LEARNING COMMUNICATION BETWEEN HETEROGENEOUS AGENTS IN MULTI-AGENT REINFORCEMENT LEARNING FOR AUTONOMOUS CYBER DEFENCE

ALEX POPA[1], ADRIAN TAYLOR[2], RANWA AL MALLAH[3]

[1]Department of Electrical and Computer Engineering, Royal Military College of Canada, Kingston Ontario, Canada
[2]Defence Research and Development Canada, Ottawa Ontario, Canada
[3]Department of Computer Engineering and Software Engineering, Polytechnique Montréal, Montréal Québec, Canada
E-MAIL: alex.popa@rmc-cmr.ca, adrian.taylor@drdc-rddc.gc.ca, ranwa.al-mallah@polymtl.ca

Abstract:

Reinforcement learning techniques are being explored as solutions to the threat of cyber attacks on enterprise networks. Recent research in the field of AI in cyber security has investigated the ability of homogeneous multi-agent reinforcement learning agents, capable of inter-agent communication, to respond to cyber attacks. This paper advances the study of learned communication in multi-agent systems by examining heterogeneous agent capabilities within a simulated network environment. To this end, we leverage CommFormer, a publicly available state-of-the-art communication algorithm, to train and evaluate agents within the Cyber Operations Research Gym (CybORG). Our results show that CommFormer agents with heterogeneous capabilities can outperform other algorithms deployed in the CybORG environment, by converging to an optimal policy up to four times faster while improving standard error by up 38%. The agents implemented in this project provide an additional avenue for exploration in the field of AI for cyber security, enabling further research involving realistic networks.

Keywords:

MARL; CybORG; CommFormer; AI for security; Cyber security; Inter-agent communication

1. Introduction

Modern computer and digital systems exist in a world of constant inter-connectivity where cyber threats pose a persistent risk to the daily lives of users [1]. The situation is further complicated by the rapid evolution of tactics, techniques and procedures employed by malicious actors operating in the cyber domain. State-affiliated actors from great and middle powers across the world are increasingly leveraging AI and advanced techniques to circumvent traditional defences [2].

To counter these evolving threats, researchers have looked toward Machine Learning (ML) as a means to secure the cyber domain. While Reinforcement Learning (RL) has proven effective for Network Intrusion Detection Systems (NIDS) in identifying malware and Distributed Denial of Service (DDoS) attacks, single-agent RL architectures have had limited success in responding to large-scale complex attacks [3][4]. Multi-agent Reinforcement Learning (MARL) offers a scalable alternative by distributing detection and response across the network. MARL agents can be trained to replace or support an existing Host-based Intrusion Detection System (HIDS), greatly extending the AI for security use case. Additionally, MARL agents can more closely emulate cyber security specialists, with varied skillsets, coordinating together to respond to cyber attacks. This ability to relate agents to real-life scenarios further improves the interpretability of agent actions. While these MARL algorithms provide a myriad of other benefits [5], they also suffer from additional instability [6], with inter-agent communication presented as a means to mitigate this instability [6].

Building upon the foundations established in [3], this work explores the application of MARL agents with heterogeneous capabilities, meaning varied observation and action spaces. By employing CommFormer, a state-of-the-art communication algorithm [6], this work shows that heterogeneous communication agents can outperform baselines established in the field [3].

2. Related Work

MARL has been described as a natural extension of single-agent RL, where agents engaging in a shared environment learn

to map states to actions through an iterative trial-and-error process [3][6]. MARL applications have been aimed at addressing scalability limitations associated with single-agent architectures. This increase in agent numbers generally leads to dynamic uncertainty and environment instability [6]. To mitigate this, researchers have used approaches like Centralized Training Decentralized Execution (CTDE), which allows agents to share global information during training while operating independently during execution. Communication further enhances coordination, particularly in partially observable environments where agents have limited observation spaces and must learn to share vital data. Despite the advantages associated with communication, drawbacks exist, such as increased bandwidth usage, slower inference time, and architectural complexity.

Significant research exists in the application of MARL across varied environments. However, research involving these algorithms in cyber security is limited to homogeneous agents [3][7]. In this context, MARL agents contend with partial observability, sparse rewards, and high stochasticity. Host-based agents are limited to local observations, requiring communication to coordinate or support a network-wide response [8]. Additionally, Autonomous Cyber Defence (ACD) requires multi-step responses to malicious actions, leading to delayed feedback for correct choices. Finally, the stochasticity of computer networks, where network traffic and benign user actions introduce significant noise, leads to a difficult learning environment. Despite these challenges, heterogeneous MARL agents can be tailored to niche roles, emulating cyber security experts, improving interpretability, helping minimize computational power, while limiting bandwidth usage.

The Reinforced Inter-Agent Learning (RIAL) algorithm, utilizing independent Q-networks, and the Differentiable Inter-Agent Learning (DIAL) [9] algorithm, enabling gradient sharing across agents, introduced deep Q-networks for communication. Complementary Attention for Multi-Agent reinforcement learning (CAMA) [10], also using Q-Learning, attempts to optimize observation spaces by dynamically altering observation spaces, a feature not currently supported by CybORG.

Communication Neural Network (CommNet) [11], using the REINFORCE training algorithm [12], builds a channel where agents transmit continuous vectors. CommNet's multiple communication cycles per timestep demand excessive bandwidth, making it ill-suited for applications in ACD. Individualized Controlled Continuous Communication Model (IC3Net) [13], also trained via the REINFORCE algorithm, improves efficiency by introducing a gating mechanism which allows agents to learn when to communicate. While useful for mixed cooperative-competitive settings, IC3Net lacks demonstrated support for heterogeneous agents.

Bidirectionally-Coordinated Network (BiCNet) [14], an early actor-critic example, requires the global state to be provided at each timestep, an unrealistic expectation in partially observable environments. Targeted Multi-Agent Communication (TarMAC) [15] utilizes an actor-critic architecture with a signature-based soft-attention mechanism, enabling agents to weigh message importance and target recipients. Similarly, Distributed Targeted Multi-Agent Communication (DTMAC) [16] adds historical message support. While highly effective, both algorithms were tested only on homogeneous agents.

Multi-Agent Incentive Communication (MAIC) [17] employs teammate modelling and Q-Learning to generate incentive messages that directly bias teammates' local Q-value tables. Although MAIC supports heterogeneous agents, this abstract communication lacks interpretability.

Advanced graph-based approaches include Multi-Agent Graph-attentIon Communication (MAGIC) [18], Information Maximizing Gated Spares Multi-Agent Communication (IMGS-MAC) [19] and CommFormer [6]. MAGIC employs Graph Attention Networks (GATs) via a Scheduler and Message Processor to manage multi-round communication. IMGS-MAC enforces sparsity to maximize information sharing within strict bandwidth constraints. Despite their architectural advantages, these graph models lack publicly available code and verified support for heterogeneous agents, ultimately leading to the use of CommFormer.

3. Methodology

We selected the CommFormer algorithm [6] for our use case as it was found to provide "performance levels comparable to approaches that permit unrestricted information sharing among agents" [6] while modelling agents in an intuitive way. CommFormer models multi-agent interactions as a Markov game defined by the tuple $\langle \mathcal{N}, \boldsymbol{\mathcal{O}}, \mathbf{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$ where $\mathcal{N}$ is the number of agents, $\boldsymbol{\mathcal{O}}$ is the joint observation space, $\boldsymbol{\mathcal{A}}$ is the joint action space, $\mathbf{A}$ maps $\boldsymbol{\mathcal{O}}$ and $\boldsymbol{\mathcal{A}}$ to the reward range, $mathcalP$ defines the probability function, and γ denotes the discount factor. Inter-agent communication is framed as a learnable directed graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, optimizing both the graph structure and architectural parameters concurrently via a bi-level optimization process. In this graph, nodes represent agents ($\mathcal{N}$) and edges ($\mathcal{E}$) denote unidirectional communication channels. A sparsity parameter, $\mathcal{S}$, strictly governs the number of active edges $\mathcal{S} \times N^2$, ensuring bandwidth efficiency.

To train its agents, CommFormer employs an encoder-decoder transformer architecture which leverages a self-

attention mechanism to calculate scores via Query, Key, and Value vectors, helping capture complex dependencies between agents and observations. Crucially, the learned communication graph masks information from other agents, restricting information flow strictly to authorized channels. Finally, the decoder generates joint action sequences auto-regressively, feeding each generated action back into the decoder to inform the actions of other agents operating in the environment [6].

This work utilizes CyMARL, an extension of the CybORG [20] environment, built upon PyMARL2 [21], to train MARL algorithms for "tactical decision-making in cyber defence" [7]. Our baseline scenario, features a six-host, two-subnet network where a rigidly programmed red (attacker) agent methodically seeks to exploit vulnerabilities and escalate privileges with the end goal of disrupting a critical server (OpServer).

Defending the network are two blue (defender) agents, acting as subnet Intrusion Prevention Systems (IPS), which must learn to mitigate red incursions through actions like Monitor, Remove, Analyse, Restore, and Block. The training process is guided by a team-based reward function that heavily penalizes the blue agents for allowing compromises to network assets as well as for unnecessarily disruptive defensive actions. To simulate realistic, partially observable conditions, the blue agents receive localized, binary-formatted observations of host and subnet statuses, while automatic monitoring actions only detect exploits with a 50% success rate, enabling red agents to establish covert, privileged footholds on network hosts. Blue agents are forced to learn indicators of compromise associated with undetected exploits as a result.

CommFormer is implemented using the Python language, its code features a highly modular structure that can be easily adjusted to interact with any environment capable of processing observation, reward and action sequences. The integration of the CommFormer algorithm into CyMARL required the creation of two wrappers. The wrappers developed for this work use the CommFormer Predator-Prey/Predator-Capture-Prey environment scripts created by Hu et al. [6] as a base to build from. The higher level wrapper is focused on parsing command-line variables, such as hyperparemeters and system information (i.e. CUDA GPU availability and multi-processing settings), helping set up training the environments and saving training data as needed. The underlying wrapper we developped, focuses on the interacting with CyMARL. Its pseudocode can be seen in Algorithm 1 below.

Additional changes to both the CyMARL and CommFormer codebases include the addition of support for agents with different observation and action spaces, the ability to log agent actions during training and evaluation, the creation of CybORG environment interaction workers used in multi-processing, as well as numerous tweaks to variables exchanged between CyMARL and CommFormer. The CyMARL-CommFormer repository, including details regarding code changes, is available publicly on GitHub [22].

Algorithm 1 CybORG Runner

Input: Arguments and CommFormer configuration data supplied by Train CyMARL CommFormer.
Initialize: CybORG runner class using the CommFormer runner class parent.
Calculate number of episodes using environment steps and number of multi-processing threads.
for episode in episodes **do**
  Reset environment and Replay Buffer $\mathcal{B}$.
  Log communication channel matrix for the episode.
  for step in training episode length **do**
    Sample actions from CommFormer.
    Observe reward, next observation and available actions from CyMARL.
    Insert data into buffer $\mathcal{B}$.
  end for
  Compute return and update the CommFormer network.
  Log information, actions and save model according to pre-defined logging interval.
  Run evaluation episodes according to pre-defined evaluation interval.
end for

4. Evaluation

Following the integration of CommFormer into CyMARL, the algorithm was evaluated across three progressively complex scenarios to validate its performance and scalability against established baselines.

Homogeneous Baseline. To ensure a fair comparison against prior DIAL implementations in CyMARL [3], this initial scenario maintained an identical network topology (six hosts across two subnets), red agent attack path, and uniform blue agent action/observation spaces. Previously optimized hyperparameters outlined in [3] and [6] were used to establish our baseline, with no hyperparameter tuning conducted.

Heterogeneous Agent Scenario. The scenario was then adjusted by introducing a specialized firewall agent (BlueFW). This setup established heterogeneous agents with distinct action and observation spaces. The subnet agents (Blue0 and Blue1) were tasked with monitoring hosts and executing tar-

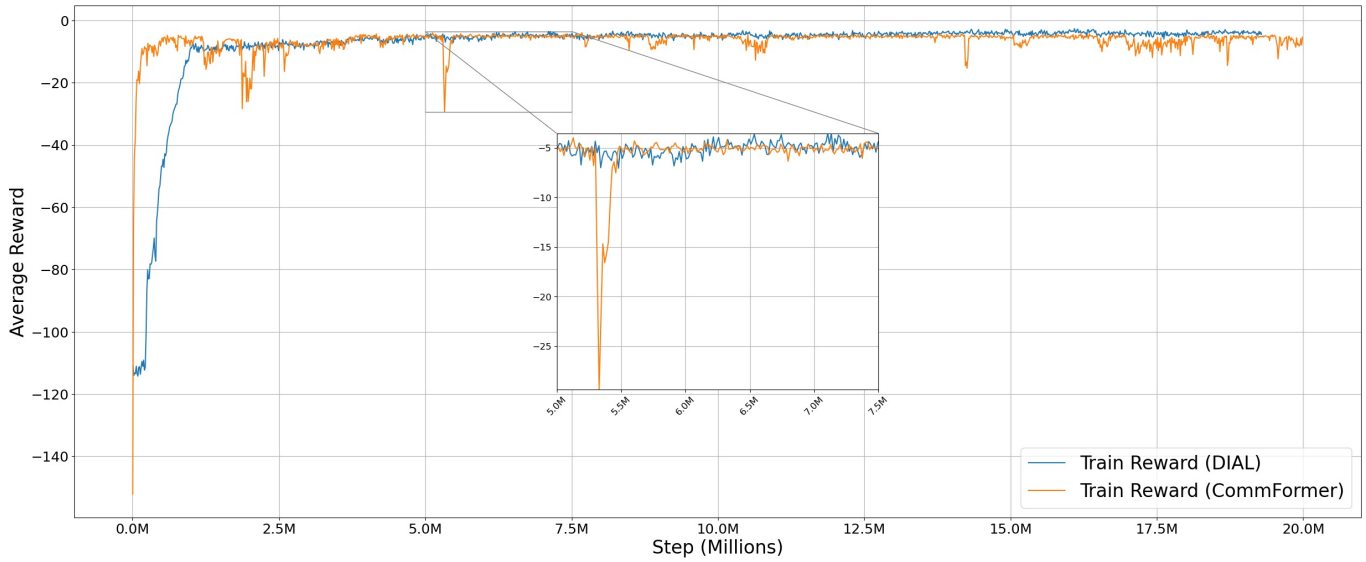

FIGURE 1. Homogeneous agent scenario training curve. The inset plot shows best case scenario performance of approximately -4 average reward.

geted mitigating actions, such as Remove, Restore, and Analyse, while the BlueFW agent was strictly limited to executing the Block action. Zero-padding was introduced to normalize the observation vector sizes across these varied agents.

Host-Based Scenario. To test CommFormer's scalability, the network was expanded to include individual blue agents for each host, simulating localized Endpoint Detection and Response (EDR) applications operating alongside the central BlueFW agent. Because this restricted each agent's observation space to a single host, training instability emerged. Consequently, the reward function was adjusted to increase penalties for unnecessary Analyse actions while discounting correctly coordinated Block actions. Unlike scenario one and two, hyperparameter tuning (e.g., entropy coefficient, PPO clip) was also conducted to prevent convergence on sub-optimal policies.

5. Results

Homogeneous Baseline. CommFormer was evaluated throughout a 20 million timestep training session using the homogeneous baseline scenario discussed in the previous section. It demonstrated superior learning efficiency, converging to a semi-optimal policy nearly four times faster than the DIAL baseline. CommFormer achieved more stable mean rewards, defined here as lower standard deviation, during training -6.54 ± 5.11 compared to DIAL -7.9 ± 14.39. Communication was critical in the agent's success, as evidenced by an agent detecting a network scan successfully and relaying this data, enabling a peer to execute targeted remediations on a compromised pivot host. Figure 1 illustrates the mean reward assigned to the CommFormer and DIAL agents as they were trained.

Heterogeneous Agent Scenario. CommFormer was subsequently trained and evaluated over 5 million timesteps using the adjusted heterogeneous agent scenario which introduced a specialized firewall agent (BlueFW) alongside subnet agents. The model experienced a minor performance drop (training mean reward of -9.18 ± 8.89) but maintained rapid convergence. Communication across agents was once again demonstrated when a subnet agent successfully detected a vulnerability scan and alerted the BlueFW agent, whose observation space is limited to the blocked/unblocked status of subnets, prompting immediate network-level mitigation via a Block subnet action. Figure 2 illustrates the mean reward assigned to the CommFormer heterogeneous agents during training.

Host-Based Scenario. The final scenario simulated localized EDR agents deployed per host, strictly limiting local observations. The severe restrictions placed on CommFormer underscored the necessity for inter-agent communication. While extreme sparsity occasionally delayed initial detections, active channels ultimately routed critical indicators of compromise. Evaluation logs confirmed that isolated host agents successfully initiated Analyse and Restore actions based entirely on observations communicated by peer agents, demonstrating CommFormer's efficacy in decentralized, highly restricted environments. Figure 3 illustrates the training and evaluation curves for this scenario.

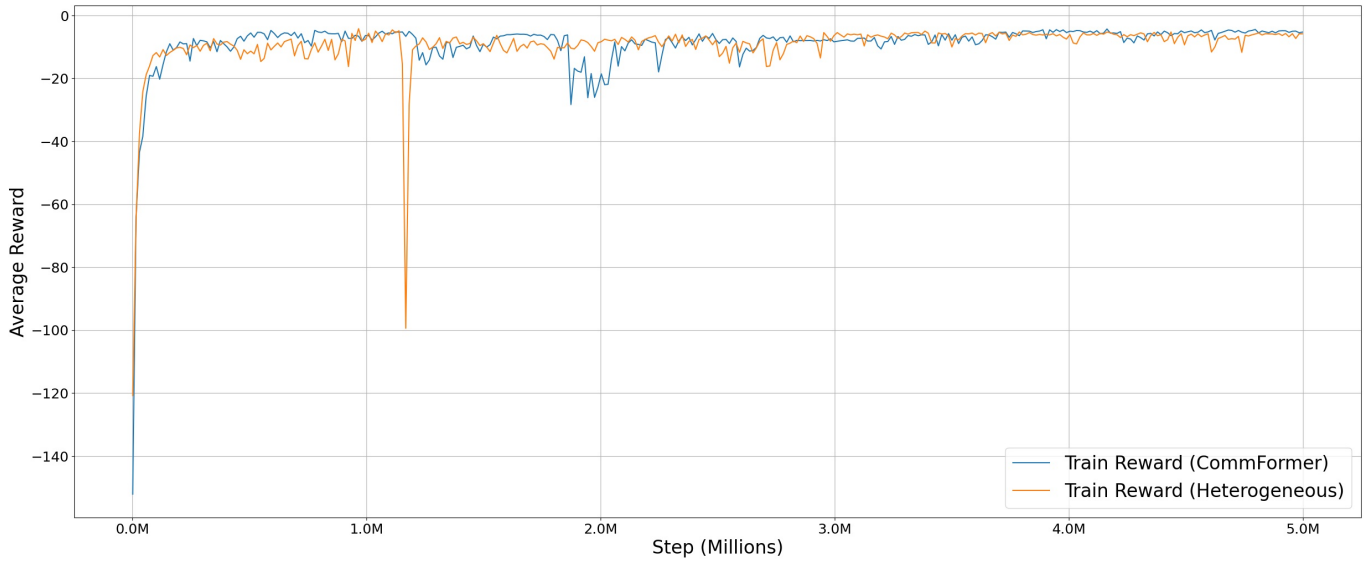

FIGURE 2. Heterogeneous agent scenario training curve.

6. Conclusion

This work successfully adapted CommFormer and integrated it into CyMARL to evaluate heterogeneous MARL for ACD. Compared to established DIAL baselines, CommFormer achieved comparable overall performance but converged to near-optimal policies faster. When scaled to realistic heterogeneous host-based scenarios, the agents effectively coordinated actions across varied observation and action spaces. Although this scalability demonstrated peak performance exceeding baselines, the model exhibited late-stage training instability, necessitating specific hyperparameter and reward tuning. MARL unlocks the ability to explore scenarios beyond those that RL agents can achieve, at the cost of training complexity. Ultimately, this research validates the applicability of heterogeneous communicating agents in ACD, establishing a robust foundation for future complex network defence simulations.

References

[1] Government of Canada, "National Cyber Threat Assessment 2025-2026", Canadian Centre for Cyber Security, Oct. 30, 2024. https://www.cyber.gc.ca/en/guidance/national-cyber-threat-assessment-2025-2026

[2] "2025 Global Threat Report," Tech. Rep., CrowdStrike, 2025.

[3] F. Contractor and R. Al-Mallah, "Learning to communicate in multi-agent reinforcement learning for autonomous cyber defence," Master's thesis, Royal Military College of Canada, 2024.

[4] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," IEEE Transactions on Neural Networks and Learning Systems, vol. 34, p. 3779–3795, Aug. 2023.

[5] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. P´erolat, D. Silver, and T. Graepel, "A unified game-theoretic approach to multi-agent reinforcement learning," Advances in neural information processing systems, vol. 30, 2017.

[6] S. Hu, L. Shen, Y. Zhang, and D. Tao, "Learning multi-agent communication from graph modeling perspective," arXiv preprint arXiv:2405.08550, 2024.

[7] J. Wiebe, R. A. Mallah, and L. Li, "Learning cyber defence tactics from scratch with multi-agent reinforcement learning," arXiv preprint arXiv:2310.05939, 2023.

[8] S. Sukhbaatar, R. Fergus, et al., "Learning multiagent communication with backpropagation," Advances in neural information processing systems, vol. 29, 2016.

[9] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent rein-

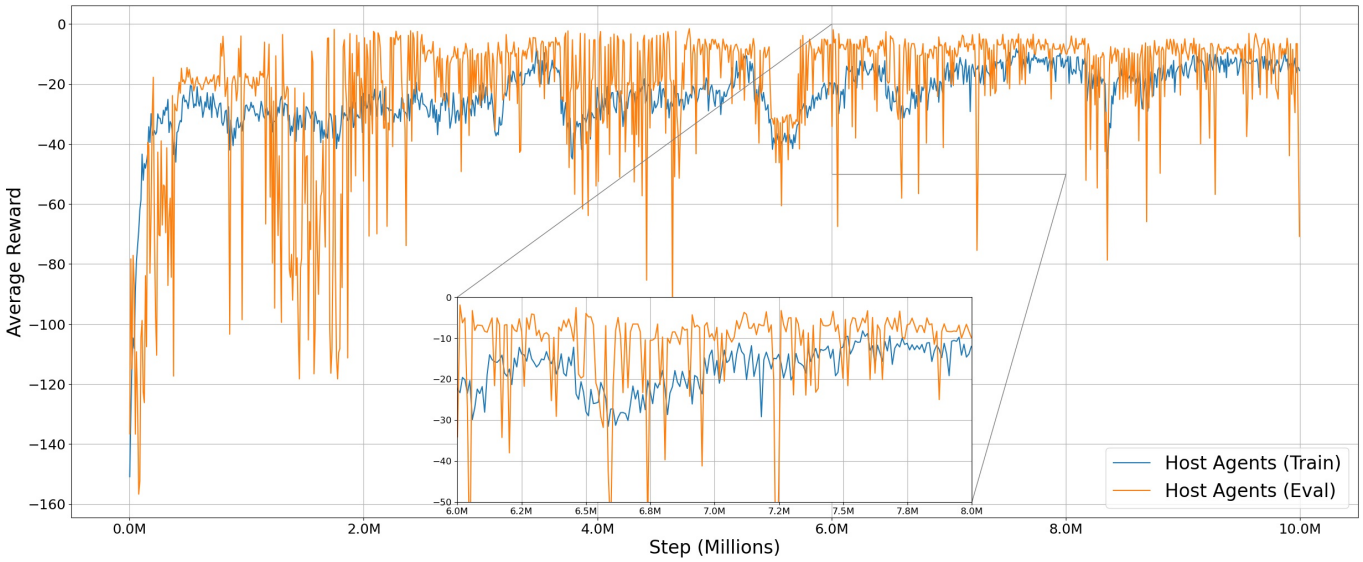

FIGURE 3. Homogeneous agent scenario training curve. The inset plot provides a closer view of peak performance across training and evaluation episodes.

forcement learning," Advances in neural information processing systems, vol. 29, 2016.

[10] J. Shao, H. Zhang, Y. Qu, C. Liu, S. He, Y. Jiang, and X. Ji, "Complementary attention for multi-agent reinforcement learning," in International Conference on Machine Learning, pp. 30776–30793, PMLR, 2023.

[11] S. Sukhbaatar, R. Fergus, et al., "Learning multiagent communication with backpropagation," Advances in neural information processing systems, vol. 29, 2016.

[12] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," Machine learning, vol. 8, no. 3, pp. 229–256, 1992.

[13] A. Singh, T. Jain, and S. Sukhbaatar, "Learning when to communicate at scale in multiagent cooperative and competitive tasks," arXiv preprint arXiv:1812.09755, 2018.

[14] P. Peng, Y. Wen, Y. Yang, Q. Yuan, Z. Tang, H. Long, and J. Wang, "Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games," arXiv preprint arXiv:1703.10069, 2017.

[15] A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau, "Tarmac: Targeted multi-agent communication," in International Conference on machine learning, pp. 1538–1546, PMLR, 2019.

[16] C. Xu, H. Zhang, and Y. Zhang, "Multi-agent reinforcement learning with distributed targeted multi-agent communication," in 2023 35th Chinese Control and Decision Conference (CCDC), pp. 2915–2920, IEEE, 2023.

[17] L. Yuan, J. Wang, F. Zhang, C. Wang, Z. Zhang, Y. Yu, and C. Zhang, "Multi-agent incentive communication via decentralized teammate modeling," in Proceedings of the AAAI conference on artificial intelligence, vol. 36, pp. 9466–9474, 2022.

[18] Y. Niu, R. R. Paleja, and M. C. Gombolay, "Multi-agent graph-attention communication and teaming.," in AAMAS, vol. 21, p. 20th, 2021.

[19] S. Karten, M. Tucker, S. Kailas, and K. Sycara, "Towards true lossless sparse communication in multi-agent systems," arXiv preprint arXiv:2212.00115, 2022

[20] M. Standen, M. Lucas, D. Bowman, T. J. Richer, J. Kim, and D. Marriott, "Cyborg: A gym for the development of autonomous cyber agents," arXiv preprint arXiv:2108.09118, 2021.

[21] PyMARL 2, 2021, accessed: 2025-07-08. [Online]. Available: https://github.com/hijkzzz/pymarl2

[22] CyMARL-CommFormer, 2025, accessed: 2025-12-03. [Online]. Available: https://github.com/Poly-AIvsAI/CyMARL-CommFormer.