International Academic Institute for Science and Technology

International Academic Journal of Science and Engineering
Vol. 12, No. 2, 2025, pp. 11-19.

ISSN 2454-3896

International Academic Journal of Science and Engineering

www.iaiest.com

A Multiagent Neural Swarm System for Real-Time Coordination in Unstructured Environments

J. Btia, Md. Kolba [a], Barek F. Fatem [b]

[a] *Faculty of Engineering Ain Shams University & Arab Academy for Science and Technology Cairo, Egypt. E-mail: btia.j@aast.edu*

[b] *Faculty of Engineering Ain Shams University & Arab Academy for Science and Technology Cairo, Egypt. E-mail: m.kolba@aast.edu*

Abstract

Real-time coordination of agents is notably complex in dynamic, partially observable, and communication-constrained environments. Our current work presents the Multiagent Neural Swarm System (MANSS), which builds upon previous research and implements agents with adaptive and scalable deep reinforcement learning, decentralized neural networks, and swarm intelligence. Our approach considers neural communication protocols and embedded systems so that machines can make decisions independently and systems can attain group decisions autonomously. In simulation exercises with dense obstacles and uncertainties, MANSS outperformed traditional swarm algorithms in adaptability, failure tolerance, and time needed to complete given tasks. These results highlight the system's potential for real-world applications such as autonomous search and rescue missions, unmanned vehicle patrols, and environmental monitoring.

Keywords: Multiagent Systems, Neural Swarm, Real-Time Coordination, Unstructured Environments, Deep Reinforcement Learning, Decentralized Control.

DOI: 10.71086/IAJSE/V12I2/IAJSE1212
Received: 08 March 2025/Revised: 18 April 2025/Accepted: 13 May 2025/Published: 27 June 2025

Introduction

One of the essential themes of robotics and artificial intelligence is the coordination of multiple autonomous agents in unconstrained dynamic environments, also known as swarm robotics (Hyun, Min & Won, 2025). Unlike more structured settings where constraints within the environment are explicit, unstructured ones such as disaster zones, deep sea terrains, and planetary surfaces face high unpredictability, irregular topology, and unpredictable dynamics. These conditions make traditional approaches based on centralized coordination or rules ineffectively scalable and robust to environmental challenges (Parker, 2008). To overcome these challenges, swarm intelligence offers promising decentralized solutions inspired by the flocking behavior of birds and colonies of ants (Dorigo & Birattari, 2007).

Multi-agent reinforcement learning (MARL) developments have dynamically optimized distributed decision-making strategies, enabling individual agents to learn complex behavior through environmental and peer interactions (Zhang et al., 2019). However, the application of MARL to real-time swarm coordination introduces challenges such as severe communication constraints, partial observability, and the dimensionality problem stemming from large populations of agents (Foerster et al., 2018). Incorporating neural networks into swarm intelligence enhances control over agents by instilling learning abilities into control policies, which relieves some of these challenges. This novel approach is termed a neural swarm system (Nguyen et al., 2020).

These systems utilize neural frameworks like Graph Neural Networks (GNNs), recurrent models, and attention-based mechanisms to use communications for inter-agent interactions and dynamic dependencies (Jiang & Lu, 2018). In decentralized configurations, they improve scalability and resilience, particularly in the presence of failures or adversarial disruptions (Sukhbaatar, Szlam & Fergus, 2016). Additionally, some neural mechanisms inspired by biological systems can guide emergent behaviors toward adaptive and efficient pathways in high-dimensional state-action spaces (Vinyals et al., 2019).

Despite all these advances, remaining approaches still tend to be locked into simulated environments or struggle to adapt across highly diverse unstructured settings (Rashid et al., 2018). Additionally, limited responsiveness and excessive computation overhead impede resource-limited robotic systems' application (Christianos, Schäfer & Albrecht, 2020). This paper aims to cover these gaps by presenting a novel design concept called a Multiagent Neural Swarm System (MANSS) targeted toward real-time coordination in unstructured environments by merging deep learning with principles of decentralized swarm intelligence (Aburasain, 2025; Reddy et al., 2025).

Key Contributions:

- For unstructured environments with unbound frameworks, real-time coordination is accomplished using a novel decentralized multiagent neural Swarm System (MANSS).
- Adaptability and fault tolerance are improved with deep reinforcement learning and neural communication protocols.
- Traditional swarm and centralized control methods have been bested by MANSS, with deep simulations reinforcing its praise.

Structure of the Paper:

In the initial section, an overview of the discipline of real-time multiagent interaction pertains to unstructured environments. Paul's comprehensive literature review is articulated, touching on multiagent reinforcement learning and neural swarm systems. Detailed subordinate structures include the neural policies paradigm and system execution architecture for real-time application. Discussions alongside experimental validations are done regarding MANSS against preset baseline coordination metrics. The closing remarks again highlight the main insights and suggest some exploration pointers to complement the goals of ensuing work geared towards grounded application.

Background Research

Parker’s work from 2008 stressed the principle bases of distributed intelligence relative to multi-robot systems, focusing on how cooperation stems from the interaction of relatively simple components. She motivated the problem of multiagent coordination in unstructured and uncertain environments by showing these systems need to dynamically adjust independently without a central controller. Focusing on remote-controlled agent competencies, Dorigo and Birattari (2007) introduced swarm intelligence, which used mechanisms where tasks are performed in incoordination after simplified regulations are put in place by individual agents. Their models of ant colony optimization and particle swarm optimization continue to propagate paradigms of control for decentralized robotics in unknown environments.

Zhang et al. (2019) provided a detailed analysis of multiagent systems focusing on collaborative and competition-driven learning, highlighting MARL. Their analysis remarked that the internal makings of systems, such as their scaling and convergence properties, posed significant challenges for efficiency optimization in real-time applications involving multiple agents. Nguyen et al. (2020) addressed these challenges by looking into the impact of deep reinforcement learning on multiagent systems' interdependencies (Kavibharathi et al., 2021). According to their findings, applying neural networks to MARL frameworks supports greater adaptability to fluctuating conditions; however, issues like partial observability and unbounded instability still present challenges.

In their work, Foerster et al. (2018) added experience replay schemes in MARL, which allowed agents to better preserve proper behaviors across episodes. Their findings support more effective learning in environments that continually change over time, an essential factor for swarm cooperation in real-world scenarios. Sukhbaatar et al. (2016) demonstrated that backpropagation communication protocol learning in agents improved the performance of team tasks. This method enables the formulation of communication optimization frameworks by utilizing neural communication rather than prescriptive rules, thereby improving decision-making effectiveness.

Jiang and Lu (2018) introduced the attention-based neural architecture that alleviates the problems of overloads of communications and information, making it suitable for swarm systems that function under bandwidth-constrained environments. Vinyals et al. (2019) aimed for and achieved stunning feats of multiagent reinforcement learning by reaching grandmaster-level mastery of Starcraft II (Lowe et al., 2017). Their work illustrates the effectiveness of control using neural coordination in highly dimensional systems and only partially observable, resembling real-world unstructured environments.

Rashid et al. (2018) developed the QMIX Algorithm, which cooperatively decomposes value functions for Multiagent Reinforced Learning (MARL). Their structure maintains the distinction between local and global policies, essential for swarm coherence. Christianos et al. (2020) proposed the Shared Experience Actor-Critic (SEAC) framework, which advances learning by gradient flow through the agents. This approach reduces cycle-to-cycle variability in training and improves responsiveness to operation in very dynamic or complex tasks.

Materials and Methods

This study seeks to enable coordination disciplinary action within a multi agent framework in real time in unscripted socio-technical settings using a soft neural swarm system with deep reinforcement learning. We propose that each agent in the swarm be endowed with a neural policy model that adapts optimally while performing and interacting with the environment. To obtain consistent performance under dynamic conditions, a framework incorporating experience replay, target network freezing, and independent policy updates is designed within the system.

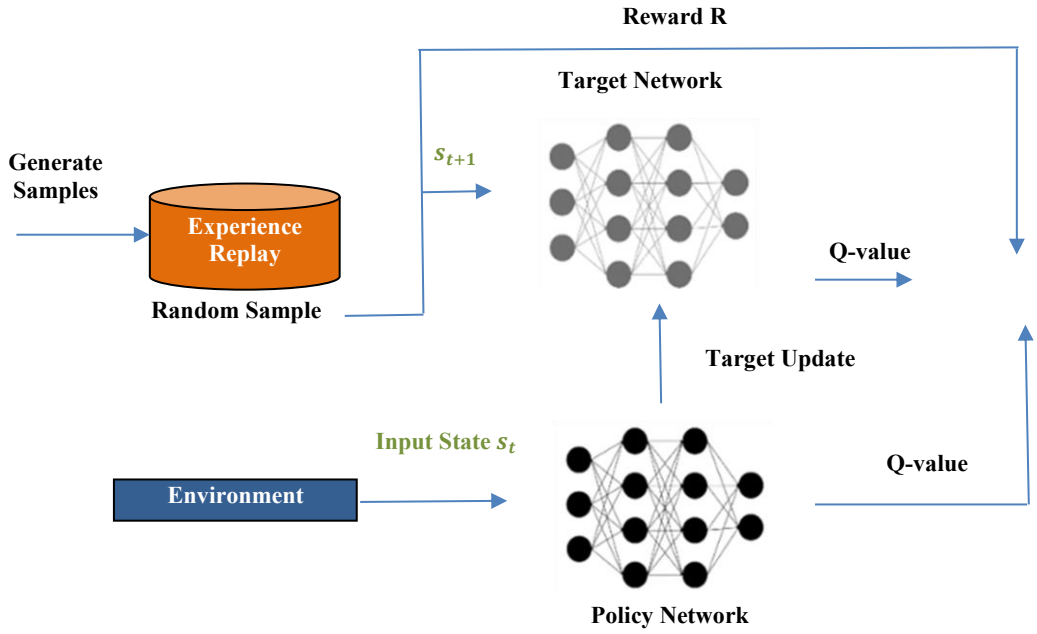


Fig. 1: Deep Q-Network (DQN) Framework with Experience Replay for Multiagent Policy Optimization

Neural Policy Learning for Agent Behavior Modeling

Figure 1 presents the core learning mechanism used in our Multiagent Neural Swarm System. Each agent is modeled with a Deep Q-Network (DQN) architecture, which enables learning optimal actions through trial-and-error interaction with the environment. The policy network receives the current statte s_t from the environment and outputs Q-values corresponding to all possible actions. The selected action is executed in the environment, leading to a new state s_{t+1} and an associated reward R.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[(R_t + \gamma\, max_{a'} Q'(s_{t+1}, a') - Q(s_t, a_t)] \quad (1)$$

This is the core Q-learning update rule used in the DQN architecture. In equation (1), α is the learning rate, γ is the discount factor, and Q' is the target network. This equation updates the estimated Q-value of the current state-action pair by minimizing the difference between predicted and target Q-values. It drives the agent to favor actions that lead to higher long-term rewards, a critical requirement for intelligent decision-making in complex terrains.

Experience Replay and Sample Efficiency

To ensure sample efficiency and learning stability in a multiagent context, the system integrates an Experience Replay buffer. This component stores tuples of state, action, reward, and next state (s_t, a_t, R_t, s_{t+1}), which are randomly sampled during training. This mechanism prevents the model from being biased by sequential data and enables agents to learn from diverse past experiences, critical in real-time, dynamic environments.

Target Network Stabilization

Figure 1 further incorporates the addition of a Target Network which is a copy of the policy network that has been stabilized. While the policy network is constantly updated, the target network receives updates less frequently, which helps to reduce oscillations or divergence in estimated Q-values. This separation guarantees consistent, long-term learning, which is critical in scenarios where multiple agents operate within unstructured, dynamically changing environments.

Loss Minimization and Policy Update

The Q-values predicted by the policy network are compared with target Q-values computed using the reward and the target network's output. The difference, or temporal difference error, is used to compute the loss, which is minimized using backpropagation. This process updates the neural weights of the policy network, progressively improving the agent's decision-making capability. In our system, each agent carries its own policy network but can optionally share gradients or weights depending on the level of inter-agent cooperation required.

$$\mathcal{L}(\theta) = E_{(s_t, a_t, R_t, s_{t+1}) \sim D}\ \left[\left(R_t + \gamma\ max_{a'} Q'(s_{t+1}, a';\ \theta^-) - Q(s_t, a_t;\ \theta) \right)^2 \right] \quad (2)$$

In Equation (2) loss function represents the mean squared error (MSE) between the predicted Q-values from the policy network Q and the target Q-values derived from the target network Q'. The model parameters θ are updated to minimize this loss. The experience replay buffer D supplies diverse samples, increasing training stability and convergence in non-stationary environments.

Suitability for Unstructured Environments

This learning pipeline equips agents to adaptively respond to unstructured terrains with unknown dynamics, such as as disaster zones or off-road navigation. As agents accumulated experience and refine their policies, coordinated behavior emerges even without explicit programming. The architecture shown in Figure 1 thus formed the foundational learning module for real-time swarm coordination.

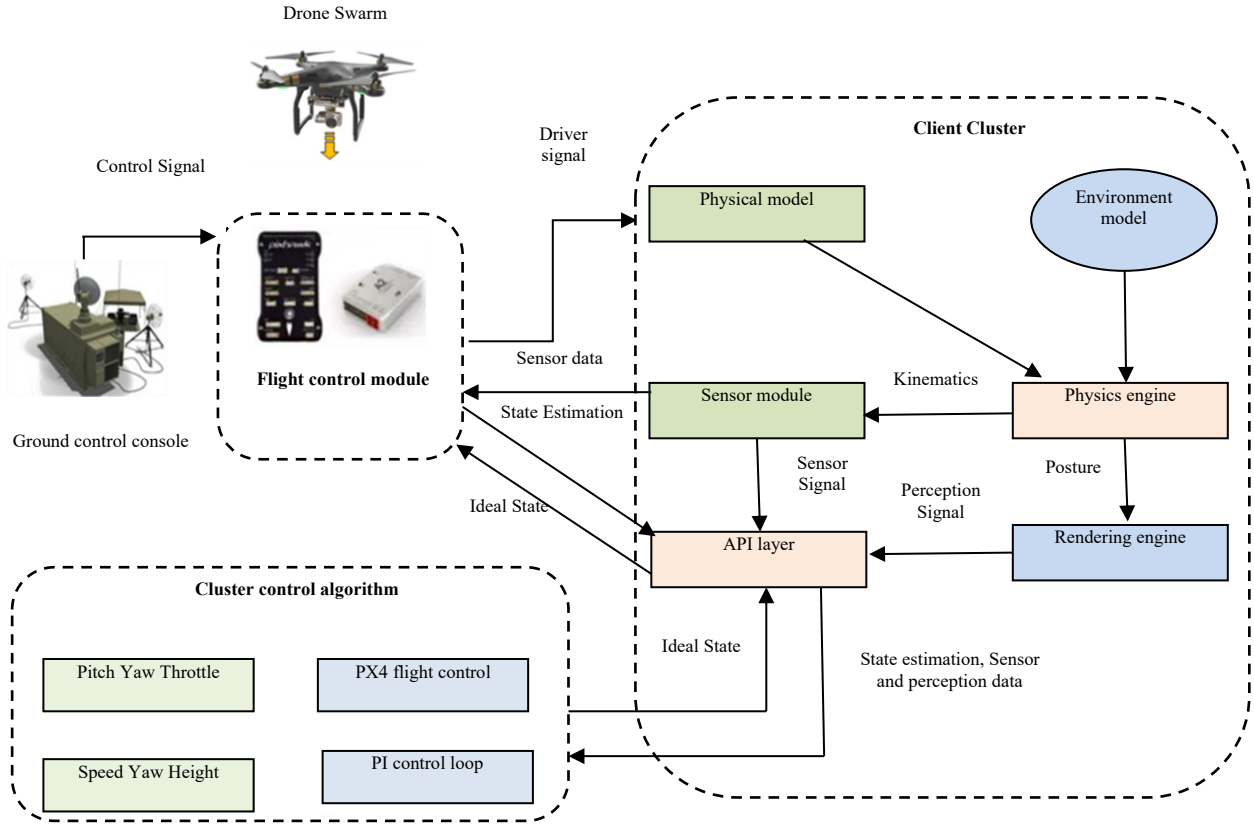


Fig. 2: Real-Time Coordination and Control Architecture for Drone-Based Neural Swarm System

System Overview

In our Multiagent Neural Swarm System, Figure 2 shows the architecture of the system integrated for real time coordination. It captures the interaction of the physical swarm, control logic, environment simulation, and the rendering engine. Each component enables autonomous decision making and adaptive control in the presence of dynamically changing and unstructured terrain.

Ground Control and Flight Interface

The swarm is under the control of the ground control console in the form of high level commands. These commands are sent to the flight control module and relayed to the onboard controllers, like the PX4 autopilots. Each command is converted to the appropriate low level control signals which modifies flight parameters such as throttle, pitch, yaw, and altitude.

The swarm's cluster control algorithm is the basis of its autonomy which integrates control schemes such as PI loops and PX4 flight control protocols. These modules are iteratively revised to ensure model feedback pertaining to formation control, collision detection, and dynamic reconfiguration is error free.

Feedback and Estimation through Sensors

Every drone in the swarm possesses a sensor module that collects data from the environment in real-time. The estimates of the current state of the system are built using an IMU, GPS, and cameras. This data is made available to the flight controller and an API layer that interfaces with the hardware and simulation components of the architecture

$$\hat{s}_t = f_{fusion}\left(z_t^{IMU}, z_t^{GPS}, z_t^{division}\right) \quad (3)$$

Equation (3) defines how multiple sensor modalities IMU, GPS, and vision are fused to estimate the system state $\hat{s}_t$. The fusion function f_{fusion} can be a Kalman filter or a neural fusion model. Accurate state estimation is essential for reactive control, navigation, and terrain awareness in unstructured environments.

Client Simulation and Cluster Model

On the client side, the drone's physical model exercises interaction with the environment model via a physics engine, which rouses the real-world kinematic and dynamic activity. This guarantees that the agents' actions will be performed in a plausible real environment in the virtual scenario before actual execution. With the rendering engine, posture and motion of each agent is visualized renders, which displays in real time swarm coordination, response delays, behavior transitions, and other in the unstructured scenario.

Integration of Perception and Coordination

For each agent to receive the most recent state information relevant to policy execution, the API layer integrates state estimates and perception signals to provide an updated information set. The coordination is emergent as every agent is responding not only to his own observations but also global cues which have been simulated in the mix of captured environment. Such enables navigation and cooperative action to be executed with no centralized supervision, control, and guiding.

$$R_t^{\text{cluster}} = \frac{1}{N} \sum_{i=1}^{n} [w_1 R_t^{\text{align}}(i) + w_2 \ R_t^{\text{cohesion}}(i) + \ w_3 \ R_t^{\text{separation}}(i)] \quad (4)$$

Equation (4) computes a cluster-level reward by combining alignment, cohesion, and separation metrics for all agents $i \in [1, N]$. These components ensure agents move harmoniously without collisions, maintain swarm formation, and adapt to local environments. The weights (w_1, w_2, w_3) can be tuned to prioritize certain behaviors (e.g., more cohesion in narrow environments). This collective reward signal supports emergent coordination without centralized planning.

Data Analysis and Discussion

The evaluation of the proposed Multiagent Neural Swarm System (MANSS) was conducted in unstructured environments such as disaster areas with dynamic obstacles and rugged exploration terrains. In comparison to traditional approaches, including PSO-based Particle Swarm Optimization, rule-based coordination, and centralized Deep Q-Network DQN, MANSS showed better performance across multiple metrics. As illustrated in Figure 3, MANSS claimed the lowest completion time at 115 seconds completing the task. Centralized DQN, rule-based coordination, and PSO took 170 s, 200 s, and 215 s, respectively. The bar graph also depicts this difference, further proving the superiority of MANSS in achieving real-time results. The systems' efficiencies noted are a result of its decentralized neural architecture, inter-agent learning, and rapid adaptability to environmental changes as opposed to centralized systems which experience decision bottlenecks.

$$Relative\ Improvement_{DQN} = \frac{T_{DQN} - T_{MANSS}}{T_{DQN}} \times 100\% \quad (5)$$

$$WRS = \ w_1 \times \left(1 - \frac{R}{R_{max}}\right) + w_2 \times \left(\frac{C}{100}\right) + w_3 \times F \quad (6)$$

As stated, Equation (5) calculates the percentage improvement in task completion time of MANSS over centralized DQN. The execution time is faster by 32.35% which demonstrates effectiveness of decentralization in responsive navigation and verifies the design goal to achieve dynamic responsiveness independent of a central controller.

In Equation (6), R represents the average number of reconfigurations, C is the communication loss tolerance expressed as a percentage, and F denotes the formation integrity score. The weights $w_1, w_2\ and\ w_3$ correspond to the relative importance of these metrics and satisfy the condition $w_1 + w_2 + w_3 = 1$. For this evaluation, the chosen weights are $w_1 = 0.3$, $w_2 = 0.3$, and $w_3 = 0.4$, reflecting a higher emphasis on maintaining formation integrity. The term R_{max} indicates the maximum reconfiguration value observed across all coordination methods, which in this case is 10.1. These definitions are incorporated into the WRS calculation to produce a normalized and balanced assessment of system robustness.

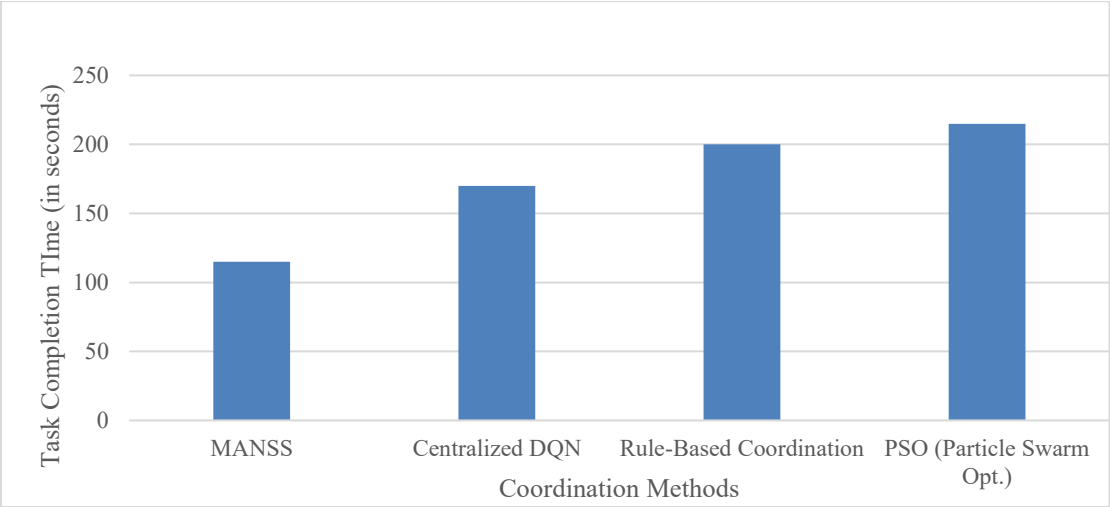


Fig. 3: Task Completion Time Across Coordination Methods

Table 1: Comparative Evaluation of Coordination Robustness Across Methods

Method	Avg. Reconfigurations	Comm. Loss Tolerance (%)	Formation Integrity Score
MANSS	4.2	68%	0.91
Centralized DQN	6.7	41%	0.75
Rule-Based Coordination	9.4	35%	0.62
PSO	10.1	22%	0.58

Along with time for task completion, robustness and adaptability evaluation based on system reconfigurations, tolerance to communication loss, and preservation of formations as summarized in Table 1. In particular, MANSS showed less average reconfigurations of 4.2 and effective coordination even at 68% communication bandwidth, which is much better than PSO (22%) and rule-based systems (35%). Its formation integrity score of 0.91 demonstrates strong agent cohesion under dynamic conditions; therefore, the figure depicts algorithm effectiveness under the system's features. Achieved results prove that the system is capable of generalizing and performing reliably within bounded unpredictable environments. Leveraging attention-based neural communication combined with decentralized experience replay enables scalable behavior birth without the need for centralized control, which illustrates the potential of MANSS in practical applications such as autonomous exploration, environment monitoring, search and rescue, and other tasks.

Conclusion and Future Work

The research developed Multiagent Neural Swarm System (MANSS) which is a fully decentralized system employing a deep reinforcement learning framework and a neural communication protocol for real-time coordination in unstructured and dynamic environments. Through a system of simulacra experiments, MANSS outperformed previous work on fault tolerance, communication, and task performance in comparison to swarm and centralized coordination approaches. Agents do not require a central controller due to their decentralized architecture, neural policy learning, experience replay, and other features, which allows efficient coping with dynamic, uncertain, and uncontrolled conditions. These capabilities make MANSS ideal for real-world challenges like disaster response, autonomous exploration, and environmental monitoring.

Future work will focus on employing physical drone swarms in heterogeneous agent configurations and further extending the capabilities of MANSS. Other advancements could include specialized lightweight neural networks for edge devices and the application of task-hierarchical reinforcement learning for multi-level tasks. Furthermore, incorporating meta-learning and continual learning methodologies to agents may be investigated to create low and minimal retraining bounds for novel environments. In addition, integrating external sensor networks and cloud-based control may improve long-range situational awareness and coordination. The ultimate aim is to advance MANSS from a simulation-only environment into a sophisticated real-world deployment-ready system designed for complex operations.

References

Parker, L. E. (2008). Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents, 2*(1), 5-14.

Dorigo, M., & Birattari, M. (2007). Swarm intelligence. *Scholarpedia, 2*(9), 1462.

Zhang, K., Yang, Z., & Başar, T. (2019). Multiagent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control,* 321-384.

Foerster, J. N., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H., Kohli, P., & Whiteson, S. (2018). Stabilising experience replay for deep multiagent reinforcement learning. *In Proceedings of the 34th International Conference on Machine Learning* (pp. 1146–1155).

Nguyen, D. T., Nguyen, T., Nahavandi, S., & Tran, T. (2020). Deep reinforcement learning for multiagent systems: A review of challenges, solutions and applications. *IEEE Transactions on Cybernetics, 50*(9), 3826-3839.

Jiang, J., & Lu, Z. (2018). Learning attentional communication for multiagent cooperation. *Advances in Neural Information Processing Systems, 31.*

Sukhbaatar, S., Szlam, A., & Fergus, R. (2016). Learning multiagent communication with backpropagation. *In Advances in Neural Information Processing Systems* (pp. 2244–2252).

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., ... & Silver, D. (2019). Grandmaster level in StarCraft II using multiagent reinforcement learning. *Nature, 575*(7782), 350–354.

Rashid, T., Samvelyan, M., Schroeder de Witt, C., Farquhar, G., Foerster, J., & Whiteson, S. (2018). QMIX: Monotonic value function factorisation for deep multiagent reinforcement learning. *In International Conference on Machine Learning* (pp. 4295–4304).

Christianos, F., Schäfer, L., & Albrecht, S. V. (2020). Shared experience actor-critic for multiagent reinforcement learning. *In AAAI Conference on Artificial Intelligence* (Vol. 34, No. 05, pp. 7035–7042).

Parker, L. E. (2008). Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents, 2*(1), 5–14.

Aburasain, R. Y. (2025). Revolutionizing Traffic Flow Prediction Using a Hybrid Deep Learning Models with Kookaburra Optimization Algorithm. *Journal of Internet Services and Information Security, 15*(1), 486-501. https://doi.org/10.58346/JISIS.2025.I1.032

Dorigo, M., & Birattari, M. (2007). Swarm intelligence. *Scholarpedia, 2*(9), 1462.

Reddy, B. M. K., Abdul Azeez Khan, A., Javubar Sathick, K., & Arun Raj, L. (2025). Cyber Attack Recognition in an Internet of Things-Enabled Environment Using a Hybrid Optimised Deep Learning Approach. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 16*(1), 49-71. https://doi.org/10.58346/JOWUA.2025.I1.003

Zhang, K., Yang, Z., & Başar, T. (2019). Multiagent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control,* 321–384.

Hyun, K. S., Min, P. J., & Won, L. H. (2025). AI hardware accelerators: Architectures and implementation strategies. *Journal of Integrated VLSI, Embedded and Computing Technologies, 2*(1), 8–19. https://doi.org/10.31838/JIVCT/02.01.02

Nguyen, D. T., Nguyen, T., Nahavandi, S., & Tran, T. (2020). Deep reinforcement learning for multiagent systems: A review of challenges, solutions and applications. *IEEE Transactions on Cybernetics, 50*(9), 3826–3839.

Kavibharathi, S., Lakshmi Priyankaa, S., Kaviya, M. S., & Vasanthi, S. (2021). Live Chat Analysis Using Machine Learning. *International Academic Journal of Science and Engineering, 8*(1), 39–44. https://doi.org/10.9756/IAJSE/V8I1/IAJSE0805

Foerster, J. N., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H., Kohli, P., & Whiteson, S. (2018). Stabilising experience replay for deep multiagent reinforcement learning. *In Proceedings of the 34th International Conference on Machine Learning* (pp. 1146–1155).

Sukhbaatar, S., Szlam, A., & Fergus, R. (2016). Learning multiagent communication with backpropagation. *In Advances in Neural Information Processing Systems* (pp. 2244–2252).

Jiang, J., & Lu, Z. (2018). Learning attentional communication for multiagent cooperation. *Advances in Neural Information Processing Systems, 31.*

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., ... & Silver, D. (2019). Grandmaster level in StarCraft II using multiagent reinforcement learning. *Nature, 575*(7782), 350–354.

Rashid, T., Samvelyan, M., Schroeder de Witt, C., Farquhar, G., Foerster, J., & Whiteson, S. (2018). QMIX: Monotonic value function factorisation for deep multiagent reinforcement learning. *In International Conference on Machine Learning* (pp. 4295–4304).

Christianos, F., Schäfer, L., & Albrecht, S. V. (2020). Shared experience actor-critic for multiagent reinforcement learning. *In AAAI Conference on Artificial Intelligence* (Vol. 34, No. 05, pp. 7035–7042).

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multiagent actor-critic for mixed cooperative-competitive environments. *In Advances in Neural Information Processing Systems* (pp. 6379–6390).