Neurocomputing 617 (2025) 129005

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Enhancing cooperative multi-agent reinforcement learning through the integration of R-STDP and federated learning

Mohammad Tayefe Ramezanlou [a,*], Howard Schwartz [a], Ioannis Lambadaris [a], Michel Barbeau [b]

[a] Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada
[b] School of Computer Science, Carleton University, Ottawa, Canada

ARTICLE INFO

Communicated by C. Gao

Keywords:
Spiking neural network
STDP
Federated learning
Consensus flying
Leader–follower flocking

ABSTRACT

This paper introduces a novel approach to enhance the stability and efficiency of R-STDP in the context of federated learning. The primary objective is to stabilize the unbounded growth of R-STDP and make it more responsive to real-time changes. The methodology involves integrating R-STDP with Spiking Neural Networks and employing the norm of the neural network model for adjusting weighted aggregation in federated learning systems. The proposed method incorporates a mechanism where weights decay over time, depending on the duration since the agent last published its model. Additionally, the sampling time is dynamically adjusted based on the Euclidean norm, which measures the distance between the weight matrices of the agents and the server. The results demonstrate that the proposed event-triggered federated learning method significantly enhances learning speed and performance. At the same time, the dynamic aggregation interval efficiently reduces communication between the agents and the central server, especially after model convergence. This research presents a significant advancement in federated learning and offers a more stable, responsive, and efficient learning process.

1. Introduction

The consensus problem in flying multi-agent systems, commonly called the “flocking” or “swarming” challenge, is fundamental to aerial robotics. It involves coordinating and controlling multiple agents to ensure cooperative behavior, avoid collisions, and align towards a shared goal. Such coordination is crucial for applications ranging from coordinated surveillance to communication, logistics, and infrastructure monitoring.

Formation control within these systems, especially cost-constrained communication, faces significant challenges. These include managing limited resources, overcoming communication constraints, and ensuring system scalability and robustness. Various studies have addressed these issues, demonstrating innovative solutions and methodologies to enhance adaptability and resilience by leveraging local sensing and communication capabilities, even on low-cost platforms [1].

Security in multi-agent systems is also a critical concern, particularly in the face of cyber threats. Adaptive mechanisms that tune communication link weights to maintain secure consensus have offered resilience in adversarial conditions [2]. Additionally, hierarchical control mechanisms, such as leader-follower dynamics, are essential for specific applications, including aerial operations. Recent advancements include finite-time control protocols that ensure stability and rapid consensus in leader-follower setups [3] and innovative path-guided control strategies that function effectively in uncertain environments [4].

Moreover, decision-making within these systems has seen enhancements through entropy-based consensus methods that facilitate more efficient cooperation by employing swarm intelligence principles for local negotiation and preference updating [5]. These methodologies promise faster convergence and scalability, which are crucial for diverse operational settings. Overall, the ongoing research in multi-agent systems continues to address the dual challenges of robust formation control and secure, efficient consensus amidst evolving operational demands and external threats.

1.1. Decentralized learning through FL

FL is an emerging paradigm in machine learning that allows for decentralized training of models across multiple agents without centralizing data [6]. Using this approach, robots can benefit from shared experiences while preserving data privacy and reducing communication overheads [7,8].

The challenge of optimizing communication efficiency in FL is addressed in [8], where a hierarchical approach is introduced, leveraging

* Corresponding author.
E-mail address: MohammadTayefeRamez@cmail.carleton.ca (M.T. Ramezanlou).

https://doi.org/10.1016/j.neucom.2024.129005
Received 30 April 2024; Received in revised form 19 September 2024; Accepted 21 November 2024
Available online 29 November 2024

adaptive staleness control. This method emphasizes that system-level and data-level heterogeneity in FL must be considered. The complexities of wireless network constraints in FL are explored in another study, where the focus is placed on optimizing the convergence time. This research indicates that the wireless environment introduces a layer of complexity to the FL framework [9].

The practical applicability of FL in real-world scenarios is demonstrated in [10], which focuses on hierarchical trajectory planning for narrow-space automated parking. This study highlights the vast potential of FL in automation and robotics. An exciting direction in which neuromorphic learning and FL converge is presented in [11], suggesting future trends where bio-inspired computing and FL might merge. A holistic view of the advancements, challenges, and future directions in FL is provided in a comprehensive survey, making it clear that FL is poised to reshape the landscape of machine learning [12].

FedFa presents a novel FL algorithm that utilizes a double momentum gradient approach and a specialized weighting strategy to enhance the fairness and accuracy of model training across distributed networks [13]. By incorporating considerations of accuracy and participation frequency in the weighting of client updates, FedFa significantly improves upon the stability of convergence and fairness in the learning process compared to traditional methods. These advancements are demonstrated through rigid practical testing on synthetic and real datasets.

1.2. Decentralized reinforcement learning: RL meets FL

Real-time processes are essential for decision-making and adaptability in dynamic environments. The Reinforcement Learning (RL) algorithms offer a promising approach to address these challenges by enabling agents to learn optimal policies through interactions with their environment, enabling them to adapt to dynamic scenarios and uncertainties. Specifically, they allow agents to autonomously learn from their experiences, making them well-suited for tasks that require decentralized decision-making and adaptability to unforeseen situations [14].

Evolutionary dynamics provide a novel perspective on agent learning in MAS. For instance, [15] explores the application of replicator dynamics from evolutionary game theory to Q-learning, offering insights into the exploration-exploitation mechanisms in MAS, which could inform more effective consensus strategies in aerial robotics.

Further complicating the implementation of RL in MAS, new methodologies suggest deterministic limits to temporal difference learning, offering potential improvements in learning stability and response to environmental dynamics in MAS [16].

In addressing the complexities of MAS, recent advances have modeled the regret minimization dynamics across large populations, offering a robust framework to better understand and predict agent behavior in collaborative aerial tasks [17].

In addition to RL, incorporating FL offers a decentralized approach, further advancing learning and adaptability in dynamic and distributed environments. The integration of FL with RL presents opportunities for drones to learn and update their policies in a distributed manner collaboratively, leveraging the collective intelligence of the swarm [18]. The advent of the FL offers a decentralized training paradigm, allowing agents to learn collaboratively while keeping their data localized.

The FedDSR model combines FL and Deep Reinforcement Learning (DRL) to optimize daily schedule recommendations while maintaining user privacy, demonstrating superior performance in dynamic and privacy-sensitive environments through integrating curriculum learning and a novel similarity aggregation algorithm [19].

The opportunity and challenge presented by IoT devices in FL are highlighted in [18], where edge computing and deep reinforcement learning are combined for traffic management in IoT. The evolving nature of FL applications, branching out from traditional use cases, is demonstrated by this work. Regarding the aggregation methods in FL, robust algorithms that provide guarantees against adversarial attacks are introduced, underscoring the security aspect of FL [20].

By integrating RL with FL, researchers envision a new era where flying agents learn and adapt in real-time, making consensus problems more manageable. Fusing these advanced learning techniques can revolutionize consensus mechanisms in flying multi-agent systems. The application of these integrated learning methods in multi-agent systems presents various challenges and opportunities, particularly in formation control and security aspects.

1.3. Spiking neural networks in MAS

According to recent research, event-triggering mechanisms in multi-agent systems can improve communication and computation. This approach operates with predetermined communication patterns, reducing messages to save resources while ensuring stability and convergence [21]. While optimizing communication in multi-agent systems, emerging methods like SNN offer groundbreaking approaches to robotic learning and control.

SNNs have gained popularity in robotics due to their ability to replicate the structure and functioning of networks. One crucial characteristic of SNNs is their capacity to encode and handle information through spikes, which has been demonstrated as a resource energy-saving approach [22,23]. The versatility of SNNs is further expanded through their ability to learn and adapt, which is particularly beneficial in complex tasks like movement planning and nonlinear system control.

One of the remarkable features of SNNs is their ability to learn and adapt. Here, Reward-modulated Spike-Timing-Dependent Plasticity (R-STDP) plays a key role. R-STDP is a bio-inspired learning rule based on the relative timing of pre- and post-synaptic spikes. Several studies have elucidated the constraints on Hebbian and R-STDP learned weights in spiking neurons, revealing the underlying mechanisms that make this learning paradigm so effective [24]. Furthermore, research has demonstrated that networks trained with local rules, such as R-STDP, can exhibit continuous learning, showcasing their potential in lifelong learning scenarios [25]. To fully utilize the potential of SNNs, developing hybrid models and exploring various training methodologies are crucial in enhancing their adaptability and efficiency.

In robotics, the application of SNNs has shown promise in various tasks, including movement planning within confined operational spaces. Such applications leverage the temporal dynamics of spiking neurons to achieve collision-free motion, demonstrating the capability of SNNs to handle complex spatial–temporal challenges [26]. Beyond movement planning, SNNs have also been employed for nonlinear systems control, providing a robust and adaptive control mechanism [27].

The evolution of SNN has seen the emergence of models that aim to blend the advantages of machine-inspired approaches. These hybrid systems have enhanced the adaptability of SNNs, making them more suitable for environments encountered in robotics [28]. The advancements in classification capabilities offered by integrate and fire models have also expanded the range of SNN applications in robotics [29]. Notably, Deep Spiking Q Networks, which are trained directly, have shown performance in tasks highlighting the potential of integrating deep learning techniques with SNNs to achieve human-level control [30]. Exploring direct and indirect training methodologies for SNNs is vital, particularly in applications such as autonomous vehicle control, where end-to-end learning is essential.

Direct and indirect training methodologies for SNNs have been extensively explored for achieving end-to-end control of vehicles in tasks like lane keeping [31]. Furthermore, incorporating meta neurons into SNNs has further improved their effectiveness in spatial learning tasks [32]. The adaptability of SNNs, enhanced through reinforcement and evolutionary learning, opens new avenues in robotics, particularly motor control and changing operational needs. The exploration of learning dynamics in neural networks, particularly in developing neural units that can learn rules and robot dynamics, represents a significant advancement in the capabilities of SNNs.

In line with research on learning dynamics, there is a growing interest in developing networks consisting of neural units. When combined together, these units possess the ability to learn both learning rules and spiking dynamics, thereby enhancing the capabilities exhibited by SNN [33].

The application of reinforcement and evolutionary learning to train SNNs for motor control has opened new avenues in the field of robotics. Such training methods use SNNs' adaptability for the changing needs of robots [34–37]. Integrating FL with SNNs presents the synergy between decentralized training strategies and advanced neural network architectures, offering collaborative and privacy-preserving learning opportunities.

Another frontier in applying SNNs in robotics is integrating FL. The FL has been combined with SNNs to achieve collaborative learning across multiple agents while harnessing the efficiency of SNNs in distributed settings [38].

Several challenges emerge in exploring integrating SNNs with FL. The transmission of SNN-specific parameters, such as spike timings, introduces considerable communication overhead in the FL setup [39]. Additionally, aggregating SNN models from diverse devices in a federated context is non-trivial, often leading to challenges in achieving effective global learning. The unique dynamics of spiking neurons, coupled with the distributed nature of FL, can also result in training instabilities. Lastly, the inherent complexity and potential size of SNNs raise concerns about memory requirements and the feasibility of model aggregation in distributed scenarios [40].

The provided literature has set the stage by exploring integrating algorithms such as SNN and FL in MAS, underscoring their key roles in enhancing robotics' capabilities. These methods facilitate efficient data handling and learning in decentralized settings and address the critical challenge of achieving consensus among autonomous agents in dynamic environments. The paper will elaborate on these themes, presenting novel methodologies for improving learning processes and network stability and discussing the practical challenges and solutions encountered in real-world implementations.

1.4. Contributions

This paper employs the SNN model to train a group of swarm agents that follow a leader. Each agent has its own SNN, trained independently using the R-STDP algorithm. Each agent receives position data from the agents nearby. The goal is for each agent to keep a commanded distance from the leader agent and the other agents in the group. The encoding and decoding processes for the input and output layers of the SNN are considered fuzzy encoding, and a novel method is introduced to stabilize the network dynamics considering the reward function. This work presents several key contributions:

- The paper presents a comprehensive method for stabilizing and enhancing the learning process in SNN. This method focuses on controlling the unbounded growth of synaptic weights in SNNs, utilizing a strategy that dynamically adapts to changes in reward conditions and coefficients. It introduces a decay rate and learning rate adjustment based on the status of synaptic weights and enhances the responsiveness of the SNN weights to reward change.
- In terms of advancements in FL with R-STDP, the paper addresses the FL challenges in the R-STDP framework. It introduces an event-triggered mechanism for model publishing and receiving within the network, improving network traffic. Additionally, the paper implements a novel weighted aggregation method on the server. This method calculates weights based on the models' arrival time, effectively tackling the asynchronous issues in FL.

The remainder of this paper is structured as follows: Section 2, titled "Preliminaries", provides an overview of the FL algorithm as it applies to consensus flying, including a discussion of the neuron model employed for training and key parameters for the algorithms proposed. Section 3, "Proposed Method", establishes the basis of our investigation and examines the training algorithm and learning through R-STDP. This section presents our innovative approach to weight stabilization called R-CSE method, outlines the network architecture designed for our study, and discusses the application of FL in achieving consensus flying, with a comprehensive explanation of how SNN models are aggregated on a central server. Section 4, "Results and Discussion", presents the outcomes of various simulations and investigates the effects of reward change within FL. Finally, Section 5 concludes the paper by summarizing our principal discoveries and contemplating the implications of our research.

2. Preliminaries

2.1. Consensus Flying Problem

The "Consensus Flying Problem" deals with ensuring drones can work together in real-time to agree on their flight paths and positions. When many drones are close together, like in swarms, avoiding crashes is vital. Advanced algorithms and communication methods are needed so drones can exchange information and handle changing situations and unexpected obstacles.

As shown in Fig. 1, a swarm of agents (follower drones) flies around a leader. The leader is controlled from a remote base station, and the swarm agents should learn to fly safely with the leader. The leader sends its position to all agents, and each agent only sees two neighboring agents. The swarm aims to learn how to keep a commanded distance from each other and the leader. The commanded distance is provided from the leader. Each agent uses the onboard sensors to find the distance and line of sight from neighboring agents.

The follower agents are equipped with an SNN, and their learning algorithm incorporates R-STDP and FL. Each follower agent trains a local network (M^n_{loc}) using R-STDP and sends its model to the leader as the central server. The leader aggregates models and sends back the global model (M^{Global}).

2.2. Neuron model

The LIF neuron model provides a simplified yet powerful representation of neuronal dynamics. Fundamentally, using basic electrical circuit elements, the LIF model captures the behavior of a neuron's membrane potential in response to incoming currents. It incorporates membrane potential decay, realistically simulating how neurons respond to inputs with high frequencies. This enhances the model's accuracy in predicting neuronal behavior under dynamic conditions, offering a more precise tool for neuroscientific research and computational simulations. Mathematically, the LIF model is characterized by a linear differential equation that describes the voltage response to a current input as follows [41],

$$\tau_m \frac{dV_m(t)}{dt} = V_m(t) + E_L + R_m I_e(t) \qquad (1)$$

where τ_m is the membrane time-constant, V_m is the membrane potential, E_L is the reversal potential, R_m is the membrane resistance, and $I_e(t)$ is the input current to the neuron. When the neuron's potential reaches the threshold potential ($V_{threshold}$), it spikes, and the potential immediately returns to the resting potential (V_{rest}).

For scenarios with constant input current, the Inter-Spike Interval (t_{isi}) can be analytically determined by separating variables in the governing differential equation. This allows for the derivation of the t_{isi} as follows [42],

$$t_{isi} = \tau_m \ln\left(\frac{E_l + R_m I - V_{rest}}{E_l + R_m I - V_{threshold}}\right) \qquad (2)$$

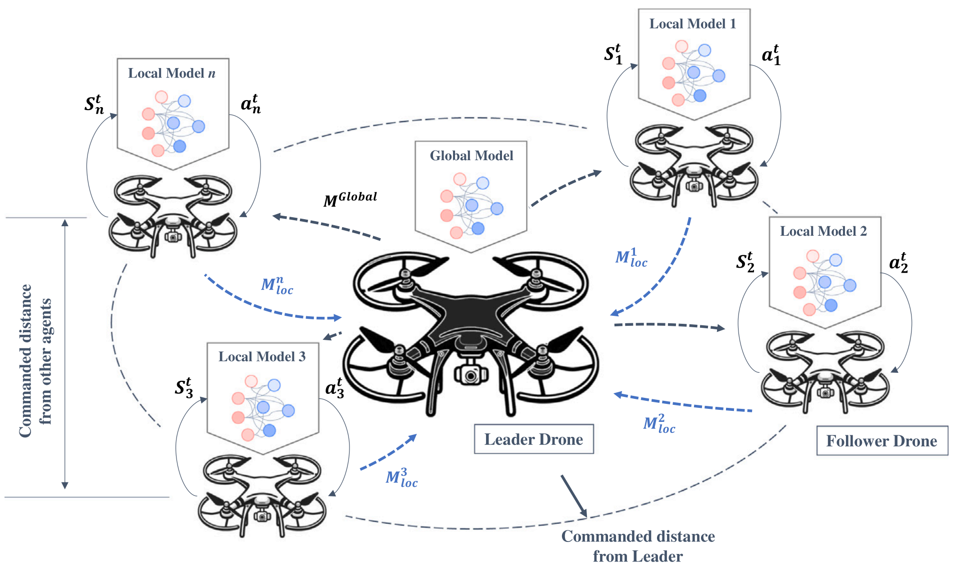


Fig. 1. The central server (the leader) and the surrounding follower agents (white drones). The follower agents learn to fly in a formation to maintain the commanded distance. The local models trained individually by follower agents are sent to the leader. The leader aggregates the models and sends back the global model for another round of training on the follower agents.

It is imperative to note that this solution is dependent on an input current magnitude that induces a transition of the membrane potential from V_{rest} to $V_{threshold}$. Through this analytical framework, the LIF model provides insights into the modulation of neuronal spiking dynamics based on constant input currents.

The neuron does not fire when the input current is minimum and t_{isi} approaches infinity. This occurs when the input current is insufficient to drive the membrane potential to the threshold potential from its resting potential. By setting the t_{isi} to infinity, one obtains the condition:

$$E_l + R_m I - V_{threshold} = 0 \tag{3}$$

or

$$I^{min} = \frac{V_{threshold} - E_L}{R_m} \tag{4}$$

where I^{min} is the minimum input current that makes the neuron reach the potential below the threshold voltage. Conversely, the lowest possible t_{isi} defines the maximum input current. In this paper, we consider the t_{isi} to be one sample time Δt. By considering $\ln(1+z) \approx z$, one can derive the maximum input current that would drive the neuron to spike on every sample time as follows,

$$I^{max} = \frac{\tau_m \left(V_{threshold} - V_{rest}\right)}{\Delta t R_m} + \frac{V_{threshold} - E_L}{R_m} \tag{5}$$

These derived equations determine the range of operation for neurons and establish the boundaries for the maximum weights in the SNN. Specifically, positive synaptic weights indicate the limits of excitement that a neuron can generate. In this context, the minimum input current (or synaptic weight) represents the stimulating influence a neuron can exert without triggering an action potential. In contrast, the maximum input current signifies the most powerful stimulating influence possible. Conversely, considering weight values, these currents reflect the minimum and maximum inhibitory effects. In this case, the minimum and maximum input currents determine how much a neuron can inhibit other neurons from firing.

3. Proposed method

3.1. Network structure

This paper assumes that each agent detects only two neighboring agents besides the leader. The information obtained from other agents includes the Line-of-Sight (LOS) angle and the distance. Each agent's neural network consists of three sub-layers in the input layer, as shown in Fig. 2. Two sub-layers correspond to the two neighboring follower agents (F_1 and F_2), and the third is dedicated to the leader (L). Inputs for these sub-layers are encoded using the Gaussian Receptive Fields (GRF) that use fuzzy membership functions. The network uses the difference between current and commanded distances within the swarm (r_{cmd}) and between followers and the leader (R_{cmd}) to stimulate input neurons.

Every input sub-layer is split into two parts. The first part deals with distances greater than the commanded distance, while the second focuses on the space between the agent and the commanded distance. Within each part, the LOS angle is encoded with fuzzy membership functions. The difference between the current and commanded distance is represented as the error. We transform this difference into an amplitude value using the *tanh* function so that it is bounded between 0 and 1. An error of zero leads to an amplitude of zero, and as the error increases towards infinity, the amplitude approaches one. Consequently, the encoding function for the input layer is expressed as follows,

$$\mu_I(\phi_i, r_i) = \left|tanh(r - r_i)\right| \cdot \exp\left(-\frac{(\phi_i - \zeta)^2}{2\sigma^2}\right) \tag{6}$$

where ζ and σ are the Gaussian membership functions' center and standard deviation. The r_i is the distance from the corresponding agent, ϕ_i is the LOS angle, and μ_I is the vector of the membership degrees. Here, r is a placeholder that can either represent r_{cmd} or R_{cmd}, depending on the context. The firing strengths from fuzzy encoders are then converted to the spiking input based on the neuron model as follows [43],

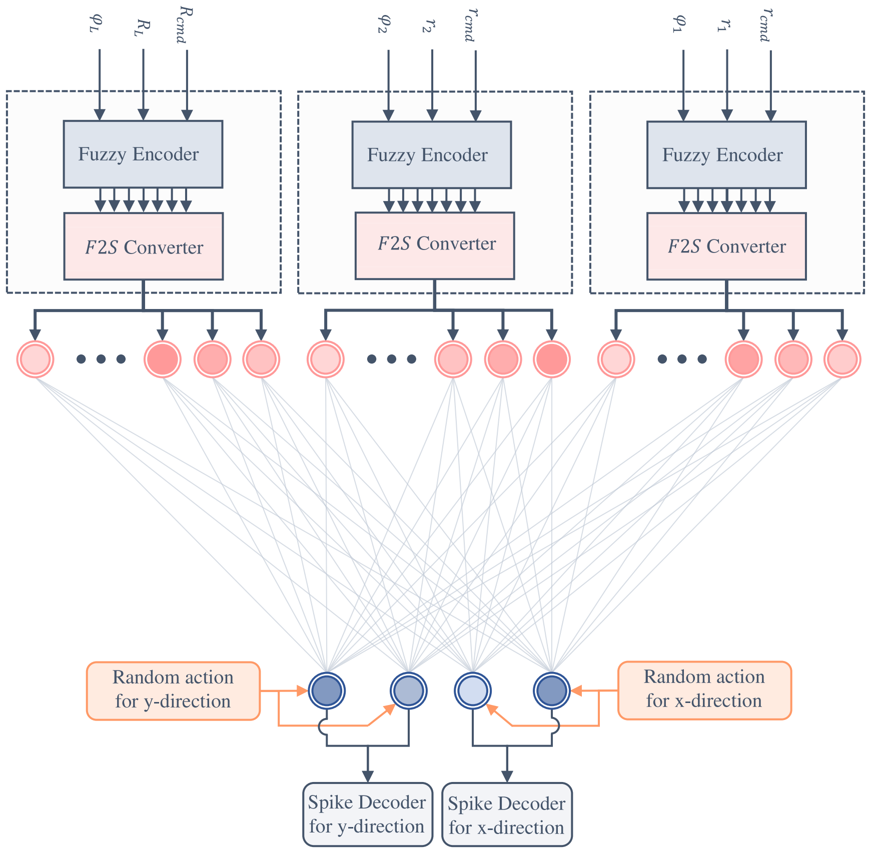
$$I_{sub-layer} = \left(I^{max} - I^{min}\right) \mu_I(\phi_i, r_i) + I^{min} \tag{7}$$

or

$$I_{sub-layer} = \frac{\tau_m \left(V_{th} - V_{res}\right)}{\Delta t R_m} \mu_I(\phi_i, r_i) + \frac{V_{th} - E_l}{R_m} \tag{8}$$

The encoding process is shown in Fig. 3. The Fuzzy-to-Spiking (F2S) block uses (8) to calculate the inputs for the associated sub-layer.

The output layer has two sub-layers, and each sub-layer has two neurons. The first sub-layer determines the Δx, and the second one determines Δy. The first neuron of the sub-layers is for negative values, and the second one is for positive values. Each neuron is associated with the output sign, and the magnitude of the Δx and Δy is encoded into



φ_L
R_L
R_{cmd}
φ_2
r_2
r_{cmd}
φ_1
r_1
r_{cmd}
Fuzzy Encoder
F2S Converter
Random action for y-direction
Random action for x-direction
Spike Decoder for y-direction
Spike Decoder for x-direction

Fig. 2. SNN structure with encoding and decoding layers. Each sub-layer consists of a fuzzy encoder and the F2S Converter, with the output layer receiving inputs from synaptic weights and a random action selector. During the training phase, the output layer receives input only from the random action selector, which then shifts to synaptic weight inputs after the training.

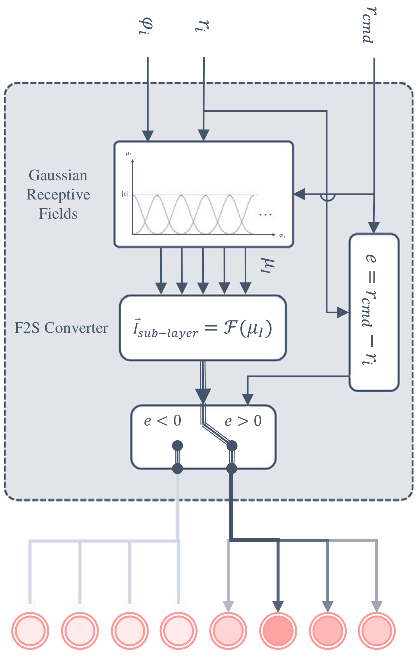
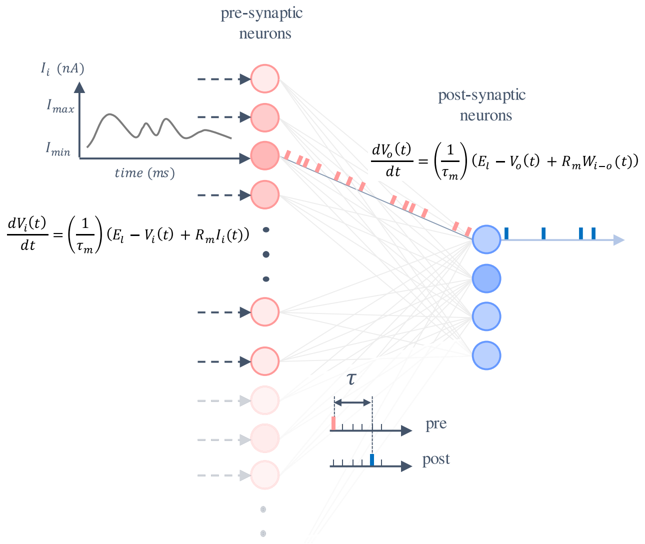


Fig. 3. The fuzzy encoding principle for the input sub-layer.



pre-synaptic neurons
post-synaptic neurons
$I_i\ (nA)$
I_{max}
I_{min}
time (ms)
$\frac{dV_o(t)}{dt} = \left(\frac{1}{\tau_m}\right)(E_l - V_o(t) + R_m W_{i-o}(t))$
$\frac{dV_i(t)}{dt} = \left(\frac{1}{\tau_m}\right)(E_l - V_i(t) + R_m I_i(t))$
τ
pre
post

Fig. 4. Input and output of the SNN.

the output sub-layers based on the minimum and maximum synaptic weights. Eq. (8) is used to encode the magnitude of the random action into the output sub-layers. The only difference is that a function called μ_O is used to normalize the maximum step between 0 and 1 as follows,

$$\mu_{Ox} = \frac{\Delta x}{\Delta X_{max}} \tag{9}$$

$$\mu_{Oy} = \frac{\Delta y}{\Delta Y_{max}} \tag{10}$$

where Δx and Δy are selected actions, and ΔX_{max} and ΔY_{max} are maximum steps (displacements) in X and Y directions. Two random actions, one for Δx and one for Δy, are generated for the training process.

The decoding of the spiking output is determined by the difference in the firing rates of the output neurons within each sub-layer. Let us denote $f(t)$ as the activity of the output neurons that control the movement in the x and y-directions:

$$f(i) = \begin{cases} 1 & \text{if the neuron spikes at time } i, \\ 0 & \text{otherwise,} \end{cases}$$

The equation for decoding this activity can be expressed as:

$$\Delta x_{decoded} = \left[\sum_{i=t-\Delta T}^{t} \left(f^{x+}(i) - f^{x-}(i) \right) \right] \Delta X_{max} \tag{11}$$

where $f^{x+}(i)$ and $f^{x-}(i)$ are the activities of the two output neurons associated with the x-direction. A similar process is applied for decoding in the y-direction:

$$\Delta y_{decoded} = \left[\sum_{i=t-\Delta T}^{t} \left(f^{y+}(i) - f^{y-}(i) \right) \right] \Delta Y_{max} \tag{12}$$

where ΔT is the time window the network updates weights.

One of the challenges in robotic applications is ensuring smooth transitions in actions to prevent abrupt and potentially harmful changes. Therefore, the recursive random number generation method is used to produce correlated random numbers. This method ensures that during training, the current displacements of the robot are influenced by its previous displacements, leading to smoother transitions. The recursive random number generation can be formulated as,

$$_t = \gamma \cdot {}_{t-1} + (1-\gamma) \cdot Y_t \tag{13}$$

where $_t$ is the random action at time t, γ is a correlation coefficient, and Y_t is a random number drawn from a standard distribution (e.g., Gaussian) at time t. This equation ensures that the random action at any given time t is a weighted combination of the previous action and a new random number.

3.2. Training algorithm

The R-STDP algorithm is a learning technique inspired by biological processes in the brain, which is believed to be fundamental to certain learning processes [44]. The algorithm's core principle is that when a pre-synaptic neuron activates just before its post-synaptic counterpart, the synapse's strength connecting them should increase, and vice versa if the post-synaptic neuron fires first.

Within the SNN framework, pre-synaptic neurons are the input neurons, and post-synaptic neurons function as the output neurons. The function $STDP(\tau)$ can be defined as the firing timelines of both input and output neurons [45] as,

$$STDP_{kl}(\tau) = \mathcal{A} \exp\left(-\frac{\tau}{\tau_s}\right) \text{ for } \tau \geq 0 \tag{14}$$

where $\mathcal{A}$ stands as the exponential function's amplitude, and τ is the difference between the firing time of the input neuron (k) and output neuron (l) (see Fig. 4). Meanwhile, τ_s acts as the time constant, setting the decay rate for the $R-STDP$ function. Should τ_s approach infinity, the exponential function converges to 1, neutralizing time's effect on the $R-STDP$ function.

The adjustment of synaptic weights follows the given equation:

$$\dot{W}_{kl}(t) = STDP_{kl}(\tau)\mathcal{R}(t) \tag{15}$$

where $\dot{W}_{kl}(t)$ denotes the rate of change of the synaptic weight that connects neurons k and l. This weight determines the input that the post-synaptic neuron receives upon the spiking of its pre-synaptic neuron, which is quantified as $I(t)$. The term $\mathcal{R}(t)$ represents the reward that is received at time t.

The reward functions used in this paper are as follows,

$$\mathcal{R}^{Fj}_{Fi}(t) = C^{Fj}_{Fi} \left[r^{Fj}_{Fi}(t-1) - r^{Fj}_{Fi}(t) \right] tanh(r^{Fj}_{Fi}(t) - r_{cmd}) \tag{16}$$

$$\mathcal{R}^{L}_{Fi}(t) = C^{L}_{Fi} \left[r^{L}_{Fi}(t-1) - r^{L}_{Fi}(t) \right] tanh(r^{L}_{Fi}(t) - R_{cmd}) \tag{17}$$

where, $\mathcal{R}^{Fj}_{Fi}$, r^{Fj}_{Fi}, and r_{cmd} denote the reward, distance, and commanded distance between two i and j follower agents, respectively. Similarly, $\mathcal{R}^{L}_{Fi}$, r^{L}_{Fi}, and R_{cmd} represent the reward, distance, and the commanded distance between the follower agent i and the leader (L), respectively. The terms C^{Fj}_{Fi} and C^{L}_{Fi} are the reward coefficients and the $tanh(r^{Fj}_{Fi}(t) - r_{cmd})$ and $tanh(r^{L}_{Fi}(t) - R_{cmd})$ functions determine the reward's sign according to the agents' relative distance and the commanded distance. The expressions $r^{Fj}_{Fi}(t-1) - r^{Fj}_{Fi}(t)$ and $r^{L}_{Fi}(t-1) - r^{L}_{Fi}(t)$ specify the magnitude of the instantaneous reward.

If an agent finds itself farther away from the commanded distance than a neighboring agent or the leader, it will be rewarded positively for decreasing its distance. Conversely, moving closer results in a negative reward if the agent is within the commanded distance from a neighboring agent or the leader. This system is designed to encourage the maintenance of a commanded distance: being too far away from the commanded distance invites a penalty. At the same time, positive reinforcement is given for closing the gap between the current distance and the commanded distance.

One of the challenges in R-STDP is the unbounded growth or decay of synaptic weights, which can impede effective learning in neural networks. The following section introduces a novel weight-stabilization method to address this challenge and enhance the algorithm's applicability.

3.2.1. Weight stabilization using reward-modulated competitive synaptic equilibrium (R-CSE)

Controlling the excessive increase of synaptic weights in SNNs is important to maintain network resilience and function. If not controlled, this growth can lead to saturation, affecting the network's ability to learn and adapt. When the network receives fuzzy sets of firing strengths as input, the synaptic weights grow in a pattern influenced by the Gaussian function's shape used for fuzzy encoding. Imposing a limit on synaptic weights disrupts this growth pattern over time, and eventually, all the synaptic weights reach the maximum. Weight normalization, while preventing excessive growth in one part of the network, can inhibit overall growth; when a synaptic connection reaches its maximum, its activation subsequently diminishes other weights.

Traditional methods like L1 regularization and weight decay employ a constant decay rate, which can slow the network's responsiveness to changes in rewards. Alternatively, a more advanced approach, the Bienenstock, Cooper, and Munro (BCM) method, dynamically adjusts both a threshold and a decay rate in response to input variations. However, this method does not provide a control mechanism for the fuzzy inputs. In this chapter, we introduce a method called R-CSE to manage the unbounded growth of synaptic weights while maintaining the gradual change in the synaptic weights formed due to differences in firing strength from fuzzy membership functions. Our method also dynamically adjusts the network when the reward changes by adjusting the maximum synaptic weight based on the reward.

The enhanced version of the R-STDP method considering the control mechanism from R-CSE algorithm is expressed as follows,

$$\dot{\boldsymbol{W}}(t) = \boldsymbol{\alpha} \odot \mathbf{STDP}(\tau) \odot \mathcal{R}(t) - \boldsymbol{\Theta} \odot \mathrm{sgn}(\boldsymbol{W}) \tag{18}$$

where $\odot$ is the Hadamard product, $\boldsymbol{\alpha}$ is the learning rate matrix, and $\boldsymbol{\Theta}$ is the decay rate matrix. The primary distinction between the R-CSE method and the approach detailed in Section 2.4 lies in the decay rate, which allows the learning system to remain adaptable after the learning phase, and in the learning rate, which is represented as a matrix rather than a scalar value affecting all synaptic weights uniformly. These modifications enhance the learning algorithm's flexibility in responding to reward changes and provide greater control over synaptic weight adjustments.

Let us define $\mathcal{S}$ as the set of input and output neurons that fired at time t in one of the network sections. If we consider $W^S_{max}(t)$ as the maximum weight among the firing neurons in set $\mathcal{S}$, then we can characterize the learning rate using a Sigmoid function. The learning rate value ($\boldsymbol{\alpha}^S(t)$) gradually transitions from 1 to 0 as the learning process advances, as explained below:

$$\boldsymbol{\alpha}^S(t) = \frac{1}{1+\exp\left[\frac{1}{\epsilon}\left(|W^S_{max}(t)| - \Psi^S\right)\right]}, \quad \left(\Psi^S = \frac{\mathcal{R}^S_{max}}{\mathcal{R}^G_{max}} I^{max}\right) \tag{19}$$

where $\mathcal{R}^S_{max}$ is the maximum reward in the network section (e.g., $max(\mathcal{R}^{Fj}_{Fi})$), $\mathcal{R}^G_{max} = max(\mathcal{R}^{Fj}_{Fi}, \mathcal{R}^L_{Fi})$, and ϵ is a small positive number that controls the curvature of the function around $W^S_{max}(t) = \Psi^S$. This model determines the learning rate by the highest synaptic weight among the active input and output neurons. This mechanism is similar to the "winner-takes-all" approach. When a synaptic connection reaches its weight limit, it prevents further changes in the adjacent synaptic weights.

The network contains a variety of reward functions, each with its own maximum and minimum values. The highest reward value in a specific area of the network sets the limit for the synaptic weight in that area. The synaptic weight limit is linked to the ratio of the local maximum reward ($\mathcal{R}^S_{max}$) to the global maximum reward ($\mathcal{R}^G_{max}$). As a result, the network section with the highest local maximum reward ($\mathcal{R}^S_{max} = \mathcal{R}^G_{max}$) attains the maximum allowable synaptic weights because $\frac{\mathcal{R}^S_{max}}{\mathcal{R}^G_{max}} = 1$, while sections with lower local maximum rewards reach only a proportional fraction of the maximum weight. The adjustment of the learning rate transforms into a competitive algorithm that modifies the growth rate of individual synaptic weights by considering network parameters, like reward and maximum synaptic weight.

A significant challenge in learning algorithms is their capacity to adapt to changes in rewards. Commonly, once the learning rate reduces to zero, weight adjustments stop. To address this, a variable decay rate is introduced to prevent weights in each network section from indefinitely remaining at their peak values. In our method, the decay rate is represented as a matrix, and it is calculated using the SoftPlus function, enabling it to adjust according to the current stage of learning. This method ensures that weight modifications continue to respond effectively to changes in the learning environment.

This chapter defines the decay rate as a function of the maximum synaptic weight among neurons in the set $\mathcal{S}$. This approach is designed to address a critical aspect: when the maximum synaptic weight in $\mathcal{S}$ reaches its peak ($|W^S_{max}(t)| = \Psi^S$), it is essential that the learning rate remains above zero. This condition is necessary to allow weight change and prevent the learning rate from stagnating at zero ((18)). Simultaneously, the learning rate must not exceed the maximum acceptable rate of weight change, which is $\mathcal{A} \times \mathcal{R}^G_{max}$. When the reward coefficients change after training, it can cause $|W^S_{max}(t)|$ to exceed Ψ^S for set $\mathcal{S}$. With these considerations, we propose that the decay rate should be set to $\mathcal{A}/\lambda \times \mathcal{R}^G_{max}$ when $|W^S_{max}(t)| = \Psi^S$ and increase to $\lambda\mathcal{A} \times \mathcal{R}^G_{max}$ when $|W^S_{max}(t)| = 2\Psi^S$, where λ is a coefficient that controls the rate of decay when $|W^S_{max}(t)| > \Psi^S$.

By applying the mentioned condition and solving for the SoftPlus function, the decay rate function can be obtained as follows,

$$\Theta^S = \left(\frac{\eta}{\beta}\right)\log\left(1+\exp\left[\beta\left(|W^S_{max}(t)| - \Psi^S\right)\right]\right) \tag{20}$$

where $\eta = \frac{\mathcal{A}\mathcal{R}^G_{max}\ln(2^{\lambda^2}-1)}{\lambda\Psi^S\log(2)}$ is a scaling parameter that can adjust the output scale of the function, and $\beta = \frac{\ln(2^{\lambda^2}-1)}{\Psi^S}$ controls the curvature of the function. A higher β makes the SoftPlus function approach a step function, making it closer to the binary behavior. Conversely, a smaller β makes the function smoother and more gradual. Eq. (20) can be represented as,

$$\Theta^S = \left(\frac{\mathcal{A}\mathcal{R}^G_{max}}{\lambda\log(2)}\right)\log\left(1+\exp\left[\left(\frac{\ln(2^{\lambda^2}-1)}{\Psi^S}\right)\left(|W^S_{max}(t)| - \Psi^S\right)\right]\right) \tag{21}$$

The choice of setting the decay rate to $\lambda\mathcal{A} \times \mathcal{R}^G_{max}$ when $|W^S_{max}(t)| = 2\Psi^S$ is based on the feature of reward coefficients. Specifically, when the reward coefficients in (16) and (17) increase, leading to new condition where $\mathcal{R}^S_{max}$ or $\mathcal{R}^G_{max}$ change, the $|W^S_{max}(t)|$ is allowed to increase. Conversely, a decrease in the reward coefficient, resulting in $|W^S_{max}(t)| > \Psi^S$, necessitates a higher decay rate to reduce the $|W^S_{max}(t)|$ back to Ψ^S.

When $|W^S_{max}(t)| < \Psi^S$, the reward adjusts the synaptic weights, and there is no weight decay to disturb the learning process. When $|W^S_{max}(t)| > \Psi^S$, the decay rate changes the synaptic weights and brings the maximum weight to the reward zone, where $|W^S_{max}(t)| < \Psi^S$ and the networks responds to reward change.

Lemma 3.1. *The R-CSE method is asymptotically stable in its equilibrium point $W^S_{max}(t) = \Psi^S$.*

Proof. We consider the dynamical system given by the equation of synaptic weights for $W^S_{max}(t) \geq 0$ that receives a positive reward ($\mathcal{R}(t) > 0$) as,

$$\begin{aligned}\dot{W}^S_{max}(t) =& \frac{1}{1+\exp\left[\frac{1}{\epsilon}\left(W^S_{max}(t) - \Psi^S\right)\right]} STDP(\tau)\mathcal{R}(t) \\ &- \left(\frac{\mathcal{A}\mathcal{R}^G_{max}}{\lambda\log(2)}\right)\log\left(1+\exp\left[\left(\frac{\ln(2^{\lambda^2}-1)}{\Psi^S}\right)\left(W^S_{max}(t) - \Psi^S\right)\right]\right),\end{aligned} \tag{22}$$

To assess the stability of this system around the equilibrium point, we introduce a Lyapunov function candidate $V(z)$, where $z = W^S_{max}(t) - \Psi^S$. A common choice for such analyses is a quadratic function of the deviation from the equilibrium:

$$V(z) = \frac{1}{2}z^2. \tag{23}$$

This positive definite function has a minimum at the equilibrium point, satisfying the essential criteria for a Lyapunov function. The derivative of $V(z)$ with respect to time, $\dot{V}(z)$, is then calculated to determine the rate of change of the Lyapunov function along the trajectories of the system:

$$\dot{V}(z) = z\dot{z} \tag{24}$$

Substituting $\dot{W}^S_{max}(t)$ from (22) into the above expression, we have,

$$\begin{aligned}\dot{V}(z) = z \times \Bigg[& \frac{1}{1+\exp\left[\frac{1}{\epsilon}(z)\right]} STDP(\tau)\mathcal{R}(t) - \left(\frac{\mathcal{A}\mathcal{R}^G_{max}}{\lambda\log(2)}\right)\log\Bigg(1+\exp \\ & \left[\left(\frac{\ln(2^{\lambda^2}-1)}{\Psi^S}\right)(z)\right]\Bigg)\Bigg]\end{aligned} \tag{25}$$

A negative $\dot{V}(z)$ indicates that the system's energy decreases over time, concluding that the equilibrium point is asymptotically stable. Conversely, a positive $\dot{V}(z)$ in any region would suggest the presence

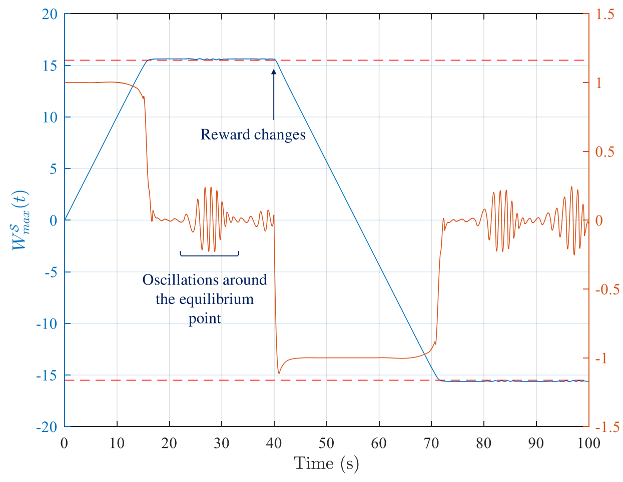


Fig. 5. Synaptic weight change for $\lambda = 5, \Psi^S = 15.5$, and $\mathcal{AR}^G_{max} = 1$.

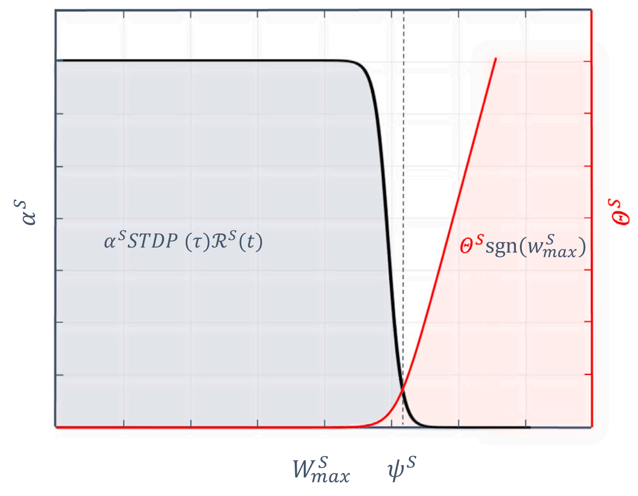

α^S
Θ^S
$\alpha^S STDP(\tau)\mathcal{R}^S(t)$
$\Theta^S \mathrm{sgn}(w^S_{max})$
W^S_{max}
ψ^S

Fig. 6. Reward-based learning rate and decay rate functions. In the blue region (active learning rate), the reward adjusts the weights, and in the red region (active decay rate), the RCSE method controls synaptic growth.

of instability or regions of attraction that do not encompass the entire state space.

In analyzing the system's stability, we focus on the behavior of the derivative of the Lyapunov function, $\dot{V}(z)$, across different regions of z. We decompose the dynamics of $\dot{z}$ into its constituent components to systematically analyze the stability conditions. We assess the relative magnitudes of the two main components influencing $\dot{V}(z)$:

1. The first term, represented as $\frac{1}{1+\exp\left[\frac{1}{\epsilon}z\right]} STDP(\tau)\mathcal{R}(t)$, denotes the effect of learning rate and is inherently positive when the synaptic connection receives positive reward consistently.
2. The second term, $\left(\frac{\mathcal{AR}^G_{max}}{\lambda \log(2)}\right) \log\left(1+\exp\left[\left(\frac{\ln(2^{\lambda^2}-1)}{\Psi^S}\right)z\right]\right)$, captures the dynamic decay rate of synaptic weights, which is governed by the SoftPlus function.

Analysis for $z < 0$:

In this region, we observe that the term $\exp\left[\left(\frac{\ln(2^{\lambda^2}-1)}{\Psi^S}\right)z\right]$ goes to zero and makes the term inside the log function go to 1. This implies that the contribution of this term to $\dot{V}(z)$ is negligible in this region. Moreover, the first term remains positive throughout, and given that it is multiplied by z (which is negative in this region), the overall contribution to $\dot{V}(z)$ is negative. Consequently, $\dot{V}(z)$ is negative for $z < 0$, indicating that any perturbations from the equilibrium in this region will decrease over time, thereby contributing to the system's stability.

Analysis for $z > 0$:

For this region, the magnitude of the second term significantly exceeds that of the first term. This predominance is critical as it is associated with a negative sign in the $\dot{V}(z)$ equation. Therefore, the negative contribution of this component ensures that $\dot{V}(z)$ remains negative throughout this region. It indicates that any deviation from the equilibrium state results in the system's energy decreasing over time, leading to the conclusion that the equilibrium point $W^S_{max}(t) = \Psi^S$ is asymptotically stable for $z > 0$. □

Fig. 5 demonstrates the performance of the R-CSE when it regulates the synaptic weights to prevent unbounded growth. The red dotted line represents the value of Ψ^S, which is derived from the maximum reward value of the corresponding network section. As shown in Fig. 5, the synaptic weight oscillates around Ψ^S, and when it drops below Ψ^S, the learning rate is set to 1 by (19). This allows any changes in the reward function to be applied to the synapse.

According to Fig. 6, when $W^S_{max} \leq \Psi^S$, synaptic weights in set $\mathcal{S}$ increase. If the reward changes, Ψ^S also changes. Depending on the current value of W^S_{max}, the R-CSE either increases or decreases the synaptic weights within set $\mathcal{S}$.

Fig. 7 shows how the maximum synaptic weight of the active synapses in set $\mathcal{S}$ stops the adjacent synaptic connections' growth by setting the learning rate of the set to 0.

3.2.2. Federated learning for consensus flying

In FL, a key challenge is centralizing various models on one server. This process must effectively combine these models to create a unified global model without compromising the specific adjustments made to each model. A critical strategy involves choosing models that contain substantial information. Another significant aspect is determining the frequency of model aggregation. Shorter intervals between aggregations can enhance learning efficiency but may strain network resources, particularly as the number of participating agents and devices grows. Conversely, longer intervals might slow down the learning process due to delayed updates of the global model. This section proposes an aggregation method for SNN. Our focus is on reducing network usage and energy consumption.

This approach allows clients to upload their local model updates at different times rather than synchronously. Such a method is particularly beneficial in reducing the negative impacts of device heterogeneity, which can include varying computational capacities and network connectivity among devices [46]. In traditional FL setups, delays caused by poor network signals or unexpected client crashes can significantly prolong the time the server takes to receive updates from all clients. By adopting asynchronous aggregation, the server processes and aggregates models as they are received without synchronizing with all clients. This strategy accelerates the training process, making FL more efficient and adaptable to diverse client conditions.

Our proposed FL model aggregation algorithm aims to establish an efficient and event-triggered system for global and local model publishing. This system relies on the similarity between consecutive global and local models and publishes updates only when significant changes are detected, thus avoiding redundant updates and improving overall efficiency. Unlike the uniform model updates in FedAvg [47,48], our approach allows individual agents to evaluate and send their local models based on a similarity threshold with the global model, thereby enabling a potentially more effective update process. Our aggregation

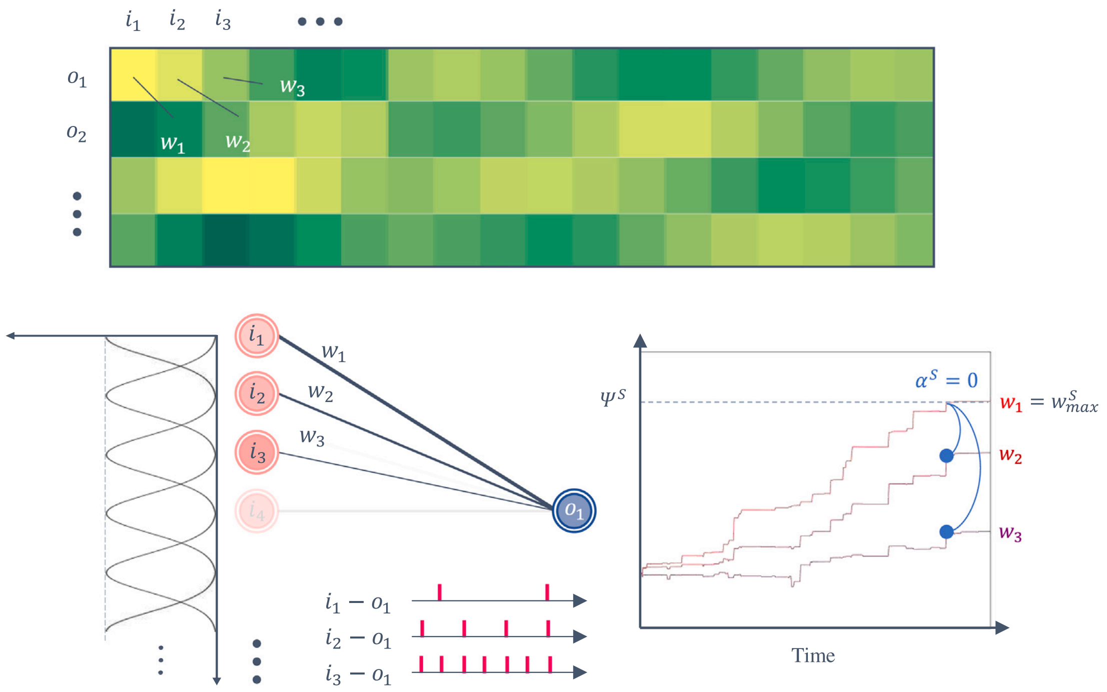


Fig. 7. RCSE working principle in inhibiting the adjacent synaptic connections. The heatmap shows the synaptic weight matrix. Neurons have different firing strengths due to the difference in fuzzy membership values, which affects the increase or decrease rate and shapes the patterns in the synaptic weight matrix.

strategy emphasizes similarity metrics for model updates, which is not commonly emphasized in methods like FedNova [49], adding a layer of context sensitivity to our approach.

In our approach, considering the difference in agents' neural network parameters and maximum and minimum synaptic weights, the weights are normalized to align them on a uniform scale ranging from −1 to 1. This normalization process makes the neural model values comparable across the network. Based on the maximum and minimum synaptic weights outlined in (5), and taking into account the highest excitation (I^{max}) and inhibition ($-I^{max}$), the normalization of synaptic weights is performed as follows:

$$\overline{\boldsymbol{W}}_k(t) = \frac{1}{I_k^{max}}[\boldsymbol{W}_k(t)] \tag{26}$$

where $\boldsymbol{W}_k(t)$ represents the matrix of synaptic weights, $\overline{\boldsymbol{W}}_k(t)$ denotes the normalized synaptic weight matrix for agent $k \in \{1, 2, 3, ..\}$, and I_k^{max} is the maximum synaptic weight for agent k.

The global model on the server (Leader) is then computed using a weighted average,

$$\overline{\boldsymbol{W}}_G(t) = \frac{\sum_{k=1}^{N} \omega_k.\overline{\boldsymbol{W}}_k(t)}{\sum_{k=1}^{N} \omega_k} \tag{27}$$

where $\overline{\boldsymbol{W}}_G(t)$ is the global normalized model on the central server, N is the number of agents, and ω_k is the aggregation weight for each SNN model, defined as,

$$\omega_k = \frac{1}{\sqrt{mn}}\|\overline{\boldsymbol{W}}_k(t)\|_F \exp\left(-\frac{t-T_k}{\tau_{cs}}\right) \quad (t \geq T_k) \tag{28}$$

where the term $\|.\|_F$ is the Frobenius norm, and m and n are the dimensions of the matrix $\overline{\boldsymbol{W}}_k(t)$, used for normalizing the Frobenius norm. T_k indicates the time at which agent k last transmitted its local model to the central server, and τ_{cs} is a time constant that reduces the weight to zero if there is no recent update from the agent.

Both agents and the central server employ an event-triggered mechanism for transmitting local and global models. Throughout the training phase, each agent calculates the Euclidean distance between the most recent global model from the central server and its current synaptic weights matrix, as follows,

$$\mathcal{D}_a(\overline{\boldsymbol{W}}_k(t), \overline{\boldsymbol{W}}_G(T_{cs})) = \frac{1}{2\sqrt{mn}}\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij}-b_{ij})^2} \tag{29}$$

where $\mathcal{D}_a$ is the Euclidean distance on the agent side, T_{cs} is the time when the central server published the global model, and a_{ij} and b_{ij} are elements of the latest global model and the current local model, respectively. If this distance exceeds a certain threshold, set between 0 and 1, the agent transmits its model to the central server.

If the $\mathcal{D}_a$ on the agent k reaches the threshold and it does not receive any update from the server, the agent sends its model to the server, and then it calculates the $\mathcal{D}_a$ between current synaptic weights $\overline{\boldsymbol{W}}_k(t)$ and the model it recently sent to the server $\overline{\boldsymbol{W}}_k(T_k)$ until it receives a new model update from the central server.

The central server follows a similar procedure as the agents, evaluating the distance $\mathcal{D}_G$ between the current and recently published model at time T_{cs},

$$\mathcal{D}_G(\overline{\boldsymbol{W}}_G(t), \overline{\boldsymbol{W}}_G(T_{cs})) = \frac{1}{2\sqrt{mn}}\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij}-b_{ij})^2} \quad (t \geq T_{cs}) \tag{30}$$

Incorporating the proposed FL method with the R-CSE algorithm, the modified R-STDP equation can be represented as follows,

$$\begin{aligned}\dot{\boldsymbol{W}}_k(t) = (1-\delta(t-T_{cs}))\left[\boldsymbol{\alpha} \odot \textbf{STDP}(\tau) \odot \mathcal{R}(t) - \boldsymbol{\Theta} \odot \operatorname{sgn}(\boldsymbol{W}_k(t))\right] \\ + \delta(t-T_{cs})I_k^{max}\left(\overline{\boldsymbol{W}}_G(t) - \overline{\boldsymbol{W}}_k(t)\right)\end{aligned} \tag{31}$$

where δ is the Dirac delta function. Algorithm 1 shows the step-by-step implementation process of the proposed method.

Algorithm 1 High-Level Algorithm for the Proposed FL Algorithm

```
Require: Initialization of Central Server and Agents
Ensure: Updated Global Model on the Central Server and Local Models on Agents
1:  Initialize the agents and Central Server with default parameters for model publication threshold, Euclidean distance, and model publish status
2:  Initialize the Global Model on the Central Server
3:  if $t$ is greater than 0 then
4:      Normalize synaptic weights of local models using (26)
5:      Aggregate models from all agents at the Central Server using (27)
6:      Calculate the $\mathcal{D}_G$ between the current and previous global models on the Central Server using (30)
7:      if $\mathcal{D}_G >$ the Central Server's threshold then
8:          Publish the global model
9:          set $T_{cs} = t$
10:     end if
11:     for each Agent in the network do
12:         if Central Server publishes a new global model then
13:             Update the local model of the Agent with the global model using (31)
14:         else
15:             Agents evaluate their local models against the latest global model ($\mathcal{D}_a$) using (29)
16:             if $\mathcal{D}_a >$ the Agent's threshold then
17:                 Send the model to the Central Server
18:                 set $T_k = t$
19:             end if
20:         end if
21:     end for
22: end if
```

The proposed algorithm allows agents to communicate less often and save energy. It only sends essential updates to the Central Server, which helps when many agents have different SNN models and communication interfaces. This method reduces unnecessary data transmission, making the whole system more efficient.

4. Results and discussion

In this section, we conducted a numerical simulation to validate the performance of the proposed method. The simulation involves a group of five agents flying around a leader who is moving in a circular path. Initially, a scenario without implementing FL was conducted to evaluate the performance of the SNN in achieving coordinated flight. During this phase, the effect of the change in reward was simulated to examine the R-CSE method. In the second part of the simulation, the proposed FL aggregation algorithm is used, where the leader agent acts as a central server. Finally, the algorithm was tested both before and after changing the rewards.

4.1. Simulation without FL

In this simulation, we modeled five agents, each equipped with its own SNN model, capable of reaching a maximum speed of 1 m/s. The architecture of each agent's neural network included 72 input neurons. Since each agent was designed to detect three distinct objects within its environment, the input layer was organized into sub-layers, with 24 neurons dedicated to each object. The network's output layer comprised 4 neurons, divided equally to represent Δx and Δy movements. The SNN model in the simulation is a fully connected network, and the parameters of the LIF neuron are also represented in Table 1.

The R-STDP mechanism updated synaptic weights at 10 ms intervals. During these intervals, the learning algorithm adjusted the agent's states based on received data from other agents and the leader while

Table 1
Parameter values for LIF neuron model [50].

Parameter	Value	Description
R_m	40 MΩ	Membrane Resistance
τ_m	30 ms	Membrane time constant
E_l	−70 mV	Resting potential
V_{res}	−70 mV	Reset potential
V_0	−70 mV	Initial membrane potential
V_{th}	−50 mV	Threshold membrane potential

Table 2
Simulation parameters.

Parameter	Value	Description
ΔT	10 ms	Weight and state update sample time
τ_s	2 ms	Time constant for R-STDP
$\mathcal{A}$	1	Amplitude in R-STDP function
λ	5	Decay rate coefficient
Δx and Δy	0.01 m	Max step per ΔT
σ	0.5	Gaussian function's std. deviation
Δt	1 ms	Minimum inter-spike interval
I^{min}	0.5	Lower bound of synaptic weight
I^{max}	15.5	Upper bound of synaptic weight
γ	0.95	Correlation Coefficient

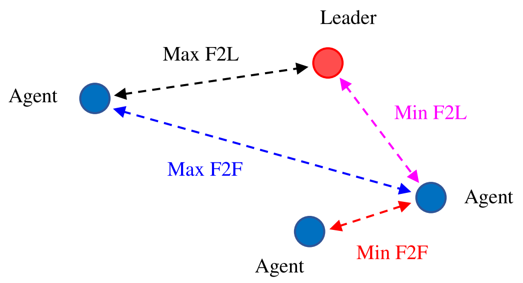


Fig. 8. Measured distances used for evaluating swarm flight performance and collision detection.

simultaneously generating random outputs as part of an exploration strategy.

Table 2 shows the simulation parameters. The simulation was done in a 10 m by 10 m area, and the leader followed a circular path centered at (5,5) with a 2.5 m radius and a 0.1 m/s speed.

To monitor swarm performance, the minimum and maximum distances of each agent from other agents and the minimum and maximum distances of the swarm from the leader were measured. Fig. 8 shows the definition of the distances.

The simulation included two phases. During the initial phase, the objective was for the agents to learn to maintain the commanded distance from each other and the leader. This phase took 600 s for training, and the reward coefficient among followers (C_{Fi}^{Fj}) was set at 0.02, while the coefficient between followers and the leader (C_{Fi}^{L}) was set at 0.07. These parameters were derived from a series of numerical simulations. A higher value of C_{Fi}^{L} signifies an increased emphasis on the leader in the learning process, which means that the distance to the leader is more important than the commanded distance between agents.

Fig. 9 shows the simulation results for the R-CSE method. According to the results, the agents rapidly aligned around the leader within 6.89 s, and the maximum distance was reduced from 7.632 meters to the target distance of 2 m. The swarm completed the formation around the leader in approximately 8.94 s, avoiding collisions (see Fig. 10).

As mentioned in Section 3, the input encoding uses the error between current and commanded distance. Therefore, one of the advantages of the encoding and learning method in this paper is that the learned policies are independent of the commanded distance. The commanded distance can be changed after training since the SNN uses the distance error.

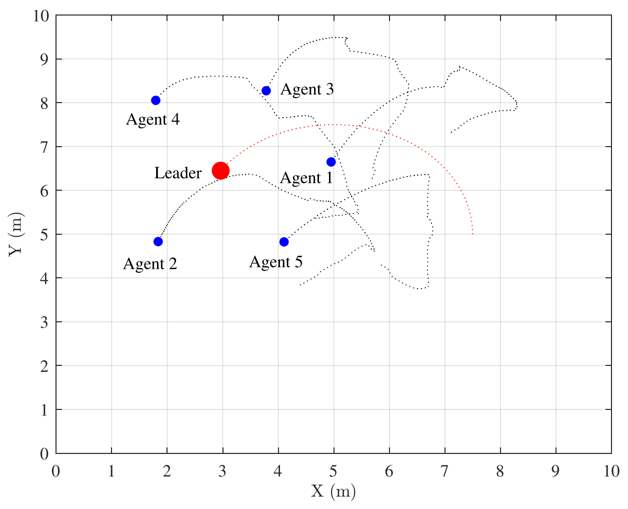


Fig. 9. Agents' trajectory during the test phase.

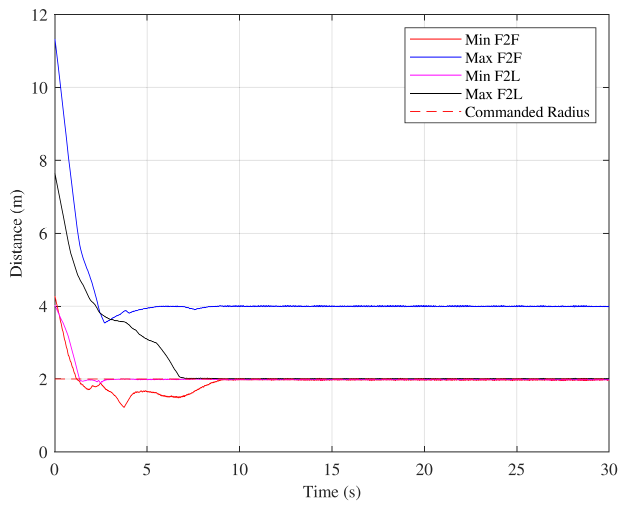


Fig. 10. Variation of distances within the swarm during the test phase.

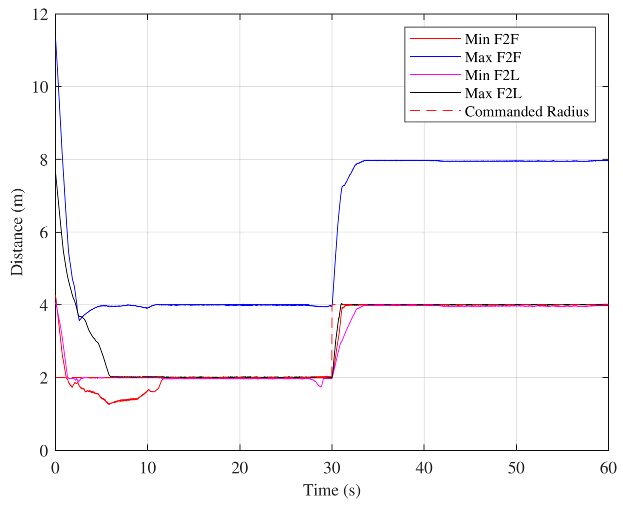


Fig. 11. Adaptive response to commanded distance adjustments: dynamic reconfiguration during the test phase.

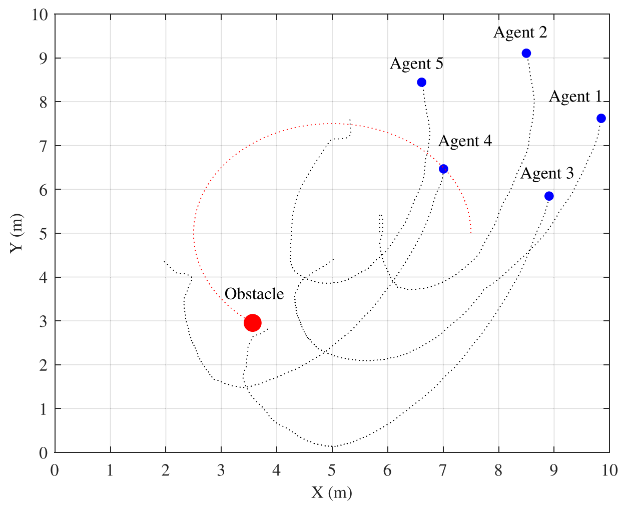


Fig. 12. Trajectory adaptations of following agents in response to reward change for the leader during the test phase.

Another key feature of our SNN implementation is that when the agent successfully aligns with the commanded distances, resulting in zero error, the input neurons cease to spike. This characteristic leverages the sparsity of spikes in SNNs, significantly reducing energy consumption, as the spike is the primary energy consumer in these networks [51]. This absence of spiking under zero-error conditions demonstrates the network's precision and operational efficiency, as it minimizes unnecessary computational activity and power usage. While quantized ANNs are beneficial in reducing model size and computational demands, they often face challenges in maintaining accuracy due to reduced precision, which can be critical in complex decision-making contexts.

Fig. 11 shows the agents' response to changes in commanded distance after training. According to this figure, when the commanded distance is changed at 30 s, the swarm immediately responds to this change in 2.98 s without disrupting the formation or any collision.

After 600 s, the leader is changed into an obstacle, and its reward coefficient C_F^L is changed to 0.0175. The reward sign function, tanh in (17), is also changed to −1, so the reward function for the leader is changed as follows,

$$\mathcal{R}_{Fi}^L(t) = -C_{Fi}^L \left[r_{Fi}^L(t-1) - r_{Fi}^L(t) \right] \tag{32}$$

When the leader is transformed into an obstacle, the encoding equation for the input layer must be changed. This is because the obstacle has no commanded distance, and the agents must maintain a commanded distance only from each other. Therefore, the commanded distance from the obstacle encoder in the input layer must be removed. Therefore, (6) can then be rewritten as follows:

$$\mu_I(\phi_i) = \exp\left(-\frac{(\phi_i - \zeta)^2}{2\sigma^2} \right) \tag{33}$$

The simulation proceeded for an additional 1200 s, during which the synaptic weights were adjusted in accordance with the new reward function given by (32). The results of the reward change are shown in Figs. 12 and 13, which indicate that the agents quickly reduced their initial distance to the commanded distance of 2 m. Simultaneously, the minimum distance from the obstacle, the leader, increased over time, indicating that the agents adapted their behavior to maintain a greater distance from the obstacle. Fig. 12 shows the trajectory of each agent after the reward change.

In order to better understand the effect of reward change on the SNN, the synaptic weights matrix before and after reward change

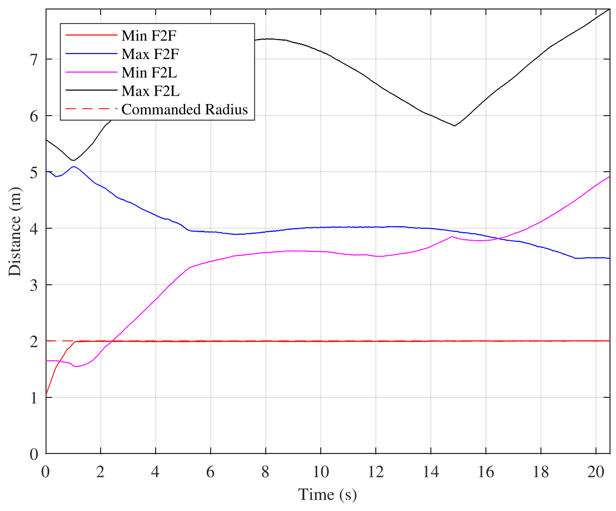


Fig. 13. Variations in distances after reward changes and Leader becomes Obstacle - test phase.

has been illustrated in Figs. 14 and 15, the vertical axis shows the output neurons. The first output neuron is for negative displacement in the x-direction, while the second output neuron is dedicated to positive displacement in the x-direction. Similarly, the third output neuron corresponds to negative displacement in the y-direction and the fourth output neuron to positive displacement in the y-direction. The horizontal axis shows the input neurons. The neuron IDs from 1 to 24 are for the first sub-layer dedicated to the neighboring follower. The network has two sub-layers for the neighboring follower agents, but only one is shown since they are similar in the case of synaptic weight values. The neuron numbers from 25 to 48 are for the sub-layer dedicated to the leader. The R-CSE method aims to maintain the synaptic weight matrix gradient while adapting to changes in the reward signal.

Considering the numerical values presented in Table 2 along with the reward coefficients $C_{Fi}^{Fj} = 0.02$ and $C_{L}^{Fj} = 0.07$, and $r_{Fi}^{Fj} = 1$ m/s and $r_{Fi}^{L} = 0.1$ m/s, the maximum rewards at each weight update interval (ΔT) for $\mathcal{R}_{Fi}^{Fj}$ and $\mathcal{R}_{Fi}^{L}$ are calculated using (16) and (17) as 4×10^{-4} and $7.7{\times}10^{-4}$, respectively. Consequently, $\mathcal{R}_{max}^{G} = \max(\mathcal{R}_{Fi}^{Fj}, \mathcal{R}_{Fi}^{L}) = 7.7\times 10^{-4}$. The Ψ^S for the follower section in the network is $\left[\frac{4{\times}10^{-4}}{7{\times}10^{-4}}\right] 15.5 = 8.0519$, and for the leader section, it is $\left[\frac{7{\times}10^{-4}}{7{\times}10^{-4}}\right] 15.5 = 15.5$. The η and β for the follower section within the network are $\frac{\Delta R_{max}^{G}\ln(2^{\lambda^2}-1)}{\lambda\Psi^S \log(2)} = 0.0011$ and $\beta = \frac{\ln(2^{\lambda^2}-1)}{\Psi^S} = 2.152$, respectively. For the leader section, these values are $\frac{\Delta R_{max}^{G}\ln(2^{\lambda^2}-1)}{\lambda\Psi^S \log(2)} = 5.719 \times 10^{-4}$ and $\beta = \frac{\ln(2^{\lambda^2}-1)}{\Psi^S} = 1.118$, respectively.

The visual patterns observed in the synaptic weights matrix in Figs. 14 and 15, specifically, the gradual increases and decreases in values across weights, directly result from applying Gaussian membership functions for encoding. As illustrated in the heatmap visualization, regions of higher values denote areas closer to the function's center, where the degree of membership peaks. Conversely, areas of lower values reflect points moving away from the center, where the membership degree decreases according to the Gaussian distribution's tails.

Since the reward coefficients for followers and leaders are different, their maximum allowed synaptic weights are also different. The proposed method for controlling the unbounded growth of synaptic weights has successfully stabilized the network.

Fig. 15 shows the synaptic weights after the reward change. In this case, since the reward coefficients are changed, the η and β values are changed for the represented sub-layers, and the proposed method has helped the R-STDP algorithm to adjust the weights based on the new situation in the environment.

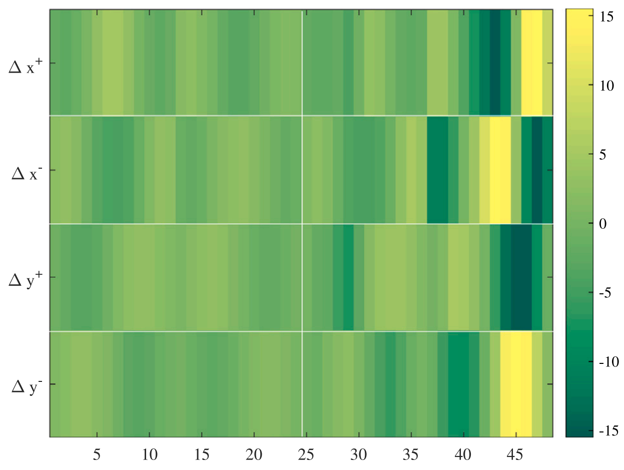


Fig. 14. Synaptic Weights before Reward change.

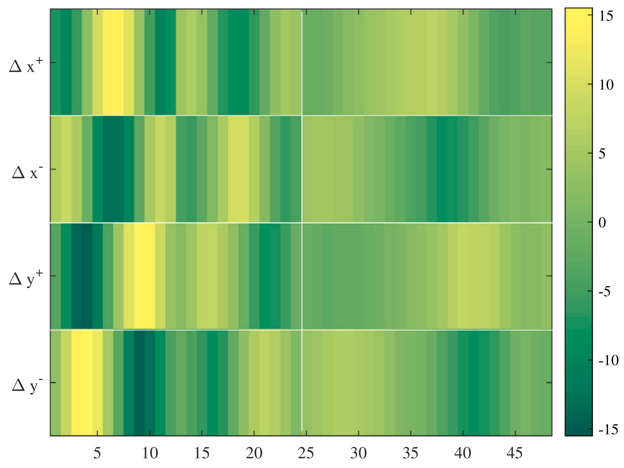


Fig. 15. Synaptic Weights after Reward change.

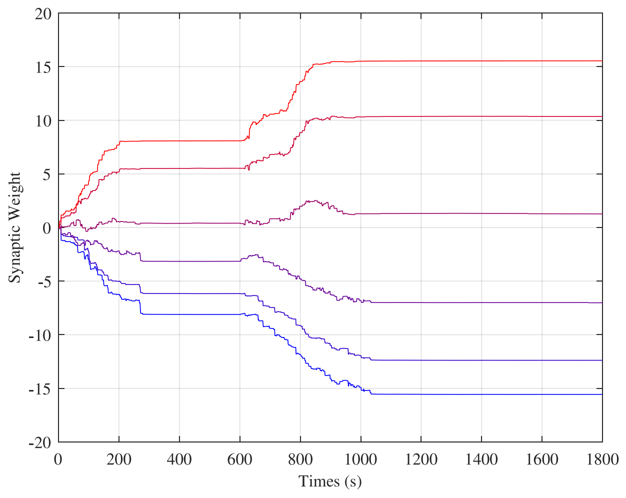


Fig. 16. Synaptic weights increase after reward change.

Figs. 16 and 17 show the changes in the synaptic weights before and after the reward change for each section of the neural network.

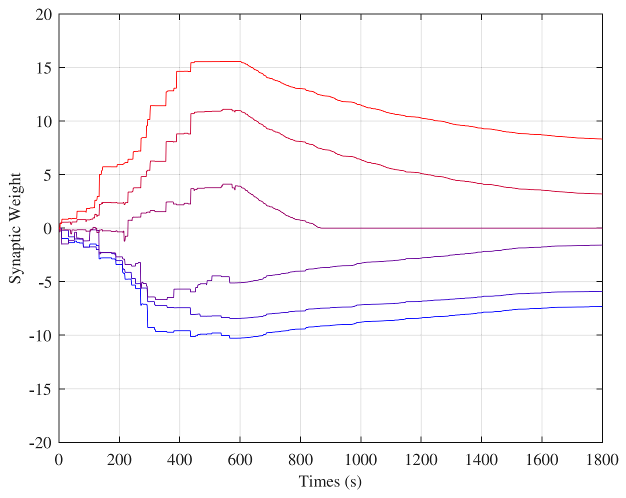


Fig. 17. Synaptic weights decrease after reward change.

According to the figures, the maximum synaptic weight converges to the maximum threshold defined based on each section's reward value.

4.2. Simulation with FL and R-CSE

In this section, the proposed aggregation algorithm is tested. In this case, the agents only send their models when the Euclidean distance between the current and previously published model or the latest global model reaches a threshold. In the first phase, the simulation was done in 600 s, and the agents learned to follow the leader. The threshold for publishing the agents' and server models was 0.0005 and 0.00051, respectively. The reason for choosing the server's threshold higher than the agents' is that as soon as the first agent sends its model to the server, the Euclidean distance between the current and previously published model by the server reaches 0.0005, and the server distributes the model immediately. Therefore, the serve's threshold is set higher than the agents' threshold, so it waits for the other agents to send their models.

Fig. 18 shows the distances between agents and the leader before the reward change. According to the figure, the agents converge to the solution faster than the non-federated learning scenario without any error. Fig. 19 shows the simulation results for the reward change scenario. According to the figure, the proposed event-triggered FL method has improved the learning performance so that the swarm converges to the solution in 6 s.

To verify the method's effectiveness, the number of agents is increased to 7. Figs. 20 and 21 show the performance of the proposed method in formation flying. According to Fig. 21, the agents can fly around the leader without colliding with each other. Because of the complexity of the environment in this case, it takes more time (around 22 s) for the agents to stabilize the formation.

As shown in Fig. 22, the norm of the synaptic weights can represent the changes in the synaptic weights due to the change in the environment during the training process, which can be used to adjust the learning process in SNNs. In the proposed event-triggered FL method, agents communicate with the Central Server (leader) during training. According to Fig. 23, the aggregation step time is small at the beginning of the training and increases as the SNN models converge to the final solution. The rate of change of the norm of the synaptic weights determines the communication sample time of the aggregation process, which results in small aggregation intervals when the change rate is high and larger intervals when it reduces.

Therefore, the aggregation frequency is very high initially, and it reduces after the change in synaptic weights goes to zero because of

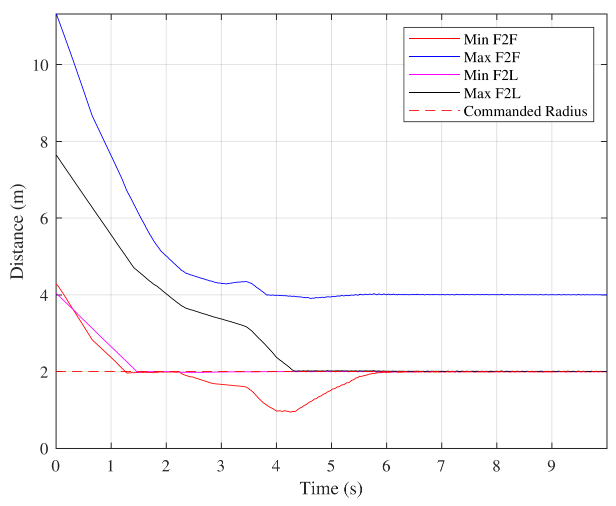


Fig. 18. Distance measurements during the test phase before reward change in the proposed event-triggered FL method.

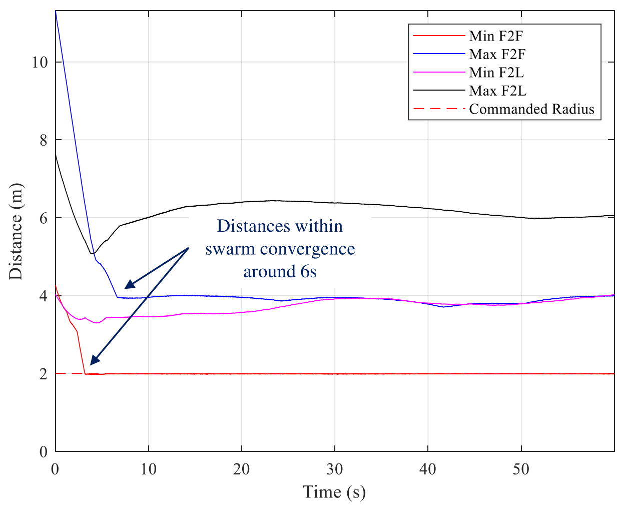


Fig. 19. Distance measurements during the test phase after reward change in the proposed event-triggered FL method.

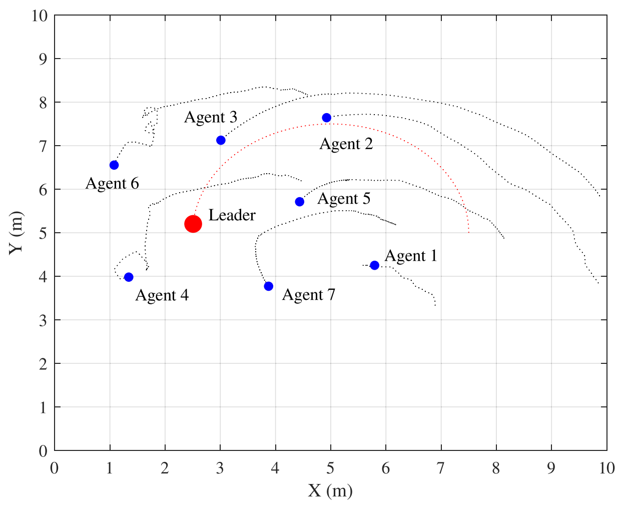


Fig. 20. Formation flying of 7 agents in the proposed event-triggered FL method.

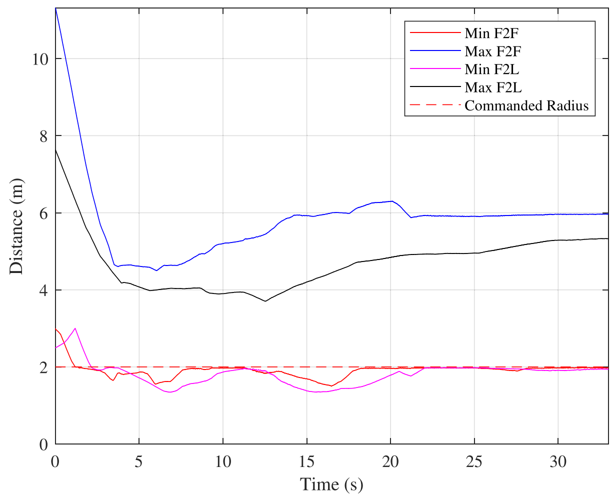


Fig. 21. Distances between 7 agents and the leader in the proposed event-triggered FL method.

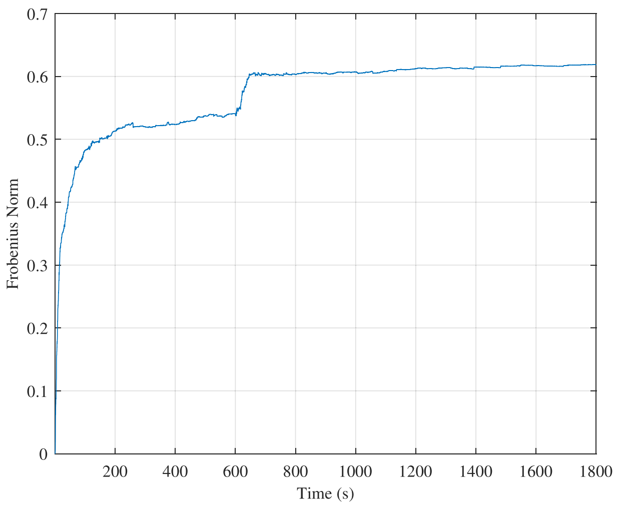


Fig. 22. Frobenius norm of the Agent 1 during the learning phase. The reward changes for the Leader after 600 s.

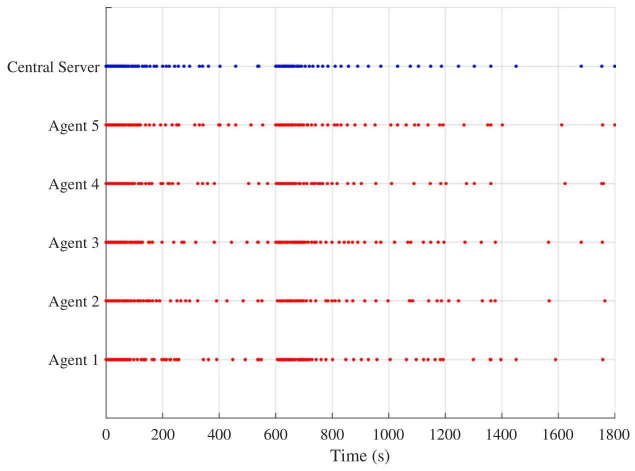


Fig. 23. Communication times for agents and the Central Server (Leader). Red and blue dots show the times that agents and the Central Server have sent their model, respectively.

Table 3
Comparative performance analysis of the proposed aggregation Algorithm+R-CSE and R-CSE.

	FL+R-CSE	R-CSE
Learning Time - Training Phase (s)	261.92	554.71
Max Distance Convergence Error - Test Phase (%)	0.71	1.95
Convergence Time for Distance - Test Phase (s)	5.81	8.94

the learning rate in (19). Also, after the reward changes and the leader becomes an obstacle after 600 s, the Euclidean distance between the converged and current models increases and reaches the threshold. As soon as the first agent sends its model to the server, the aggregation process starts again, and the agents adjust the associated synaptic weight.

Table 3 compares the results and focuses on three critical metrics: learning time, maximum error after convergence, and convergence time. The proposed FL algorithm demonstrates significant improvements in terms of efficiency and accuracy, as evidenced by its considerably shorter learning and convergence times and a notable reduction in error after convergence.

Our approach advances beyond traditional FL frameworks by introducing an innovative event-triggered aggregation mechanism tailored for the real-time adaptive capabilities inherent in SNNs. This method contrasts with those seen in literature, where aggregation weights are primarily calculated based on rewards [52]. Instead, our model considers both the time of arrival and the substantive content of each model, quantified through the Euclidean norm of the synaptic weights, to determine the aggregation weight, providing a nuanced approach to model integration.

Additionally, while other studies address challenges related to abrupt model changes due to global model updates using fixed aggregation intervals, our model inherently avoids these disruptions through its event-triggered mechanism [53,54]. This is evident from the smooth transitions observed in the plot of the Euclidean norm of each agent, illustrating a more stable model evolution without the sudden changes typically associated with periodic global updates. In our framework, agents autonomously decide the optimal time to send their updates based on real-time changes, effectively resolving issues related to model selection and synchronization prevalent in systems with predetermined aggregation schedules.

Moreover, most of the papers employ methods like Deep Q-Network (DQN) requiring a data buffer for training [55]; our use of an SNN facilitates an online learning paradigm. This online method allows our model to continually learn and adapt without storing data, enhancing efficiency and suitability for real-time applications.

Through these distinctions, our approach improves communication efficiency and learning adaptability and provides a robust solution for dynamic and distributed learning environments, setting a new standard for deploying federated learning in multi-agent systems.

5. Conclusion

In this paper, we presented a comprehensive approach addressing the challenges of uncontrolled growth in synaptic weights and the limited responsiveness of R-STDP to real-time changes within SNNs. Our proposed solution integrates the R-CSE method with a dynamic aggregation interval in FL and significantly reduces learning time while improving performance. The R-CSE method introduces a novel mechanism to manage the unbounded growth of synaptic weights by dynamically adjusting the decay rate through the SoftPlus function. This adjustment is sensitive to the learning stages and rewards changes, ensuring synaptic weight adjustments remain responsive over time. By addressing the challenge of synaptic weight saturation, the R-CSE method facilitates a balanced approach to weight adjustment, preventing network saturation and promoting continuous learning adaptability. We introduce a novel approach that uses FL in SNN and employs the Frobenius norm to adjust weighted aggregation in FL. Additionally, we include weight decay proportional to the time elapsed since an agent's last model publication. This improves the efficiency and responsiveness of the learning process. Our model's dynamic nature of the model aggregation time adjusts based on the Euclidean norm. This metric measures the distance between the weight matrices of the agents and the server, determining reduced intervals for model publication. Our results show that the proposed aggregation method significantly accelerates agents' learning while increasing the accuracy of the swarm in following the commanded distance. Moreover, the dynamic aggregation interval effectively reduces communication overhead between the agents and the central server, particularly after model convergence. This reduction is critical when communication bandwidth is limited or costly. This approach is particularly advantageous in 5G networks, where efficient bandwidth use can enhance the overall throughput and reduce latency in real-time applications. Moreover, the adaptive use of communication resources aligns with the scalable and flexible infrastructure of 5G, optimizing network performance even during peak demand periods.

CRediT authorship contribution statement

Mohammad Tayefe Ramezanlou: Writing – original draft, Software, Methodology, Formal analysis, Conceptualization. **Howard Schwartz:** Writing – review & editing, Supervision, Funding acquisition. **Ioannis Lambadaris:** Writing – review & editing, Supervision, Funding acquisition. **Michel Barbeau:** Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We acknowledge financial support from Ericsson, Mitacs, Canada, and NSERC (Natural Sciences and Engineering Research Council) of Canada.

Data availability

Data will be made available on request.

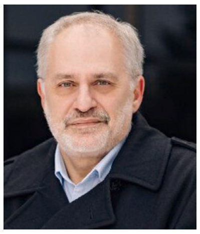
References

[1] S.K. Lee, Distributed deformable configuration control for multi-robot systems with low-cost platforms, Swarm Intell 16 (3) (2022) 169–209.
[2] M. Zhao, J. Xi, L. Wang, K. Xia, Y. Zheng, Edge-based adaptive secure consensus for nonlinear multiagent systems with communication link attacks, Neurocomputing (2023).
[3] M. Yu, J. Xia, J.-e. Feng, S. Fu, H. Shen, Leader–follower output consensus of multiagent systems over finite fields, Neurocomputing 550 (2023).
[4] Z. Peng, Y. Jiang, L. Liu, Y. Shi, Path-guided model-free flocking control of unmanned surface vehicles based on concurrent learning extended state observers, IEEE Trans. Systems, Man, Cybern: Syst (2023).
[5] C. Zheng, K. Lee, Consensus decision-making in artificial swarms via entropy-based local negotiation and preference updating, Swarm Intell (2023) 1–21.
[6] H. Yang, K.-Y. Lam, L. Xiao, Z. Xiong, H. Hu, D. Niyato, H. Vincent Poor, Lead federated neuromorphic learning for wireless edge artificial intelligence, Nature Commun 13 (1) (2022) 42–69.
[7] M. Chen, H.V. Poor, W. Saad, S. Cui, Convergence time optimization for federated learning over wireless networks, IEEE Trans. Wireless Commun. 20 (4) (2020) 2457–2471.
[8] Q. Wu, X. Chen, T. Ouyang, Z. Zhou, X. Zhang, S. Yang, J. Zhang, Hiflash: Communication-efficient hierarchical federated learning with adaptive staleness control and heterogeneity-aware client-edge association, IEEE Trans. Parallel Distrib. Syst. 34 (5) (2023) 1560–1579.
[9] M. Chen, H.V. Poor, W. Saad, S. Cui, Convergence time optimization for federated learning over wireless networks, IEEE Trans. Wireless Commun. 20 (4) (2020) 2457–2471.
[10] Z. Yuan, Z. Wang, X. Li, L. Li, L. Zhang, Hierarchical trajectory planning for narrow-space automated parking with deep reinforcement learning: A federated learning scheme, Sensors 23 (8) (2023).
[11] H. Yang, K.-Y. Lam, L. Xiao, Z. Xiong, H. Hu, D. Niyato, H. Vincent Poor, Lead federated neuromorphic learning for wireless edge artificial intelligence, Nat. Commun 13 (1) (2022).
[12] R. Gupta, T. Alam, Survey on federated-learning approaches in distributed environment, Wirel. Personal Commun 125 (2) (2022) 1631–1652.
[13] W. Huang, T. Li, D. Wang, S. Du, J. Zhang, T. Huang, Fairness and accuracy in horizontal federated learning, Inform. Sci. 589 (2022) 170–185.
[14] M.-A. Lahmeri, M.A. Kishk, M.-S. Alouini, Artificial intelligence for UAV-enabled wireless networks: A survey, IEEE Open J. Commun Soc 2 (2021) 1015–1040.
[15] K. Tuyls, K. Verbeeck, T. Lenaerts, A selection-mutation model for q-learning in multi-agent systems, in: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, 2003, pp. 693–700.
[16] W. Barfuss, J.F. Donges, J. Kurths, Deterministic limit of temporal difference reinforcement learning for stochastic games, Phys. Rev. E 99 (4) (2019).
[17] Z. Wang, C. Mu, S. Hu, C. Chu, X. Li, Modelling the dynamics of regret minimization in large agent populations: a master equation approach, in: IJCAI, 2022, pp. 534–540.
[18] A. Jarwan, M. Ibnkahla, Edge-based federated deep reinforcement learning for IoT traffic management, IEEE Internet Things J. 10 (5) (2022) 3799–3813.
[19] W. Huang, J. Liu, T. Li, T. Huang, S. Ji, J. Wan, Feddsr: Daily schedule recommendation in a federated deep reinforcement learning framework, IEEE Trans. Knowl. Data Eng. 35 (4) (2021) 3912–3924.
[20] A.B. Mansour, G. Carenini, A. Duplessis, D. Naccache, Federated learning aggregation: New robust algorithms with guarantees, in: 2022 21st IEEE International Conference on Machine Learning and Applications, ICMLA, 2022, pp. 721–726.
[21] Y. Yang, W. He, S. Li, Refined dynamic event-triggering cluster consensus of multiagent systems with fixed/switching topology, IEEE Trans. Cybern. (2022).
[22] H. Lu, J. Liu, Y. Luo, Y. Hua, S. Qiu, Y. Huang, An autonomous learning mobile robot using biological reward modulate STDP, Neurocomputing 458 (2021) 308–318.
[23] J. Liu, H. Lu, Y. Luo, S. Yang, Spiking neural network-based multi-task autonomous learning for mobile robots, Eng. Appl. Artif. Intell. 104 (2021) 104–362.
[24] D. Chu, H. Le Nguyen, Constraints on hebbian and STDP learned weights of a spiking neuron, Neural Netw. 135 (2021) 192–200.
[25] D. Antonov, K. Sviatov, S. Sukhov, Continuous learning of spiking networks trained with local rules, Neural Netw. 155 (2022) 512–522.
[26] D. Xing, J. Li, T. Zhang, B. Xu, A brain-inspired approach for collision-free movement planning in the small operational space, IEEE Trans. Neural Netw. Learn. Syst. 33 (5) (2022) 2094–2105.
[27] J. Pérez, J.A. Cabrera, J.J. Castillo, J.M. Velasco, Bio-inspired spiking neural network for nonlinear systems control, Neural Netw. 104 (2018) 15–25.
[28] G. Liu, W. Deng, X. Xie, L. Huang, H. Tang, Human-level control through directly trained deep spiking Q-networks, IEEE Trans. Cybern. (2022).
[29] S.A. Lobov, A.N. Mikhaylov, M. Shamshin, V.A. Makarov, V.B. Kazantsev, Spatial properties of STDP in a self-learning spiking neural network enable controlling a mobile robot, Front. Neurosci 14 (2020).
[30] C. Teeter, R. Iyer, V. Menon, N. Gouwens, D. Feng, J. Berg, A. Szafer, N. Cain, H. Zeng, M. Hawrylycz, et al., Generalized leaky integrate-and-fire models classify multiple neuron types, Nat. Commun 9 (2018).
[31] J. Shen, Y. Zhao, J.K. Liu, Y. Wang, Hybridsnn: Combining bio-machine strengths by boosting adaptive spiking neural networks, IEEE Trans. Neural Netw. Learn. Syst. (2021).
[32] Z. Bing, C. Meschede, G. Chen, A. Knoll, K. Huang, Indirect and direct training of spiking neural networks for end-to-end control of a lane-keeping vehicle, Neural Netw. 121 (2020) 21–36.
[33] X. Cheng, T. Zhang, S. Jia, B. Xu, Meta neurons improve spiking neural networks for efficient spatio-temporal learning, Neurocomputing 531 (2023) 217–225.
[34] P. Bertens, S.-W. Lee, Network of evolvable neural units can learn synaptic learning rules and spiking dynamics, Nat. Mach. Intell. 2 (12) (2020) 791–799.
[35] M. Białas, J. Mańdziuk, Spike-timing-dependent plasticity with activation-dependent scaling for receptive fields development, IEEE Trans. Neural Netw. Learn. Syst. 33 (10) (2021) 5215–5228.
[36] J.L. Lobo, J. Del Ser, A. Bifet, N. Kasabov, Spiking neural networks and online learning: An overview and perspectives, Neural Netw. 121 (2020) 88–100.
[37] D. Haşegan, M. Deible, C. Earl, D. D'Onofrio, H. Hazan, H. Anwar, S.A. Neymotin, Training spiking neuronal networks to perform motor control using reinforcement and evolutionary learning, Front. Comput. Neurosci 16 (2022).
[38] Y. Venkatesha, Y. Kim, L. Tassiulas, P. Panda, Federated learning with spiking neural networks, IEEE Trans. Signal Process. 69 (2021) 6183–6194.
[39] Y. Wang, S. Duan, F. Chen, Efficient asynchronous federated neuromorphic learning of spiking neural networks, Neurocomputing 557 (2023).
[40] S.A. Tumpa, S. Singh, M.F.F. Khan, M.T. Kandemir, V. Narayanan, C.R. Das, Federated learning with spiking neural networks in heterogeneous systems, in: IEEE Computer Society Annual Symposium on VLSI, ISVLSI, 2023, pp. 1–6.

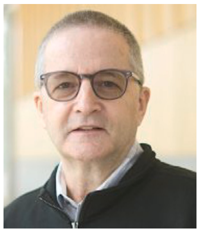
[41] Y.-H. Liu, X.-J. Wang, Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron, J. Comput. Neurosci 10 (2001) 25–45.
[42] L. Long, F. Guoliang, A review of biologically plausible neuron models for spiking neural networks, AIAA Infotech@ Aerospace (2010).
[43] M. Tayefe Ramezanlou, H. Schwartz, I. Lambadaris, M. Barbeau, S.H.R. Naqvi, Learning a policy for pursuit-evasion games using spiking neural networks and the STDP algorithm, in: IEEE International Conference on Systems, Man, and Cybernetics, SMC, 2023, pp. 1918–1925.
[44] K. Kasaura, S. Miura, T. Kozuno, R. Yonetani, K. Hoshino, Y. Hosoe, Benchmarking actor-critic deep reinforcement learning algorithms for robotics control with action constraints, IEEE Robot. Autom. Lett. (2023).
[45] J. Eßerer, N. Bach, C. Jestel, O. Urbann, S. Kerner, Guided reinforcement learning: A review and evaluation for efficient and effective real-world robotics, IEEE Robot. Autom. Mag. (2022).
[46] P. Qi, D. Chiaro, A. Guzzo, M. Ianni, G. Fortino, F. Piccialli, Model aggregation techniques in federated learning: A comprehensive survey, Future Gener. Comput. Syst. 150 (2024) 272–293.
[47] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, 2017, pp. 1273–1282.
[48] Z. Wang, Z. Peng, X. Fan, Z. Wang, S. Wu, R. Yu, P. Yang, C. Zheng, C. Wang, Fedave: Adaptive data value evaluation framework for collaborative fairness in federated learning, Neurocomputing 574 (2024).
[49] J. Wang, Q. Liu, H. Liang, G. Joshi, H.V. Poor, Tackling the objective inconsistency problem in heterogeneous federated optimization, Adv. Neural Inform Process. Syst 33 (2020) 7611–7623.
[50] A. AbdelAty, M. Fouda, A. Eltawil, On numerical approximations of fractional-order spiking neuron models, Commun. Nonlinear Sci. Numer. Simul. 105 (2022).
[51] M. Dampfhoffer, T. Mesquida, A. Valentian, L. Anghel, Are SNNs really more energy-efficient than ANNs? An in-depth hardware-aware study, IEEE Trans. Emer. Topics Comput. Intell 7 (3) (2022) 731–741.
[52] T.M. Ho, K.-K. Nguyen, M. Cheriet, Federated deep reinforcement learning for task scheduling in heterogeneous autonomous robotic system, IEEE Trans. Autom. Sci. Eng. 21 (1) (2022) 528–540.
[53] S. Na, T. Rouček, J. Ulrich, J. Pikman, T. Krajník, B. Lennox, F. Arvin, Federated reinforcement learning for collective navigation of robotic swarms, IEEE Trans. Cognit Develop Syst 15 (4) (2023) 2122–2131.
[54] M. Krouka, A. Elgabli, C.B. Issaid, M. Bennis, Communication-efficient and federated multi-agent reinforcement learning, IEEE Trans. Cognit. Commun Netw 8 (1) (2021) 311–320.
[55] R. Luo, W. Ni, H. Tian, J. Cheng, Federated deep reinforcement learning for RIS-assisted indoor multi-robot communication systems, IEEE Trans. Veh. Technol. 71 (11) (2022) 12321–12326.

Mohammad Tayefe Ramezanlou is a Ph.D. candidate at the Department of Systems and Computer Engineering at Carleton University, Canada. He received a B.Sc. in Mechanical Engineering in 2016 and an M.Sc. in Mechatronics Engineering in 2018 from the University of Tabriz, Iran. His research interests are neuro-robotics, neuromorphic computation, Federated Learning (FL), and multi-agent systems.

Professor Howard M. Schwartz received his B.Eng. degree from McGill University, Montreal, Canada, in June 1981 and his M.Sc. degree and Ph.D. from the Massachusetts Institute of Technology, Cambridge, MA, in 1982 and 1987, respectively. He is a Professor in the Department of Systems and Computer Engineering at Carleton University. His research interests include adaptive and intelligent control systems, robotics and process control, system modeling, and system identification. His most recent research is in multiagent learning with applications to teams of drones and mobile robots.



Professor Ioannis Lambadaris received his M.Sc. degree in engineering from Brown University, Providence, RI, USA, in 1985 and his Ph.D. in electrical engineering from the University of Maryland, College Park, MD, USA, in 1991. He was employed as a Research Associate at Concordia University, Montreal, QC, Canada, from 1991 to 1992. He joined the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada, in September 1992, where he is currently a Chancellor's Professor. His research interests include applied stochastic processes, stochastic control, queueing theory, and their application for modeling/simulation and performance analysis of computer communication networks. He has made numerous contributions to Quality of Service (QoS) control for IP networks, resource allocation in optical networks, and optimal routing and flow control in ad hoc wireless systems. His recent research focuses on hardware and software solutions for mobile applications and platforms, including biomedical, remote monitoring, and security applications.



Professor Michel Barbeau is a professor of Computer Science. He got a Bachelor's (Universite de Sherbrooke, Canada '85), a master's (Universite de Montreal, Canada '87 & '91), and a Ph.D. (Universite de Montreal, Canada '91), all in Computer Science. From '91 to '99, he was a professor at Universite de Sherbrooke. During the '98–'99 academic year, he was a visiting researcher at the University of Aizu, Japan. Since 2000, he has worked at Carleton University, School of Computer Science, Canada. His main topic of expertise is computer networks: architecture and protocols. Specific research interests include underwater communications and networks, flying drone networks, quantum networks, cyber–physical security, postquantum cryptography, and network control systems.