Future Technology
Open Access Journal
ISSN 2832-0379

February 2026| Volume 05 | Issue 01 | Pages 148-158

Journal homepage: https://fupubco.com/futech

https://doi.org/10.55670/fpll.futech.5.1.13

Future Publishing LLC

Article

Exploring various neural network configurations for the NN-based MPC in a multi-agent system

Piyush Chaubey[1,2], Anilkumar Markana[1*], Dhaval Vyas[1], Deepak Kumar Goyal[2]

[1]Pandit Deendayal Energy University, Gandhinagar Gujarat, India
[2]Government Engineering College, Bharatpur Rajasthan, India

ARTICLE INFO

Article history:
Received 18 August 2025
Received in revised form
15 October 2025
Accepted 02 November 2025

Keywords:
Multimodal fusion, Context awareness, Smart kitchens, Reinforcement learning, Personalized recommendation

*Corresponding author
Email address:
Anil.markana@spt.pdpu.ac.in

DOI: 10.55670/fpll.futech.5.1.13

ABSTRACT

Multi-robot cooperation, unmanned aerial vehicle (UAV) formation control, intelligent transport systems, and distributed sensor networks are just a few domains where multi-agent systems are crucial, as they require coordinated behavior to achieve common goals such as exploration, resource allocation, distributed sensing, and target tracking. This paper investigates various neural network configurations utilized in the NN-MPC framework for consensus control of multi-agent robotic systems. The NN-MPC control is applied to the consensus problem of a leader-follower multi-agent system, where agents coordinate to achieve collective behavior. In this approach, MPC is utilized to predict the future values of the control objective, which is optimized by minimizing a cost function with various neural network architectures. Different neural network configurations based on feed-forward, recurrent neural networks, Fitnet, and cascade networks are explored for the NN-MPC-based multi-agent systems. The analysis is performed through a simulation-based model of a quadrotor fleet system. Results show that the follower agents achieve consensus 60% faster than with RNN-MPC in comparison to the feedforward neural network, whereas the results are more effective when compared with the cascade network configuration-based MPC, where agents reach consensus 90% early if paired with suitable training structures. Overall, the article contributes to the recent topic of research on learning-based MPC of the multi-agent system in achieving consensus for the leader-follower strategy.

1. Introduction

Multi-agent systems are gaining prominence in the area of applications like autonomous vehicles, smart grids, healthcare systems, and environmental monitoring [1]. MAS presents unique challenges due to the need for coordination and cooperation among multiple agents, often in dynamic and uncertain environments [2]. The consensus problem in a leader–follower MAS refers to the process by which a group of agents (followers) coordinate their states to match that of a designated leader through local interactions and information exchange. In such types of problems, the leader acts as a reference providing a desired trajectory or state, while the followers adjust their states according to their neighbor states and, in some cases, directly from the leader. The main objective is to design control protocols that ensure all followers asymptotically track the leader's state despite challenges such as communication delays, switching topologies, nonlinear dynamics, or external disturbances. Leader–follower consensus algorithm proves helpful in applications like formation control of autonomous vehicles, cooperative robotics, sensor networks, and distributed decision-making systems, where achieving coordination with minimal communication overhead is crucial. Model Predictive Control has evolved as a smart control strategy for controlling complex systems, offering advantages such as constraint handling, disturbance rejection, and trajectory optimization [3]. The integration of learning techniques with the MPC has opened new avenues for enhancing the performance and adaptability of multi-agent systems. Integrating learning techniques with MPC offers the potential to improve the multi-agent systems' performance considerably, enabling adaptation to dynamic environments, learning from past experiences, and refining decision-making processes [4]. Driverless vehicle [5], power system management [6], and industrial control [7-9] are just a few of the control challenges that MPC has been employed to address. Neural network-based learning enhances MPC by adapting the system dynamics, cost functions, or constraint sets based on data [10]. The shallow neural network involves learning intricate functions, presenting inherent limitations when contrasted with deep architectures [11]. There are various architectures for shallow neural networks with distinguished characteristics.

Abbreviations & List of Symbols

NN	Neural Network
RNN	Recurrent Neural Network
MPC	Model Predictive Control
MAS	Multiagent system
SAC	Soft actor–critic
$\theta(t)$	Mode of communication topology in continuous time (t)
$\theta(k)$	Switching topology mode in discrete time at sample (k_{th}) instant
$x_i(k)$	Position variable of i_{th} follower agent in discrete form at (k_{th}) sample instant
$x_0(k)$	Position variable of the leader agent in a discrete form at (k_{th}) sample instant
$U_i(t)$	Input control variable in continuous time (t)
U(k)	Consensus control input in discretized form at sample (k_{th}) instant
P_p	Positive Definite Matrix
K_k	Optimal control input gain
$d_i(t)$	Disturbance Input
E(k)	Error between the i_{th} follower and the leader agent state variable
J(E(k))	Predicted cost function over a future horizon h
V(E(k))	Quadratic cost function associated with the error state

The FitNet neural network topology is the most basic feed-forward neural network; it has no feedback connections within or between layers and propagates activity unidirectionally from the input to the output stage. Feed-forward networks schematically stack perceptron layers on top of one another, allowing for the approximation of complex non-linear functions through the composition of simple linear transformations and non-linear activation functions [12]. Another neural network structure is the cascade-forward network, in which each layer receives input from all previous layers [13]. In the cascade network structure, unlike standard feed-forward networks, where only the first layer directly receives the input, every layer receives the input directly, facilitating the learning of more intricate and hierarchical representations of the input data [14]. Recurrent neural networks incorporate feedback connections, allowing them to model systems with memory and temporal dependencies [15]. The use of internal memory enables RNNs to retrieve data from past history, enabling a loop from the hidden node to itself [16]. The key contributions of this article are as follows:

- This work compares different neural network architectures within the NN-MPC framework for achieving optimal leader–follower consensus in multi-agent systems.
- As the prediction is done with MPC and optimization is carried out by the neural network-based architectures, the computational burden on the MPC is minimized, which helps in the improvement of system performance.
- The results are validated using mean square error (MSE) for all the structures and outcomes, with the best training function presented with a trade-off between fast response and least error performance.
- Results are compared with previously published findings on event-triggered control, and it has been shown that the results of RNN-based MPC follower agents achieve consensus faster than the previous work suggested.

2. Literature review

This section describes the recent trends in the control techniques for the leader-follower multi-agent system consensus problem. This includes event-triggered-based control strategy, MPC-based strategy, and learning based strategy. Finally, discussed the research gap in the present literature and the future scope for improvement in overcoming these research gaps.

2.1 Event-triggered-based strategy

Event-triggered control to reduce communication overhead and handle faults very effectively. Over the past few decades, event-triggered leader–follower consensus control strategy has gained significant importance for improving communication efficiency and robustness in multi-agent systems. Chen and Peng [17] proposed an event-triggered impulsive control scheme capable of handling packet loss in leader–follower networks, achieving the consensus using the Lyapunov stability theory, where sufficient criteria are identified to realize leader–follower quasi-consensus, ensuring reliable consensus under intermittent communication links. Similar work is explained by Zhi et al. [18], where a finite-time consensus control scheme employing an observer is proposed for second-order systems under velocity unknown, thereby achieving reduced communication updates through terminal sliding mode control. Also, Wu et al. [19] developed a fixed-time event-triggered consensus approach that guarantees convergence within a predetermined time despite delays and disturbances.

2.2 MPC-based strategy

The application of Model Predictive Control (MPC) to multi-agent systems is well established, as it effectively facilitates leader–follower consensus by predicting future trajectories, managing system constraints, and optimizing controls in real time. Kuriki et al. [20] explained how to combine consensus-based control with a decentralized MPC technique for multi-UAV formation, allowing for collision avoidance while preserving formation goals. In order to improve scalability and safety, Dubay and Pan [21] have extended this concept by proposing a distributed MPC framework for multiple quadcopters. In this framework, each agent computes its control action locally to reach consensus while avoiding collisions. By resetting the MPC optimization under specific circumstances, Saeednia and Khayatian [22] presented a reset MPC-based control technique for the MAS with fast convergence speed and robustness. Collectively, these research investigations demonstrate that MPC is ideal for applications like autonomous vehicles, cooperative robotics, and UAV swarms because it not only guarantees precise leader tracking but also offers a methodical approach to integrating safety, constraints, and optimal performance into leader–follower consensus control.

2.3 Learning based strategy

Learning-based controllers identify unknown dynamics while maintaining synchronization. Reinforcement learning frameworks effectively coordinate follower behavior without prior knowledge of the leader's state. Applying learning-based techniques to the leader-follower system has advanced significantly in recent times, especially for situations with non-linearities, uncertainties, and communication limitations. The aim of recent developments in learning-based control for multi-agent systems (MAS) is to reduce communication requirements, robustness to uncertainty, and model-free adaptation. For nonlinear MAS, Filiberto et al. [23] suggested a distributed control method based on Gaussian

Process regression, which ensures Lyapunov stability and allows for precise leader-follower agreement without the need for explicit system models. In order to enable accurate formation tracking with constrained residual errors, Yang et al. [24] presented a dynamical neural network-based control method that approximates unknown nonlinearities and disturbances. By combining radial basis function neural networks with fixed, relative, and switch triggering strategies, Wang et al. [25] built on communication efficiency to provide an adaptive event-triggered leader–follower control framework that prevents Zeno behavior and maintains consensus. Lastly, Li et al. [26] designed an adaptive distributed formation control method using a recurrent SAC reinforcement learning algorithm, enabling agents to achieve formation tracking in dynamic and uncertain environments with improved stability and adaptability. These works collectively reflect an ongoing shift toward data-driven, adaptive, and communication-effective solutions for the leader-follower consensus, enabling MAS to perform reliably in complex, uncertain, and resource-constrained environments. Table 1 presents a comparative analysis of event-triggered-based, MPC-based, and learning-based strategies. Various advantages and limitations of these strategies can be easily identified, and research gaps can be identified to further improve performance, such as fast convergence, reduced computational cost, stability, and constraint handling.

2.4 Research gap

From Table 1, it can be observed that even with great advancements in this research area, there are still considerable research gaps in applying event-triggered, MPC, and learning-based approaches to the leader-follower consensus problem. Designing asynchronous triggers for event-triggered control that ensure stability and performance regardless of packet failures, communication delays, and heterogeneous agent dynamics, while completely avoiding Zeno behavior, continues to be a challenge. Despite its effectiveness in managing restrictions and maximizing performance, the MPC-based approaches have limitations in terms of scalability for large networks, real-time viability on platforms with limited resources, and tolerance to nonlinearity and uncertainty. Although learning-based techniques like neural networks and reinforcement learning provide flexibility in unpredictable situations, they frequently lack formal stability guarantees, have significant data requirements, and present difficulties for safe real-world implementation. Moreover, integrated frameworks combining these methods remain underexplored, particularly in developing hybrid schemes that balance communication efficiency, scalability issues, robustness, and fast convergence for leader–follower consensus in dynamic and uncertain multi-agent environments. Table 2 shows a wide scope for improvement across various aspects of addressing the consensus problem in multi-agent systems, including model dependencies, constraint handling, and the computational burden imposed by the MPC strategy.

Table1. Comparison of related past work for the consensus problem of multi-agent systems

Strategy	Related Works	Key Features / Working Principle	Advantages	Limitations
2.1 Event-triggered based strategy	[17-19]	• Control actions and communications occur only when specific events or thresholds are triggered, reducing communication load. • Uses Lyapunov-based conditions for stability and convergence.	• Reduces unnecessary communication and energy consumption. • Handles packet loss and intermittent communication effectively. • Provides finite-time or fixed-time convergence.	• Requires careful design of triggering conditions to avoid Zeno behavior. • Performance may degrade with high network delays or noise. • Limited scalability for very large networks.
2.2 MPC-based strategy	[20-22]	• Uses Model Predictive Control to predict future trajectories and optimize control inputs under constraints. • Each agent solves an optimization problem locally to achieve consensus while respecting safety and collision avoidance.	• Systematic handling of constraints (safety, collision avoidance). • Provides optimal and coordinated performance. • Enables scalability and robustness for multi-agent systems.	• High computational cost due to online optimization. • Requires accurate system models and prediction horizons. • Limited applicability in real-time or highly dynamic environments with communication delays.
2.3 Learning-based strategy	[23-26]	• Employs data-driven or reinforcement learning techniques to handle unknown dynamics and uncertainties. • Uses neural networks or Gaussian Process regression for adaptive control without explicit models.	• Model-free and adaptive—suitable for uncertain and nonlinear systems. • Reduces dependency on accurate modeling and prior knowledge. • Capable of learning optimal coordination policies over time.	• Training requires extensive data and computation. • Stability and convergence proofs are complex. • May suffer from poor generalization or instability under unseen conditions. • Implementation in real-time may be challenging.

Table2. Improvement of AI-based MPC over traditional leader–follower consensus strategies

Aspect	Event-triggered Control	Classical MPC	Learning-based Control	AI-based MPC – Improvements
Model Dependence	Relies on known system dynamics and triggering conditions for stability.	Requires accurate mathematical models for prediction and optimization.	Model-free but lacks constraint interpretability.	AI learns or approximates system dynamics online, reducing dependency on exact models while retaining MPC's structure.
Adaptability	Limited adaptability to nonlinear or time-varying systems.	Struggles with strong nonlinearities unless extended (e.g., nonlinear MPC).	Highly adaptive but sometimes unstable.	AI enables online adaptation using reinforcement or continual learning, improving robustness to dynamic environments.
Computational Efficiency	Reduces communication but not computation.	Computationally heavy due to repeated optimization.	High training cost, sometimes offline only.	AI-based surrogates or neural approximators replace solvers, improving real-time efficiency.
Communication Efficiency	Excellent – triggers only on events.	No inherent communication saving.	Sometimes includes communication-efficient frameworks.	AI-based MPC can learn optimal communication schedules, combining event-triggering with adaptive learning.
Constraint Handling	Heuristic or limited.	Strong theoretical constraint handling.	Weak or implicit constraint management.	Maintains MPC's explicit constraint satisfaction while learning new constraints adaptively.
Convergence and Stability	Strong analytical guarantees under known models.	Stable if model is accurate.	Difficult to guarantee convergence formally.	Combines safe RL and Lyapunov-based design for provable stability under learned models.
Application Scope	Suitable for resource-constrained or periodic update systems.	Best for structured and well-modeled systems.	Effective for uncertain and nonlinear systems.	Unified approach—robust, adaptive, and safe; ideal for autonomous vehicles, UAVs, and robotics.

Also, communication failures encountered by event-triggered strategies, which work effectively only under excellent trigger conditions, make the control strategy unreliable. Lastly, the independently used learning-based strategies also lack convergence to the optimal solution and are ineffective at handling constraints in the problem. Among the control strategies used in the past literature, the NN-based MPC strategy (Figure 1) can be shown to be helpful in combining the advantages of learning-based and MPC strategies and eliminating limitations such as constraint handling, high computational cost, and model-based dependencies, as it allows operation in a model-free environment. NN- MPC also reduces the chances of communication failure faced by event-based strategies as it provides distributed control through proper communication between leader–follower and follower-follower with graph theory and switching topology using Markovian switching. Overall, these advantages help achieve consensus in the minimum time, as discussed in the results section.

3. Methodology

This section details the framework and analytical formulation used to develop and validate a NN-MPC system for a multi-agent robotic system. The section details the method for optimizing the consensus-based objective problem in multi-agent systems, where future states are predicted using model predictive control over the prediction horizon.

This integration of a neural network-based strategy with MPC shows remarkable improvement in achieving consensus for the multi-agent systems.

3.1 Preliminary

The problem formulation and mathematical analysis of the control strategy implemented are discussed in this section. This includes details about the consensus among followers and the leader. It also provides a brief on graph theory and switching topology, which provides the basic information on communication among agents.

Consensus tracking of multiple agents: This research work addressed the consensus problem in a leader-follower multi-agent system (MAS), where the goal is for all agents to achieve a desired state that is common to all through local interactions over a communication network [29–31]. For a network of N followers connected through a communication system, where the followers Xi(t), where i = 1, 2…. N aims to track the trajectory of a leader $X_0(t)$. Consensus tracking is achieved if, for any initial conditions, the states of all followers reach the consensus, $\text{Lt}_{t\to\infty} |X_i(t)-X_0(t)|=0$.

Graph theory: The basic structure used for communication among agents in the graph theory is modeled using a weighted graph G = {V, E}. The vertex set V = {v1, v2, …, vN} represents the placement of agents in the communication network, while the edge set $E \subseteq V\times V$, denotes the communication links. The weighted adjacency matrix is given by $A=[aij] \in N\times N$ of the graph G is defined such that aij > 0 if $(vj,vi) \in E$ and aij=0 otherwise. This graph connection is the

basic communication network of the leader-follower agents, where followers communicate with other follower agents and with the leader. A directed graph is used in this work.
Switching leader-follower connection: The proposed work of neural network-based MPC works with a switching topology of the follower agents with a leader connection. The switching is performed on the basis of the switching system and the Laplacian matrix as $l_{ii}(\theta(k)) = \sum_{j=1,j\neq i}^{N} a_{ij}(\theta(k))$ and $l_{ij}(\theta(k)) = -a_{ij}(\theta(k))$ for $j \neq i$. Here, $\theta(k)$ is the switching topology mode. This switching is followed by a probability matrix based on a Markovian chain, which is available in the simulation study section of this paper.

3.2 Problem formulation

Each agent is modeled with a first-order integrator system for the multi-agent system:

$$\dot{x}_i(t) = u_i(t) \quad (1)$$

where $x_i(t)$ and $u_i(t)$ are the state and control variables, respectively.
The consensus control input in discretized form is defined as:

$$u(k) = (L(\theta(k) * K_k)\, x(k) \quad (2)$$

where Kk represents the control gain and x(k) is the collective state variables of all followers.
To achieve the control objective given in equation (6), we consider an NN-MPC control strategy as illustrated in Fig. 1. The MPC layer predicts system behavior and computes a cost function, while the NN layer learns to optimize the cost by adjusting K_k using inputs x(k) and θ(k). In a more general setting, the agent dynamics can include non-linearity and unknown disturbance:

$$\dot{x}_i(t) = Ax_i(t) + Bu_i(t) + f(t, x_i(t)) + B_d\, d_i(t) \quad (3)$$

where $f(t, x_i(t))$ models nonlinear internal dynamics.
The proposed approach is simulated using MATLAB 2022, and the performance is analyzed for various NN training algorithms to achieve consensus in a fleet of quadrotors [32].

This section outlines the integration of prediction with MPC and optimization using a neural network for achieving consensus in the quadrotor fleet multi-agent system as proposed in reference [16]. Consider the first-order multi-agent system as given in equation (1), the objective of the follower agent is to track the leader state position. The consensus is shown as an error between the i_{th} follower and the leader agent state variable:

$$E(k) = x_i(k) - x_0(k) \quad (4)$$

where $x_i(k)$ and $x_0(k)$ are the state variables of the i_{th} follower and leader agent.
To guide the system towards the consensus, we define a quadratic cost function associated with the error state:

$$V(E(t), \theta(t)) = E^T(t) P_p\, E(t), \quad (5)$$

The predicted cost function is given by

$$J(E(k)) = \sum_{t=k+1}^{K+h+1} E_{k|k}\, [V(E(t), \theta(t))] \quad (6)$$

where J(E(k)) is the predicted cost function over a future horizon h.
To minimize this cost, the optimal control input gain is computed by:

$$K_k = \underset{K_k,\ldots K_{k+h}}{\arg\min}\, J(E(k)) \quad (7)$$

where K_k is the optimal control gain for the optimization function J(E(k)).
This optimization is handled by an NN-based learning mechanism that updates the gain Kk by training the network to reduce the predicted cost J(E(k)) over iterations. This combined NN-MPC framework enables the multi-agent system to reach consensus effectively in the dynamic network environment.

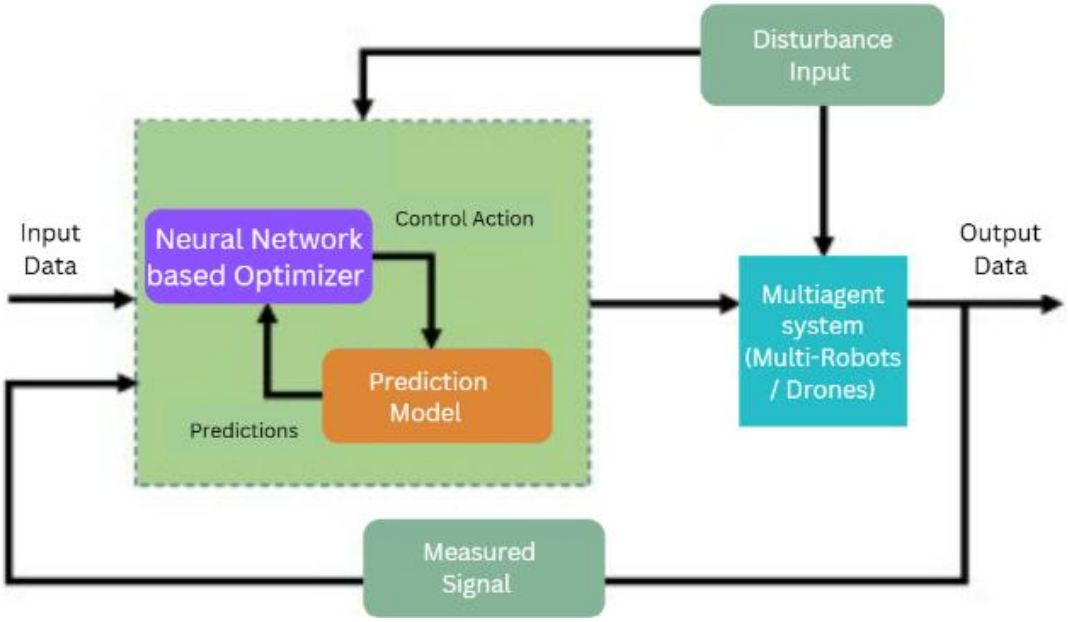


Figure 1. Block diagram of NN-Based MPC for multi-agent systems

3.3 Neural network architectures for NN training

This section describes various neural network architectures used for training are explained in this section. The basic training function used in each architecture is Levenberg-Marquardt. The L-M training algorithm uses a second-order approximation for the performance evaluation as a sum of squares instead of computing the actual Hessian matrix. Figure 2 shows the architecture of the Fit Net neural network. This is the basic network with one hidden layer [27]. It finds application in problems involving function approximation and regression analysis. The output equation of the Fit Net architecture is given by:

$$J = y = f(W_2 \cdot f(W_1.x + b_1) + b_2) \quad (8)$$

J = y is the optimized output from the trained network, where the network is trained by adjusting the weight matrices W_1 and W_2 during the training. Here, the sigmoid activation function f(.) is used for training the weighted inputs, whereas b_1 and b_2 are the bias vectors. The node h is a neuron layer between input x and output y where input weights are adjusted for the network during training. Once the training is completed, the inputs are applied to the network and an optimal solution y is obtained, which is the minimum of errors of the state matrices between the leader and follower states. The same is applied as control gain K_k according to equation (7) to the control gain matrix given by equation (2), which further updates the state given by the system model equation (1) at each iteration, and the process repeats until consensus is achieved. Figure 3 shows the architecture of a feedforward neural network. This network includes two or more hidden layers [27], useful for learning complex features. The output equation of the feedforward network architecture is given by:

$$J = y = f(W_3 \cdot f(W_2 \cdot f(W_1.x + b_1) + b_2) + b_3) \quad (9)$$

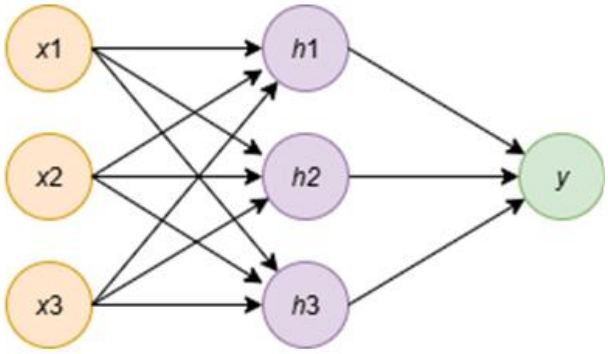


Figure 2. Architecture of the Fit Net neural network training set

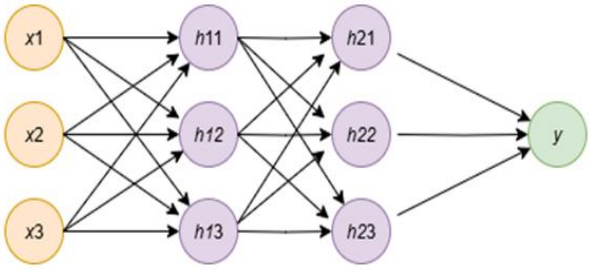


Figure 3. Architecture Of Feedforward Neural Network Training Set

This architecture is similar to the Fitnet, except it can have more hidden layers, which can improve the network's accuracy, but at the same time make the network more complex. Figure 4 shows the architecture of a recurrent neural network. A Cascade-forward network allows connections from all layers in parallel, including input to the output directly [28], enhancing learning flexibility. Equation (10) gives the output of the cascade network configuration.

$$J = y = f(W_3 \cdot f(W_2 \cdot f(W_1.x + b_1) + W_4.x + b_2) + b_3 \quad (10)$$

The uniqueness of this configuration is that it includes a direct connection between the inputs and outputs, also during the network learning. The same activation function is used in this architecture as is used in the Fitnet and feedforward network configuration. The weights are adjusted during training and learning of the network. Once the network is trained, the optimal solution is obtained and applied to the control gain matrix Kk, which further updates the system. Process repeats until consensus. Figure 5 shows the RNNs that include loops to retain memory over time steps [16, 28]. They are ideal for sequence-based tasks. The output equation for the recurrent neural network is given by Eq (11) and Eq (12).

$$h_t = f(W_x x_t + W_h h_{t-1} + b) \quad (11)$$

$$J = y_t = f(W_y h_t + c) \quad (12)$$

The above output $J = y_t$ is the optimized output from the trained recurrent. Sigmoid activation function f(.) is used for tuning the input weights for RNNs.

The hidden layer has a loop between h_t and h_{t-1}, where hidden weights W_h are adjusted for the previous hidden state during training. Once the training is completed, the inputs are applied to the network, and the optimal solution y is obtained. The input x(t) to the neural network represented by Eq (13) is as follows:

$$x(t) = [1\ K^{11} \ldots K^{1n} \ldots K^{m1} \ldots K^{pq}]' \quad (13)$$

The algorithm used for the optimization of the objective cost function is shown in Figure 6, which explains how the optimization is achieved with the various neural network architectures for NN-based predictive control. At the beginning of the simulation, the control gain K_K is assigned an initial value of zero to ensure a safe starting point for the learning process.

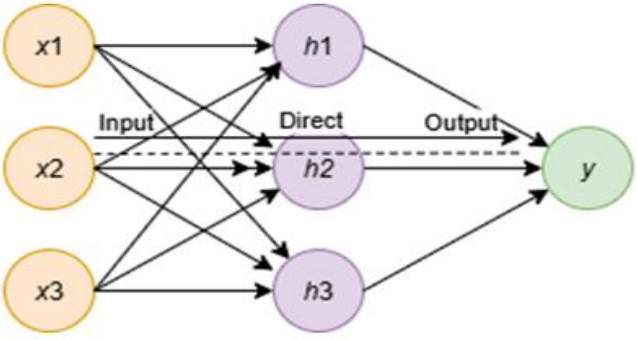


Figure 4. Architecture of cascade neural network training set

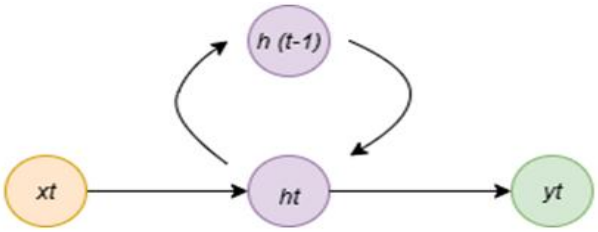


Figure 5. Architecture of the recurrent neural network training set

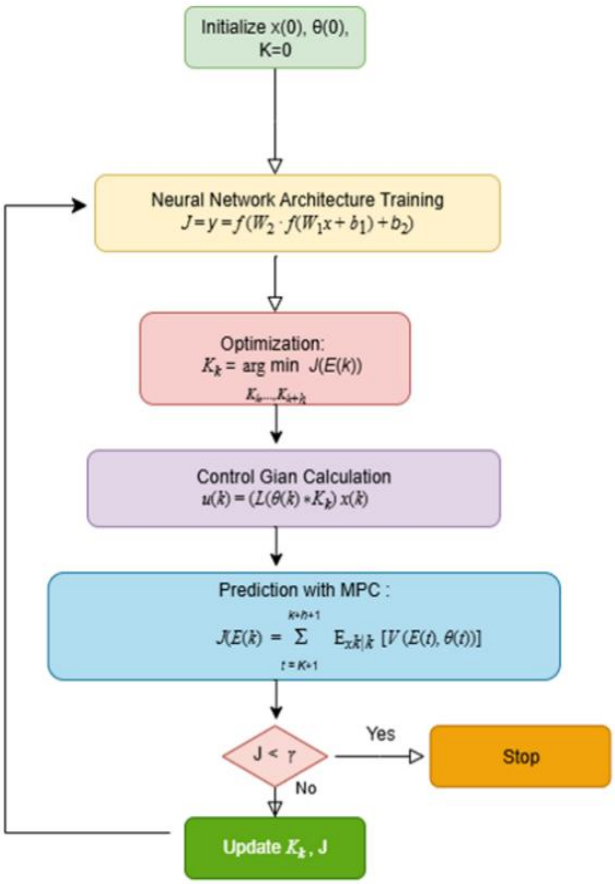


Figure 6. Flow chart of the algorithm for the NN-MPC-based multi-agent system

Figure 6 illustrates the algorithm for the NN-based MPC for a leader-follower system with various neural network configurations. The network training starts with initialization of the values x(k), θ (k), and K_k. The neural network is trained for different cases based on Eqs (8-12). Then, the initial optimization is done using the trained network. The value of the K_k is used for the control gain u(k) calculation as given by equation (2). Then, the optimal cost is predicted using equations (4-6) with MPC, based on the future values of errors in the follower and leader states. After every iteration, the criteria for optimization are checked J< γ, if the condition is satisfactory, then the iteration stops; otherwise, it continues with updating the values of J and K.

4. Results and discussion

4.1 Simulation study

The system model used for the simulation purpose is the same as that used in the reference [32], and the data is taken from [16] and [29]. For the system model given by equation (3), f (t, xi(t)) = 0.01 sin(xi(t)), and the initial values are the same as those considered in [16]. A, B, and B_d matrices are given as:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -0.5 \end{bmatrix},\ B = \begin{bmatrix} 0.8 \\ 1.2 \end{bmatrix},\ B_d = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

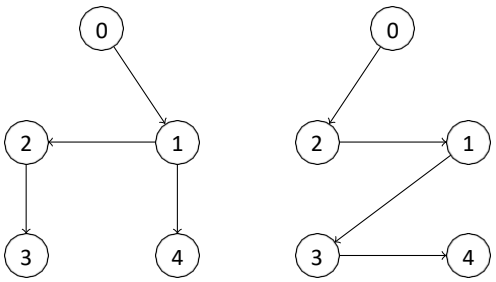
The Laplacian matrices are considered as follows,

$$L(1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix},$$

$$L(2) = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

The probability matrix is given by: $\pi = \begin{bmatrix} 0.95 & 0.05 \\ 0.02 & 0.98 \end{bmatrix}$

The directed graph is used in this work as shown in Figure 7. and switching of the graph is carried out between L(1) and L(2) based on the probability matrix π. Sampling time for the discretization is Ts = 0.01.



(a) L(1) Directed Graph (b) L(1) Directed Graph

Figure 7. Directed graph topology

The simulation is carried out using N = 10 neurons. To achieve optimal results for all the cases, the prediction horizon is taken as 100. This horizon is deemed sufficient for reaching the optimal point. Convergence is obtained for λ = 0.01. For validation, the chosen performance metric, the MSE is defined as:

$$MSE = \frac{1}{k_f}\sum_{k=0}^{k_f} |e_{ri}(k)|^2 \quad (14)$$

Where, $e_{ri}(k)$ denotes the error values $e_i(k)$.

Figure 8 reflects that for the NN-MPC with the Fit Net architecture. All the agents achieve consensus at 3 sec in 'position variable' while it reaches consensus after 4 sec for the 'velocity variable'. Figure 9 also reflects that NN-MPC has the feed-forward network configuration. It reflects that all the agents achieve consensus at 3.5 sec in the 'position variable' and 4 seconds in the 'velocity variable' respectively. The agents achieve consensus 16.6% faster with the Fit Net architecture-based NN in comparison to the feedforward-based NN architecture for multi-agent systems.

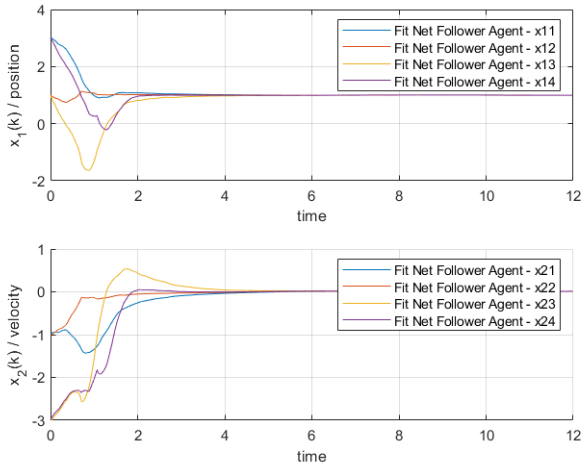


Figure 8. Position and velocity variable response for the followers with the Fit Net architecture

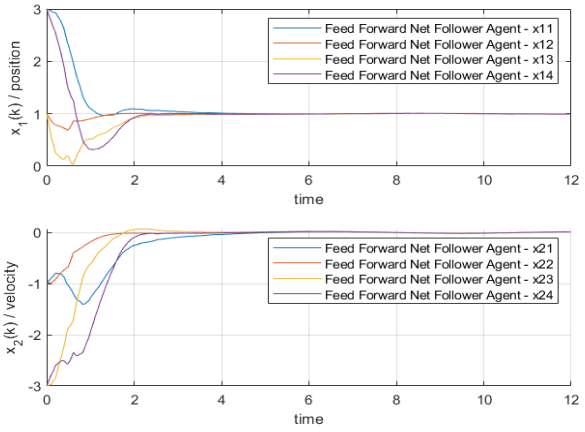


Figure 9. Position and velocity variable response for the followers with a feed-forward net architecture

Figure 10 reflects RNN-MPC with the recurrent neural network architecture. It reflects that all the agents reach consensus at 2 sec for the 'position state' while just after 3 sec for the 'velocity state'. NN-MPC with the cascade network architecture is shown in Figure 11. It reflects that all the agents reach the leader position at 4 sec for 'position state' and at 5 sec for the 'velocity state'. With RNN-MPC, control agents can achieve consensus almost 100 % faster in comparison to the cascade-forward-based NN architecture for multi-agent systems.

Table 3 shows the comparison of mean squared errors for the 'position and velocity variable' of various neural network architectures for achieving consensus of follower agents. It can be observed that the least error is found with three cases as Fit Net, RNN, and CFN for 'position state' and 'velocity state'.

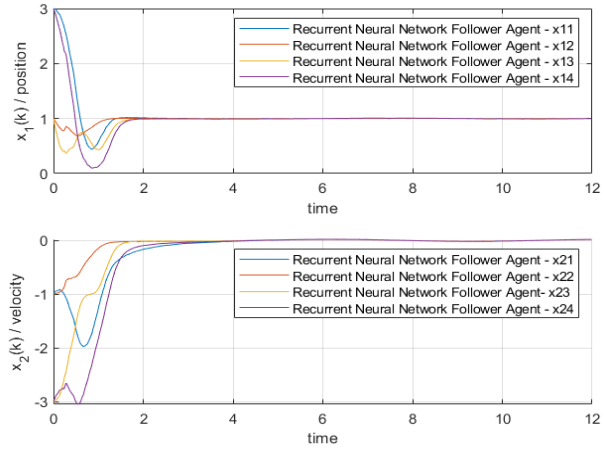


Figure 10. Position and velocity variable response for the followers with a recurrent neural network architecture

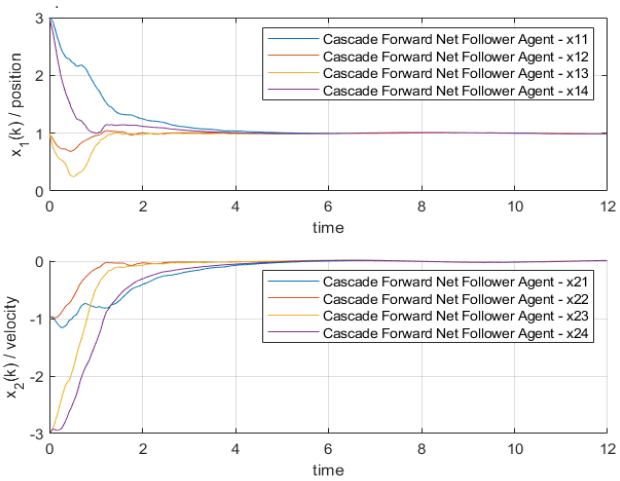


Figure 11. Position and velocity variable response for the followers with a cascade forward net architecture

Table3. Mean squared error comparison of various neural network architectures for NN-MPC based multi-agent system

(i_{th} agent, r_{th} state)	FITNET	Feed forward (FFN)	Cascade Forward Net (CFN)	Recurrent Neural Network (RNN)	Gao et al.	Least MSE
(1,1)	0.0866	0.0947	0.1028	0.0671	0.3395	RNN
(1,2)	0.0967	0.0943	0.0740	0.1278	0.0781	CFN
(2,1)	0.0010	0.0019	0.0022	0.0022	0.0588	FITNET
(2,2)	0.0204	0.0230	0.0237	0.0232	0.0194	Gao et al.
(3,1)	0.1972	0.0305	0.0149	0.0134	0.0824	RNN
(3,2)	0.3387	0.2198	0.1978	0.1948	0.1029	Gao et al.
(4,1)	0.0826	0.0664	0.0377	0.0644	0.4142	CFN
(4,2)	0.3825	0.3663	0.3324	0.4014	0.1823	CFN

MSE is least with RNN architecture for the 'position state' of follower '1' and both 'position and velocity state' of follower '3'. Whereas, MSE is least for both 'position state' and 'velocity state' error for the followers '2' if a Fit Net architecture is used, a neural network structure. In other cases, like 'Position state' of follower '4' and 'velocity state' of follower '1' and '4', the MSE is minimum with the CNN architecture. It can be observed that no single architecture can give the least MSE for all the follower agents possible, but a compromise can be possible for a configuration that can give a better response (fast response) with the least MSE required for the consensus of the agents. From the above observations, we can say that RNN-MPC reaches the consensus in the minimum time, thereby showing the least MSE for the 'position state' of followers '1' and '3'. Results of Gao et al. show minimum MSE for the 'velocity state' of followers '2' and '3'. However, there is no single training configuration that provides the least MSE for all agents for the 'position state' and 'velocity state'.

4.2 Comparison to the related work

If the comparison is made with the previous work for achieving consensus of follower agents presented by Gao et al. [32], it can be said that the RNN-MPC gives desired results in minimum time thereby follower agents reaching to the consensus in the minimum time at 2 sec thereby showing much better performance in achieving consensus than the event triggered based strategy where consensus is achieved in more than 10 seconds. However, there can be more chances of improvement as far as MSE is concerned for the position and velocity variables of the followers. The proposed NN-MPC approach differs from conventional MPC and existing learning-based MPC strategies by incorporating neural networks as FITNET, feedforwardnet, cascadenetwork, and recurrent neural network within the control framework. Traditional MPC predicts the future states and optimizes the objective cost by itself, which limits its performance in nonlinear or uncertain environments. In contrast, the proposed NN-MPC takes the optimization burden of the MPC with various neural networks—such as feedforward, recurrent, Fitnet, and cascade architectures—that learn in the complex system dynamics and inter-agent interactions. This integration allows the controller to predict future and optimize the error between the state trajectories of the leader-follower agents more accurately and adapt to changing conditions in real time. In the future, the existing learning-based MPCs that typically utilize a single neural network trained offline can be replaced by the proposed framework systematically, which can compare multiple neural architectures to determine the most effective configuration for achieving consensus in leader–follower multi-agent systems. Consequently, the NN-MPC enhances adaptability, robustness, and coordination performance while preserving the optimization and constraint-handling advantages of the MPC structure.

5. Conclusion

Conclusions drawn from the above results suggest that the RNN-MPC-based configuration of the neural network gives a fast response to the leader-follower system in achieving consensus in minimum time over other neural network-based architectures like FITNET, feedforward network, and cascade network. Also, there is a considerable reduction in the MSE for the 'position variable' of followers '1 and 3' that validates the effectiveness of the recurrent neural network-based training network. However, there is no architecture found that provides the least MSE for all agents for the 'position state' and 'velocity state'. Future research efforts should be focused on finding such a learning-based MPC that can provide fast response as well as the least MSE for most of the agents to achieve the consensus of the multi-agent quadrotor fleet system.

Ethical issue

The authors are aware of and comply with best practices in publication ethics, specifically regarding authorship (avoidance of guest authorship), dual submission, manipulation of figures, competing interests, and compliance with research ethics policies. The authors adhere to publication requirements that the submitted work is original and has not been published elsewhere.

Data availability statement

The manuscript contains all the data. However, more data will be available upon request from the authors.

Conflict of interest

The authors declare no potential conflict of interest.

References

[1] K. M. Khalil, M. Abdel-Aziz, T. T. Nazmy and A. B. M. Salem, "Machine Learning Algorithms for Multi-Agent Systems," Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication, pp. 1–5, Nov. 2015. https://doi.org/10.1145/2816839.2816925

[2] L. Canese, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re and S. Spanò, "Multi-Agent Reinforcement Learning: A Review of Challenges and Applications," Applied Sciences, vol. 11, no. 11, p. 4948, 2021.https://doi.org/10.3390/app11114948

[3] D. Q. Mayne, "Model Predictive Control: Recent Developments and Future Promise," Automatica, vol. 50, no. 12, pp. 2967–2986, 2014. https://doi.org/10.1016/j.automatica.2014.10.128

[4] A. Norouzi, H. Heidarifar, H. Borhan, M. Shahbakhti and C. R. Koch, "Integrating Machine Learning and Model Predictive Control for Automotive Applications: A Review and Future Directions," Engineering Applications of Artificial Intelligence, vol. 120, p. 105878, 2023. https://doi.org/10.1016/j.engappai.2023.105878

[5] B. Yi, P. Bender, F. Bonarens and C. Stiller, "Model Predictive Trajectory Planning for Automated Driving," IEEE Transactions on Intelligent Vehicles, vol. 4, no. 1, pp. 24–38, 2018. https://doi.org/10.1109/TIV.2018.2886683

[6] P. Kumar, M. Karamta and A. Markana, "Dynamic State Estimation for Multi-Machine Power System Using WLS and EKF: A Comparative Study," Proceedings of the 2019 IEEE 16th India Council International Conference (INDICON), pp. 1–4, Dec. 2019.https://doi.org/10.1109/INDICON47234.2019.9030371

[7] M. Anilkumar, N. Padhiyar and K. Moudgalya, "Multi-Objective Prioritized Control of a Semi-Batch Process with Multiple Feed and Multiple Products Using Economic MPC," Proceedings of the 2018 Indian Control Conference (ICC), pp. 264–269, Jan. 2018. https://doi.org/10.1109/INDIANCC.2018.8307989

[8] A. Markana, N. Padhiyar and K. Moudgalya, "Multi-Criterion Control of a Bioprocess in Fed-Batch Reactor Using EKF Based Economic Model Predictive Control," Chemical Engineering Research and Design, vol. 136, pp. 282–294, 2018. https://doi.org/10.1016/j.cherd.2018.05.032

[9] J. M. Maciejowski, Predictive Control: With Constraints. Harlow, U.K.: Prentice Hall, 2002. ISBN: 0201398230.

[10] A. Ashoori, B. Moshiri, A. Khaki-Sedigh and M. R. Bakhtiari, "Optimal Control of a Nonlinear Fed-Batch Fermentation Process Using Model Predictive Approach," Journal of Process Control, vol. 19, no. 7, pp. 1162–1173, 2009. https://doi.org/10.1016/j.jprocont.2009.03.006

[11] Y. Bengio, "On the Challenge of Learning Complex Functions," Progress in Brain Research, vol. 165, pp. 521–534, 2007. https://doi.org/10.1016/S0079-6123(06)65033-4

[12] H. C. Myung and Z. Z. Bien, "Design of the Fuzzy Multiobjective Controller Based on the Eligibility Method," International Journal of Intelligent Systems, vol. 18, no. 5, pp. 509–528, 2003. https://doi.org/10.1002/int.10101

[13] L. Dubreuil-Vall, G. Ruffini and J. A. Camprodon, "Deep Learning Convolutional Neural Networks Discriminate Adult ADHD from Healthy Individuals on the Basis of Event-Related Spectral EEG," Frontiers in Neuroscience, vol. 14, p. 251, 2020. https://doi.org/10.3389/fnins.2020.00251

[14] Y. Zhu, J. Wang, H. Li, C. Liu and W. M. Grill, "Adaptive Parameter Modulation of Deep Brain Stimulation Based on Improved Supervisory Algorithm," Frontiers in Neuroscience, vol. 15, p. 750806, 2021. https://doi.org/10.3389/fnins.2021.750806

[15] S. Seyedzadeh, F. P. Rahimian, I. Glesk and M. Roper, "Machine Learning for Estimation of Building Energy Consumption and Performance: A Review," Visualization in Engineering, vol. 6, no. 1, p. 5, 2018. https://doi.org/10.1186/s40327-018-0064-7

[16] Chaubey, P., Markana, A., Vyas, D.R. . "RNN-Based Model Predictive Control of Multi-agent System Using Switching Topologies." Data Science and Applications. ICDSA 2023. Proceedings in Lecture Notes in Networks and Systems, vol 821. Springer, Singapore, pp157-168 (February 2024). https://doi.org/10.1007/978-981-99-7814-4_13

[17] R. Chen and S. Peng, "Leader-Follower Quasi-Consensus of Multi-Agent Systems with Packet Loss Using Event-Triggered Impulsive Control," Mathematics, vol. 11, no. 13, p. 2969, 2023. https://doi.org/10.3390/math11132969

[18] Y. Zhi, Z. Zhao and M. Qi, "Event-Triggered Finite-Time Consensus Control of Leader–Follower Multi-Agent Systems with Unknown Velocities," Transactions of the Institute of Measurement and Control, vol. 45, no. 13, pp. 2515–2525, 2023. https://doi.org/0.1177/01423312221140619

[19] Y. Wu, J. Ma, X. Chen, J. Sun and F. Zhao, "Fixed-Time Leader-Follower Consensus for Multi-Agent Systems Under Event-Triggered Mechanism," in Proceedings of the Chinese Conference on Swarm Intelligence and Cooperative Control, pp. 275–284, Nov. 2023. https://doi.org/10.1007/978-981-97-3340-8_25

[20] Y. Kuriki and T. Namerikawa, "Formation Control with Collision Avoidance for a Multi-UAV System Using Decentralized MPC and Consensus-Based Control," SICE Journal of Control, Measurement, and System Integration, vol. 8, no. 4, pp. 285–294, 2015 https://doi.org/10.9746/jcmsi.8.285

[21] S. Dubay and Y. J. Pan, "Distributed MPC Based Collision Avoidance Approach for Consensus of Multiple Quadcopters," Proceedings of the 2018 IEEE 14th International Conference on Control and Automation (ICCA), pp. 155–160, June 2018. https://doi.org/10.1109/ICCA.2018.8444273

[22] N. Saeednia and A. Khayatian, "Reset MPC-Based Control for Consensus of Multiagent Systems," IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2024. , https://doi.org/10.1109/TSMC.2024.3510092

[23] F. Muñoz, J. M. Valdovinos, J. S. Cervantes-Rojas, S. S. Cruz and A. M. Santana, "Leader–Follower Consensus Control for a Class of Nonlinear Multi-Agent Systems Using Dynamical Neural Networks," Neurocomputing, vol. 561, p. 126888, 2023. https://doi.org/10.1016/j.neucom.2023.126888

[24] Z. Yang, S. Sosnowski, Q. Liu, J. Jiao, A. Lederer and S. Hirche, "Distributed Learning Consensus Control for Unknown Nonlinear Multi-Agent Systems Based on Gaussian Processes," Proceedings of the 2021 60th IEEE Conference on Decision and Control (CDC), pp. 4406–4411, 2021 https://doi.org/0.1109/CDC45484.2021.9683522

[25] Z. Wang, Y. Gao, A. I. Rikos, N. Pang and Y. Ji, "Fixed-Relative-Switched Threshold Strategies for Consensus Tracking Control of Nonlinear Multiagent Systems," Proceedings of the 2025 IEEE 19th International Conference on Control & Automation (ICCA), pp. 899–905, June 2025. https://doi.org/10.48550/arXiv.2411.19571

[26] M. Li, H. Liu, F. Xie and H. Huang, "Adaptive Distributed Control for Leader–Follower Formation Based on a Recurrent SAC Algorithm," Electronics, vol. 13, no. 17, p. 3513, 2024. https://doi.org/10.3390/electronics13173513

[27] K. G. Dastidar, O. Caelen and M. Granitzer, "Machine Learning Methods for Credit Card Fraud Detection: A Survey," IEEE Access, 2024. https://doi.org/10.1109/ACCESS.2024.3487298

[28] K. Aitken and S. Mihalas, "Neural Population Dynamics of Computing with Synaptic Modulations," eLife, vol. 12, p. e83035, 2023. https://doi.org/10.7554/eLife.83035

[29] B. R. Floriano, A. N. Vargas, J. Y. Ishihara and H. C. Ferreira, "Neural-Network-Based Model Predictive Control for Consensus of Nonlinear Systems," Engineering Applications of Artificial Intelligence, vol. 116, p. 105327, 2022. https://doi.org/10.1016/j.engappai.2022.105327

[30] W. Ren, R. W. Beard and E. M. Atkins, "Information Consensus in Multivehicle Cooperative Control," IEEE

Control Systems Magazine, vol. 27, no. 2, pp. 71–82, 2007. https://doi.org/10.1109/MCS.2007.338264

[31] T. Zhang and Y. U. Hui, "Average Consensus in Networks of Multi-Agent with Multiple Time-Varying Delays," International Journal of Communications, Network and System Sciences, vol. 3, no. 2, pp. 196–203, 2010. http://dx.doi.org/10.4236/ijcns.2010.32028

[32] J. Gao, J. Li, H. Pan, Z. Wu, X. Yin and H. Wang, "Consensus via Event-Triggered Strategy of Nonlinear Multi-Agent Systems with Markovian Switching Topologies," ISA Transactions, vol. 104, pp. 122–129, 2020. https://doi.org/10.1016/j.isatra.2019.11.013