Man Cybern. Syst.* **2025**,*55*, 735–746.
18. Liu, G.-H.; Theodorou, E.A. Deep Learning Theory Review: An Optimal Control and Dynamical Systems Perspective. *arXiv* **2019**, arXiv.1908.10920.
19. Coombes, S.; Wedgwood, K.C.A. *Neurodynamics: An Applied Mathematics Perspective*, 1st ed.; Springer: Cham, Switzerland, 2023.
20. Stuart, A.M. Numerical Analysis of DYNAMICAL systems. *Acta Numer.* **1994**, *3*, 467–572.
21. Colonius, F.; Kliemann, W. Some Aspects of Control Systems as Dynamical Systems. *J. Dyn. Differ. Equ.* **1993**,*5*, 469–494.
22. Yu, W.; Chen, G.; Cao, M. Some Necessary and Sufficient Conditions for Second-Order Consensus in Multi-Agent Dynamical Systems. *Automatica* **2010**,*46*, 1089–1095.
23. Bahri, Y.; Kadmon, J.; Pennington, J.; et al. Statistical Mechanics of Deep Learning. *Annu. Rev. Condens. Matter Phys.* **2020**, *11*, 501–528.
24. Hirsch, M.W.; Smale, S.; Devaney, R.L. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*, 3rd ed.; Academic Press: Cambridge, MA, USA, 2013.
25. Hartman, P. *Ordinary Differential Equations*, 2nd ed.; SIAM: Philadelphia, PA, USA, 2002.
26. Hu, Y.; Abu-Dakka, F.J.; Chen, F.; et al. Fusion Dynamical Systems With Machine Learning in Imitation Learning: A Comprehensive Overview. *Inf. Fusion* **2024**, *108*, 102379.
27. Chakraborty, D.; Chung, S.W.; Arcomano, T.; et al. Divide and Conquer: Learning Chaotic Dynamical Systems With Multistep Penalty Neural Ordinary Differential Equations. *Comput. Methods Appl. Mech. Eng.* **2024**, *432*, 117442.
28. Maranhão, D.M.; Medrano-T, R.O. Periodicity in the Asymmetrical Quartic Map. *Chaos Solitons Fractals* **2024**, *186*, 115204.
29. Chen, R.T.Q.; Rubanova, Y.; Bettencourt, J.; et al. Neural Ordinary Differential Equations. In Proceedings of the Advances in Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; pp. 1–13.
30. Jin, Y.; Hou, L.; Zhong, S. Extended Dynamic Mode Decomposition With Invertible Dictionary Learning. *Neural Netw.* **2024**, *173*, 106177.
31. Volkmann, E.; Brändle, A.; Durstewitz, D.; et al. A Scalable Generative Model for Dynamical System Reconstruction From Neuroimaging Data. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 80328–80362.
32. Glad, T.; Ljung, L. *Control Theory*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2000.
33. Bernauer, C.; Leitner, P.; Zapata, A.; et al. Segmentation-Based Closed-Loop Layer Height Control for Enhancing Stability and Dimensional Accuracy in Wire-Based Laser Metal Deposition. *Robot. Comput. Integr. Manuf.* **2024**, *86*, 102683.
34. Shim, H.; Jo, N.H. An Almost Necessary and Sufficient Condition for Robust Stability of Closed-Loop Systems With Disturbance Observer. *Automatica* **2009**, *45*, 296–299.
35. Jain, S.; Garg, V. A Review of Open Loop Control Strategies for Shades, Blinds and Integrated Lighting by Use of

Real-Time Daylight Prediction Methods. *Build. Environ.* **2018**, *135*, 352–364.
36. Salo, M.; Tuusa, H. A Novel Open-Loop Control Method for a Current-Source Active Power Filter. *IEEE Trans. Ind. Electron.* **2003**, *50*, 313–321.
37. Vengsungnle, P.; Poojeera, S.; Srichat, A.; et al. Optimized Performance of Closed Loop Control Electromagnetic Field for the Electric Generators With Energy Storage. *Eng. Sci.* **2024**,*30*, 1173.
38. Rosen, R. Dynamical Systems and Control Theory. In *Optimality Principles in Biology*; Springer: New York, NK, USA, 1967; pp. 155–165.
39. Grüne, L. On the Relation Between Discounted and Average Optimal Value Functions. *J. Differ. Equ.* **1998**, *148*, 65–99.
40. Baratchart, L.; Chyba, M.; Pomet, J.-B. A Grobman-Hartman Theorem for Control Systems. *J. Dyn. Differ. Equ.* **2007**, *19*, 75–107.
41. Zhong, X.; Chen, Z.; Huang, Y. Equi-Invariability, Bounded Invariance Complexity and L-Stability for Control Systems. *Sci. China Math.* **2021**, *64*, 2275–2294.
42. Yang, R.; Chen, E.; Yang, J.; et al. Bowen's Equations for Invariance Pressure of Control Systems. *SIAM J. Control. Optim.* **2025**, *63*, 1104–1128. https://doi.org/10.1137/23M1607684.
43. Course, K.; Nair, P.B. State Estimation of a Physical System With Unknown Governing Equations. *Nature* **2023**, *662*, 261–267.
44. Brunton, S.L.; Proctor, J.L.; Kutz, J.N. Discovering Governing Equations From Data by Sparse Identification of Nonlinear Dynamical Systems. *Proc. Natl. Acad. Sci. USA* **2016**, *113*, 3932–3937.
45. Jia, D.; Zhou, X.; Li, S.; et al. Governing Equation Discovery Based on Causal Graph for Nonlinear Dynamic Systems. *Mach. Learn. Sci. Technol.* **2023**, *4*, 045008.
46. Schwenzer, M.; Ay, M.; Bergs, T.; et al. Review on Model Predictive Control: An Engineering Perspective. *Int. J. Adv. Manuf. Technol.* **2021**, *117*, 1327–1349.
47. Köhler, J.; Müller, M.A.; Allgöwer, F. Analysis and Design of Model Predictive Control Frameworks for Dynamic Operation—An Overview. *Annu. Rev. Control.* **2024**, *57*, 100929.
48. Abdelghany, M.B.; Al-Durra, A.; Zeineldin, H.H.; et al. A Coordinated Multitimescale Model Predictive Control for Output Power Smoothing in Hybrid Microgrid Incorporating Hydrogen Energy Storage. *IEEE Trans. Ind. Inform.* **2024**, *20*, 10987–11001.
49. Meyn, S. *Control Systems and Reinforcement Learning*, 1st ed.; Cambridge University Press: Cambridge, UK, 2022.
50. Kuhnle, A.; Kaiser, J.-P.; Theiß, F.; et al. Designing an Adaptive Production Control System Using Reinforcement Learning. *J. Intell. Manuf.* **2021**,*32*, 855–876.
51. Kiumarsi, B.; Vamvoudakis, K.G.; Modares, H.; et al. Optimal and Autonomous Control Using Reinforcement Learning: A Survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 2042–2062.
52. Chen, X.; Heidarinejad, M.; Liu, J.; et al. Distributed Economic MPC: Application to a Nonlinear Chemical Process Network. *J. Process Control.* **2012**, *22*, 689–699.
53. Cavone, G.; Bozza, A.; Carli, R.; et al. MPC-Based Process Control of Deep Drawing: An Industry 4.0 Case Study in Automotive. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 1586–1598.
54. Huang, K.; Wei, K.; Li, F.; et al. LSTM-MPC: A Deep Learning Based Predictive Control Method for Multimode Process Control. *IEEE Trans. Ind. Electron.* **2023**, *70*, 11544.
55. Ma, D.; Lv, J.; Xu, C.; et al. Logic-Adaptive Discrete Neural Dynamics for Distributed Cooperative Control of Multi-Robot Systems via Minimum Infinity Norm Optimization. *IEEE Trans. Fuzzy Syst.* **2025**, *33*, 1–10.
56. Li, X.; Ren, X.; Zhang, Z.; et al. A Varying-Parameter Complementary Neural Network for Multi-Robot Tracking and Formation via Model Predictive Control. *Neurocomputing* **2024**, *609*, 128384.
57. Tang, Z.; Zhang, Y.; Ming, L. Novel Snap-Layer MMPC Scheme via Neural Dynamics Equivalency and Solver for Redundant Robot Arms With Five-Layer Physical Limits. *IEEE Trans. Neural Netw. Learn. Syst.* **2025**, *36*, 3534–3546.
58. Li, S.; Zhou, M.; Luo, X. Modified Primal-Dual Neural Networks for Motion Control of Redundant Manipulators With Dynamic Rejection of Harmonic Noises. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 4791–4801.
59. Cao, Z.; Xiao, Q.; Huang, R.; et al. Robust Neuro-Optimal Control of Underactuated Snake Robots With Experience Replay. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 208–217.
60. Yan, J.; Jin, L.; Hu, B. Data-Driven Model Predictive Control for Redundant Manipulators With Unknown Model. *IEEE Trans. Cybern.* **2024**, *54*, 5901–5911.
61. Iftikhar, A.; Ghazanfar, M.A.; Ayub, M.; et al. A Reinforcement Learning Recommender System Using Bi-Clustering and Markov Decision Process. *Expert Syst. Appl.* **2024**, *237*, 121541.
62. Li, G.; Li, X.; Li, J.; et al. PTMB: An Online Satellite Task Scheduling Framework Based on Pre-Trained Markov Decision Process for Multi-Task Scenario. *Knowl.-Based Syst.* **2024**, *284*, 111339.
63. Chebotar, Y.; Hausman, K.; Zhang, M.; et al. Combining Model-Based and Model-Free Updates For Trajectory-Centric Reinforcement Learning. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 703–711.
64. Gullapalli, V.; Franklin, J.; Benbrahim, H. Acquiring Robot Skills via Reinforcement Learning. *IEEE Control. Syst. Mag.*

1994, *14*, 13–24.

65. Raffin, A.; Kober, J.; Stulp, F. Smooth Exploration for Robotic Reinforcement Learning. In Proceedings of the 6th Conference on Robot Learning, Auckland, New Zealand, 14–18 December 2022; pp. 1634–1644.
66. Sallab, A.E.; Abdou, M.; Perot, E.; et al. Deep Reinforcement Learning Framework for Autonomous Driving. *arXiv* **2017**, arXiv:1704.02532.
67. Zeng, Z.; Wang, J.; Liao, X. Global Exponential Stability of a General Class of Recurrent Neural Networks With Time-Varying Delays. *IEEE Trans. Circuits Syst. I: Fundam. Theory Appl.* **2003**, *50*, 1353–1358.
68. Li, Z.; Li, S. Recursive Recurrent Neural Network: A Novel Model for Manipulator Control With Different Levels of Physical Constraints. *CAAI Trans. Intell. Technol.* **2023**, *8*, 622–634,
69. Doostmohammadian, M.; Aghasi, A.; Pirani, M.; et al. Survey of Distributed Algorithms for Resource Allocation Over Multi-Agent Systems. *Annu. Rev. Control.* **2025**, *59*, 100983.
70. Su, H.; Chen, M.Z.Q.; Lam, J.; et al. Semi-Global Leader-Following Consensus of Linear Multi-Agent Systems With Input Saturation via Low Gain Feedback. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2013**, *60*, 1881–1889.
71. Li, K.; Liu, Q.; Zeng, Z. Multiagent System With Periodic and Event-Triggered Communications for Solving Distributed Resource Allocation Problem. *IEEE Trans. Syst. Man Cybern. Syst.* **2023**, *53*, 6245–6256.
72. Kalyva, D.; Psillakis, H.E. Distributed Control of a Mobile Robot Multi-AGENT System for Nash Equilibrium Seeking With Sampled Neighbor Information. *Automatica* **2024**, *166*, 111712.
73. Su, H.; Zhang, J.; Zeng, Z. Formation-Containment Control of Multi-Robot Systems Under a Stochastic Sampling Mechanism. *Sci. China Technol. Sci.* **2020**, *63*, 1025–1034.
74. Sun, Z.; Yu, Z.; Guo, B.; et al. Integrated Sensing and Communication for Effective Multi-Agent Cooperation Systems. *IEEE Commun. Mag.* **2024**, *62*, 68–73.
75. Caprioli, C. The Integration of Multi-Agent System and Multicriteria Analysis for Developing Participatory Planning Alternatives in Urban Contexts. *Environ. Impact Assess. Rev.* **2025**, *113*, 107855.
76. Izmirlioglu, Y.; Pham, L.; Son, T.C.; et al. A Survey of Multi-Agent Systems for Smartgrids. *Energies* **2024**, *17*, 3620.
77. Singh, A.; Raut, G.; Choudhary, A. Multi-Agent Collaborative Perception for Robotic Fleet: A Systematic Review. In Proceedings of the European Conference on Computer Vision, Dublin, Ireland, 22–23 October 2025; pp. 1–15.
78. Hou, X.; Wang, J.; Du, J.; et al. Distributed Machine Learning for Autonomous Agent Swarm: A Survey. *IEEE Commun. Surv. Tutor.* **2025**, *28*, 1597–1636
79. Katsikis, V.N.; Liao, B.; Hua, C. Survey of Neurodynamic Methods for Control and Computation in Multi-Agent Systems. *Symmetry* **2025**, *17*, 936.
80. Deng, Q.; Zhang, Y. Distributed Near-Optimal Consensus of Double-Integrator Multi-Agent Systems With Input Constraints. In Proceedings of the International Joint Conference on Neural Networks, Shenzhen, China, 18–22 July 2021; pp. 1–6.
81. Deng, Q.; Liu, K.; Zhang, Y. Privacy-Preserving Consensus of Double-Integrator Multi-Agent Systems With Input Constraints. *IEEE Trans. Emerg. Top. Comput. Intell.* **2024**, *8*, 4119–4129.
82. Yang, S.; Liu, Q.; Wang, J. A Collaborative Neurodynamic Approach to Multiple-Objective Distributed Optimization. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 981–992.
83. Leonard, N.E. Multi-Agent System Dynamics: Bifurcation and Behavior of Animal Groups. *Annu. Rev. Control.* **2014**, *38*, 171–183.
84. Olfati-Saber, R.; Murray, R. Consensus Problems in Networks of Agents With Switching Topology and Time-Delays. *IEEE Trans. Autom. Control.* **2004**, *49*, 1520–1533.
85. Le, X.; Chen, S.; Yan, Z.; et al. A Neurodynamic Approach to Distributed Optimization With Globally Coupled Constraints. *IEEE Trans. Cybern.* **2018**, *48*, 3149–3158.
86. Li, H.; Qin, S. A Neurodynamic Approach for Solving Time-Dependent Nonlinear Equation System: A Distributed Optimization Perspective. *IEEE Trans. Ind. Inform.* **2024**, *20*, 10031–10039.
87. Li, Z.; Wen, G.; Duan, Z.; et al. Designing Fully Distributed Consensus Protocols for Linear Multi-Agent Systems With Directed Graphs. *IEEE Trans. Autom. Control.* **2015**, *60*, 1152–1157.
88. Lü, H.; He, W.; Han, Q.-L.; et al. Finite-Time Containment Control for Nonlinear Multi-Agent Systems With External Disturbances. *Inf. Sci.* **2020**, *512*, 338–351.
89. Sar, G.K.; Ghosh, D. Flocking and Swarming in a Multi-Agent Dynamical System. *Chaos Interdiscip. J. Nonlinear Sci.* **2023**, *33*, 12306.
90. Khaw, Y.N.; Kowalczyk, R.; Vo, Q.B.; et al. Transition-State Replicator Dynamics. *Expert Syst. Appl.* **2021**, *182*, 115254.
91. Amirkhani, A.; Barshooi, A.H. Consensus in Multi-Agent Systems: A Review. *Artif. Intell. Rev.* **2022**, *55*, 3897–3935.
92. Olfati-Saber, R.; Fax, J.A.; Murray, R.M. Consensus and Cooperation in Networked Multi-Agent Systems. *Proc. IEEE* **2007**, *95*, 215–233.
93. Lin, L.; Cao, J.; Lam, J.; et al. Leader-Follower Consensus Over Finite Fields. *IEEE Trans. Autom. Control.* **2024**, *69*, 4718–4725.
94. Thunberg, J.; Song, W.; Montijano, E.; et al. Distributed Attitude Synchronization Control of Multi-Agent Systems With Switching Topologies. *Automatica* **2014**, *50*, 832–840.

95. Wu, J.; Yuan, S.; Ji, S.; et al. Multi-Agent System Design and Evaluation for Collaborative Wireless Sensor Network in Large Structure Health Monitoring. *Expert Syst. Appl.* **2010**, *37*, 2028–2036.
96. Ota, J. Multi-Agent Robot Systems as Distributed Autonomous Systems. *Adv. Eng. Inform.* **2006**, *20*, 59–70.
97. Ji, M.; Ferrari-Trecate, G.; Egerstedt, M.; et al. Containment Control in Mobile Networks. *IEEE Trans. Autom. Control.* **2008**, *53*, 1972–1975.
98. Ji, Z.; Wang, Z.; Lin, H.; et al. Interconnection Topologies for Multi-Agent Coordination Under Leader-Follower Framework. *Automatica* **2009**, *45*, 2857–2863.
99. Liang, H.; Zhou, Y.; Ma, H.; et al. Adaptive Distributed Observer Approach for Cooperative Containment Control of Nonidentical Networks. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 299–307.
100. Li, J.; Ren, W.; Xu, S. Distributed Containment Control With Multiple Dynamic Leaders for Double-Integrator Dynamics Using Only Position Measurements. *IEEE Trans. Autom. Control.* **2012**, *57*, 1553–1559.
101. Dai, S.; Li, S.; Tang, H.;et al. MARP: A Cooperative Multiagent DRL System for Connected Autonomous Vehicle Platooning. *IEEE Internet Things J.* **2024**, *11*, 32454–32463.
102. Allouche, M.K.; Boukhtouta, A. Multi-Agent Coordination by Temporal Plan Fusion: Application to Combat Search and Rescue. *Inf. Fusion* **2010**, *11*, 220–232.
103. Olfati-Saber, R. Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory. *IEEE Trans. Autom. Control.* **2006**, *51*, 401–420.
104. Chen, C.; Hou, Y.; Ong, Y. A Conceptual Modeling of Flocking-Regulated Multi-Agent Reinforcement Learning. In Proceedings of the International Joint Conference on Neural Networks, Vancouver, BC, Canada, 24–29 July 2016; pp. 5256–5262.
105. Li, C.; Yang, Y.; Jiang, G.; et al. A Flocking Control Algorithm of Multi-Agent Systems Based on Cohesion of the Potential Function. *Complex Intell. Syst.* **2024**, *10*, 2585–2604.
106. Brittain, M.; Wei, P. Scalable Autonomous Separation Assurance With Heterogeneous Multi-Agent Reinforcement Learning. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 2837–2848.
107. Zhang, H.-T.; Zhai, C.; Chen, Z. A General Alignment Repulsion Algorithm for Flocking of Multi-Agent Systems. *IEEE Trans. Autom. Control.* **2011**, *56*, 430–435.
108. Tahir, A.; Böling, J.; Haghbayan, M.-H.; et al. Swarms of Unmanned Aerial Vehicles—A Survey. *J. Ind. Inf. Integr.* **2019**, *16*, 100106.
109. Chen, T.-T.; Zheng, B.; Li, Y.; et al. New Approaches in Agent-Based Modeling of Complex Financial Systems. *Front. Phys.* **2017**, *12*, 128905.
110. Foerster, J.; Assael, I.A.; de Freitas, N.; et al. Learning to Communicate With Deep Multi-Agent Reinforcement Learning. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 2145–2153.
111. Wen, M.; Kuba, J.; Lin, R.; et al. Multi-Agent Reinforcement Learning is a Sequence Modeling Problem. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 16509–16521.
112. Zhang, J.; Koppel, A.; Bedi, A.S.; et al. Variational Policy Gradient Method for Reinforcement Learning With General Utilities. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 4572–4583.
113. Tan, S.; Wang, Y. Graphical Nash Equilibria and Replicator Dynamics on Complex Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 1831–1842.
114. Hennes, D.; Morrill, D.; Omidshafiei, S.; et al. Neural Replicator Dynamics: Multiagent Learning via Hedging Policy Gradients. In Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems, Auckland, New Zealand, 9–13 May 2020; pp. 492–501.
115. Roca, C.P.; Cuesta, J.A.; Sánchez, A. Evolutionary Game Theory: Temporal and Spatial Effects Beyond Replicator Dynamics. *Phys. Life Rev.* **2009**, *6*, 208–249.
116. Wang, Q.; He, N.; Chen, X. Replicator Dynamics for Public Goods Game With Resource Allocation in Large Populations. *Appl. Math. Comput.* **2018**, *328*, 162–170.
117. Branch, W.A.; McGough, B. Replicator Dynamics in a Cobweb Model With Rationally Heterogeneous Expectations. *J. Econ. Behav. Organ.* **2008**, *65*, 224–244.
118. Ioannidou, A.; Chatzilari, E.; Nikolopoulos, S.; et al. Deep Learning Advances in Computer Vision With 3D Data: A Survey. *ACM Comput. Surv.* **2017**, *50*, 20.
119. Otter, D.W.; Medina, J.R.; Kalita, J.K. A Survey of the Usages of Deep Learning for Natural Language Processing. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 604–624.
120. Kheddar, H.; Hemis, M.; Himeur, Y. Automatic Speech Recognition Using Advanced Deep Learning Approaches: A Survey. *Inf. Fusion* **2024**, *109*, 102422.
121. Zeng, Z.; Wang, J.; Liao, X. Stability Analysis of Delayed Cellular Neural Networks Described Using Cloning Templates. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2004**, *51*, 2313–2324.
122. Weinan, E. A Proposal on Machine Learning via Dynamical Systems. *Commun. Math. Stat.* **2017**, *5*, 1–11.
123. He, K.; Zhang, X.; Ren, S.; et al. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.

124. Haber, E.; Ruthotto, L. Stable Architectures for Deep Neural Networks. *Inverse Probl.* **2017**, *34*, 014004,
125. Lu, Y.; Zhong, A.; Li, Q.; et al. Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018; pp. 3276–3285.
126. He, K.; Zhang, X.; Ren, S.; et al. Identity Mappings in Deep Residual Networks. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 630–645.
127. Zhang, X.; Li, Z.; Loy, C.C.; et al. PolyNet: A Pursuit of Structural Diversity in Very Deep Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 3900–3908.
128. Larsson, G.; Maire, M.; Shakhnarovich, G. FractalNet: Ultra-Deep Neural Networks Without Residuals, In Proceedings of the International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016; pp. 1–11.
129. Zhang, Z.; Zheng, L.; Li, L.; et al. A New Finite-Time Varying-Parameter Convergent-Differential Neural-Network for Solving Nonlinear and Nonconvex Optimization Problems. *Neurocomputing* **2018**, *319*, 74–83.
130. Sclocchi, A.; Wyart, M. On the Different Regimes of Stochastic Gradient Descent. *Proc. Natl. Acad. Sci. USA* **2024**, *121*, e2316301121.
131. Li, Q.; Tai, C. Stochastic Modified Equations and Adaptive Stochastic Gradient Algorithms. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 2101–2110.
132. Chaudhari, P.; Soatto, S. Stochastic Gradient Descent Performs Variational Inference, Converges to Limit Cycles for Deep Networks. In Proceedings of the Proceedings of the Information Theory and Applications Workshop, San Diego, CA, USA, 11–16 February 2018; pp. 1–10.
133. Liu, X.; Xiao, T.; Si, S.; et al. Neural SDE: Stabilizing Neural Ode Networks With Stochastic Noise. *arXiv* **2019**, arXiv:1906.02355
134. Bai, S.; Kolter, J.Z.; Koltun, V. Deep Equilibrium Models. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 690–701.
135. Sutskever, I.; Martens, J.; Dahl, G.; et al. On the Importance of Initialization and Momentum in Deep Learning. *Proc. Int. Conf. Mach. Learn.* **2013**, *28*, 1139–1147.
136. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015; pp. 1–15.
137. Si, T.; Bagchi, J.; Miranda, P.B. Artificial Neural Network Training Using Metaheuristics for Medical Data Classification: An Experimental Study. *Expert Syst. Appl.* **2022**, *193*, 116423.
138. Hu, X.; Wang, J. Solving Pseudomonotone Variational Inequalities and Pseudoconvex Optimization Problems Using the Projection Neural Network. *IEEE Trans. Neural Netw.* **2006**, *17*, 1487–1499.
139. Wei, L.; Jin, L. Collaborative Neural Solution for Time-Varying Nonconvex Optimization With Noise Rejection. *IEEE Trans. Emerg. Top. Comput. Intell.* **2024**, *8*, 2935–2948.
140. Zhang, F.; Wang, J. Index Tracking via Sparse Bayesian Regression and Collaborative Neurodynamic Optimization. *IEEE Trans. Cybern.* **2025**, *55*, 1238–1249.
141. Zhang, Z.; Xu, Z.-Q.J. Implicit Regularization of Dropout. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 4206–4217.
142. Huang, W.; Cui, Y.; Li, H.; et al. Practical Probabilistic Model-Based Reinforcement Learning by Integrating Dropout Uncertainty and Trajectory Sampling. *IEEE Trans. Neural Netw. Learn. Syst.* **2025**, *36*, 12812–12826.
143. Bottou, L.; Curtis, F.E.; Nocedal, J. Optimization Methods for Large-Scale Machine Learning. *SIAM Rev.* **2018**, *60*, 223–311.
144. Broyden, C. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Math. Comput.* **1965**, *6*, 577–593.
145. Zhu, Q.; Wu, X.; Lin, Q.; et al. Two-Stage Evolutionary Reinforcement Learning for Enhancing Exploration and Exploitation. *Proc. AAAI Conf. Artif. Intell.* **2024**, *38*, 20892–20900.
146. Sui, S.; Chen, C.L.P.; Tong, S. Command Filter-Based Predefined Time Adaptive Control for Nonlinear Systems. *IEEE Trans. Autom. Control.* **2024**, *69*, 7863–7870.
147. Zhang, S.; Duan, G. Robust Adaptive Control of Uncertain Fully Actuated Systems With Unknown Parameters and Perturbed Input Matrices. *IEEE Trans. Cybern.* **2025**, *55*, 927–938.
148. Golmisheh, F.M.; Shamaghdari, S. Optimal Robust Formation of Multi-Agent Systems as Adversarial Graphical Apprentice Games With Inverse Reinforcement Learning. *IEEE Trans. Autom. Sci. Eng.* **2025**, *22*, 4867–4880,
149. Iervolino, R.; Manfredi, S. Global Stability of Multi-Agent Systems With Heterogeneous Transmission and Perception Functions. *Automatica* **2024**, *162*, 111510.
150. Wani, N.A.; Kumar, R.; Mamta Bedi, J.; et al. Explainable AI-Driven IoMT Fusion: Unravelling Techniques, Opportunities, and Challenges With Explainable AI in Healthcare. *Information Fusion* **2024**, *110*, 102472.
151. Černevičienė, J.; Kabašinskas, A. Explainable Artificial Intelligence (XAI) in Finance: A Systematic Literature Review. *Artif. Intell. Rev.* **2024**, *57*, 216.
152. Bennetot, A.; Donadello, I.; Haouari, A.E.Q.E.; et al. A Practical Tutorial on Explainable AI Techniques. *ACM Comput. Surv.* **2024**, *57*, 1–44.