Leszek Rutkowski · Rafał Scherer ·
Marcin Korytkowski · Witold Pedrycz ·
Ryszard Tadeusiewicz · Jacek M. Zurada (Eds.)

LNAI 15166

Artificial Intelligence and Soft Computing

23rd International Conference, ICAISC 2024
Zakopane, Poland, June 16–20, 2024
Proceedings, Part III

Springer

Lecture Notes in Computer Science

Lecture Notes in Artificial Intelligence 15166

Founding Editor

Jörg Siekmann

Series Editors

Memory Augmented Multi-agent Reinforcement Learning for Cooperative Environment

Maryam Kia(✉), Jordan Cramer, and Artur Luczak

University of Lethbridge, Lethbridge, AB T1K 3M4, Canada
{m.kiakojouri,jordan.cramer,luczak}@uleth.ca

Abstract. Multi-Agent Reinforcement Learning (MARL) offers a framework for collaborative decision-making among multiple agents with diverse objectives and observations through interactions with their environments. In this study, we introduce MA-LSTMTD3, a variant of the Twin Delayed Deep Deterministic (TD3) method incorporating long-short-term memory (LSTM) to enhance MARL performance. Our approach aims to address challenges related to coordination and partial observability in complex environments. Through comprehensive experimentation, we demonstrate the efficacy of our memory augmented algorithm across both Markov Decision Process (MDP) and Partially Observable Markov Decision Process (POMDP) settings, including scenarios with varying agent counts. Our findings underscore the potential of memory-based approaches in advancing MARL algorithms for real-world applications, particularly in environments with a higher number of collaborating agents.

Keywords: Reinforcement Learning · Multi Agent Systems · LSTM

1 Introduction

The computational field of Reinforcement Learning (RL) offers a framework within which decision making is aimed at maximizing the reward and minimizing penalties, through iterative learning [1]. An RL problem can be formulated in various ways depending on the assumptions made about the environment, and factors such as task definitions, the number of agents involved, and the initial knowledge available to these agents [2]. Primarily, single agent RL is formalized within the framework of Markov Decision Problems (MDP), where a solitary agent interacts with an environment characterized by probabilistic transitions and rewards. However, in the pursuit of addressing real-world challenges where multiple decision makers interact at the same time, such as those encountered in robotics, autonomous systems, or traffic management, the focus has shifted towards Multi-Agent RL (MARL). In MARL, where multiple agents coexist, interact, and pursue shared or conflicting objectives, the dynamic nature of the environment intricately intertwines with the decision-making process [3]. This necessitates innovations in advancing coordination mechanisms, optimizing competitive strategies, and enhancing communication protocols among agents to enable effective collaboration or competition while achieving desired outcomes.

L. Rutkowski et al. (Eds.): ICAISC 2024, LNAI 15166, pp. 92–103, 2025.
https://doi.org/10.1007/978-3-031-81596-6_8

The integration of deep neural networks in RL, referred to as deep reinforcement learning (DRL), has led to emergence of learning techniques that utilizes neural networks as function approximators. This empowers agents to process high-dimensional input data and acquire hierarchical representations that capture intricate features of the environment [4]. Given the capacity of deep neural networks to manage large state and action spaces, combinations of DRL methods with multi-agent scenarios have shown promise in addressing the complexities of multi-agent environments including non-stationarity, partial observability, continuous spaces, training schemes, and transfer learning [5, 6]. However, challenges persist, particularly in effectively capturing the temporal dependencies and long-term dependencies inherent in sequential decision-making processes. Traditional DRL approaches often struggle to adequately model such dependencies, leading to suboptimal performance in tasks where past actions significantly influence future outcomes.

Previous research has explored the integration of memory units within DRL algorithms, enabling agents' decisions to depend not only on the current state but also on the history of observed states and actions. Memory mechanisms in MARL are commonly implemented using Recurrent Neural Networks (RNNs) in deep learning methods, with adaptation to various environmental settings. For example, [7] investigated the utility of memory in environments with diverse properties, demonstrating its efficacy in scenarios where learning agents must model other agents. Similarly, authors in [8] showcased the effectiveness of memory-driven communication in enhancing coordination in MARL. They proposed a framework for multi-agent training using deep deterministic policy gradients, facilitating concurrent learning of an explicit communication protocol through a memory device. This approach resulted in improved performance compared to baseline algorithms in small-scale systems. Memory units have also proven effective in solving Partially Observable Markov Decision Processes (POMDPs), where observations provide only partial information about the underlying state. [9] employed Long Short-Term Memory (LSTM) units in a policy-gradient method, achieving promising results in POMDP car driving tasks. Additionally, [10] utilized a temporal difference approach to update the action-value function, jointly training convolutional and LSTM layers. This enabled successful learning directly from pixels, effectively addressing the POMDP version of Atari games. These developments underscore the growing importance of memory mechanisms in advancing MARL algorithms to tackle challenges related to coordination, communication, learning, and the inherent uncertainty of POMDPs in multi-agent environments.

In this paper, we introduce an approach to enhancing MARL by integrating Long Short-Term Memory (LSTM) units into actor-critic architecture. We present comprehensive experimental results across both Markov Decision Process (MDP) and Partially Observable Markov Decision Process (POMDP) versions of a cooperative environment, as well as scenarios with varying agent counts. Additionally, we Provide a comparison of the developed algorithm's performance with two other baseline methods. Through experimentation and analysis, we demonstrate the efficacy of our memory-based algorithm in facilitating coordination and handling partial observability in complex environments. This approach aids in inferring the underlying state, especially advantageous

when dealing with missing information, thus highlighting its potential for real-world applications.

The paper is structured as follows: In the background section, we delve into the theoretical foundations of multi-agent reinforcement learning (MARL) and elaborate on the components of the proposed approach. The methods section provides a detailed explanation of our proposed approach, outlining the integration of LSTM into the actor-critic architecture and the algorithmic framework employed for training and evaluation. In the experimental settings and results section, we present our experimental setup, including the cooperative environment with various settings, and report the outcomes of our experiments. Finally, in the conclusions and discussion section, we summarize our findings, discuss their implications, and provide insights into future research directions, emphasizing the significance of memory-based approaches in advancing MARL algorithms for real-world applications.

2 Background

2.1 Decision Process

Markov Decision Process (MDP). A Markov Decision Process (MDP) is a sequential decision process for a fully observable, stochastic environment with a Markovian transition model and additive rewards [11]. Formally, MDP can be defined as a 4-tuple (S, A, P, R), where S is the state space, AA is the action space, PP is the transition probability and R is the reward function. At each discrete time t, an agent selects an action $a_t \in A$ in state $s_t \in S$, transitions to the next state s_{t+1} with probability $\mathrm{P}(\mathrm{s_{t+1}} | \mathrm{s_t}, \mathrm{a_t})$, and receives the immediate reward $\mathrm{R}(\mathrm{s_t}, \mathrm{a_t}, \mathrm{s_{t+1}})$.

Partial Observable Markov Decision Process (POMDP). Partially Observable Markov Decision Process (POMDP) is a generalization of a MDP but does not assume that the state is fully observable and is defined as a 6-tuple (S, A, P, R, O, Ω), where S, A, P, and R are the same as that in MDP, with an additional observation space O and observation model Ω. Although the underlying state transition in a POMDP is the same as those in an MDP, the agent cannot observe the underlying state, instead it receives an observation $o_{t+1} \in O$ when reaching the next state s_{t+1} with the probability $\Omega(\mathrm{o_{t+1}} | \mathrm{s_{t+1}})\Omega(\mathrm{o_{t+1}} | \mathrm{s_{t+1}})$. For POMDPs, since the state s is not observable, the observation o is used in learning value functions or policy.

Markov Games. Multi-agent extension of Markov Decision Processes is called partially observable Markov Games [12]. In Markov Games, multiple agents interact with the environment and with each other, and their actions impact the reward structure. The state transitions and rewards depend not only on the agent's actions but also on the actions of other agents. Markov Games incorporate the influence of other agents' actions on the environment and rewards, making them more suitable for scenarios involving cooperation, competition, or coordination among multiple decision-makers.

Actor Critic Methods. Actor-critic methods represent a prominent approach in RL, bridging the gap between policy-based and value-based methods. While policy based methods directly parameterize the policy, which is the agent's strategy for selecting

actions, and the value based methods estimate the value of states or state-action pairs, representing the expected cumulative reward the agent can achieve from that state onward, in the actor-critic framework the actor represents a policy network that outputs a probability distribution over actions given a state, and the critic is a value network that estimates the expected return for each state or state-action pair. During training, the actor and the critic are trained jointly by using the critic's output as a baseline or a target for the actor's gradient update. This way, the actor can learn from both its own experience and the critic's feedback.

Deep Deterministic Policy Gradient. The Deep Deterministic Policy Gradient (DDPG) is a variant of actor-critic algorithms designed for solving continuous action space problems, where the objective is to learn a deterministic policy that maximizes the expected cumulative long-term reward [13]. As an off-policy algorithm, DDPG uses a replay buffer to store past experiences from the exploration phase. DDPG also uses target networks for both the actor and the critic, which are periodically updated with the parameters of the respective online networks providing smoother and more stabilized estimation during the training.

Twin Delayed Deep Deterministic Policy Gradient. The Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm is a variant of the DDPG algorithm that addresses the issue of overestimation of value functions in actor-critic methods, enhancing stability and sample efficiency [14]. TD3 has demonstrated remarkable performance in continuous control tasks, specifically in challenging high-dimensional environments. TD3 algorithm employs two critic networks, each independently estimating Q-values for the same state-action pair and uses the minimum as the target Q-value during the update process, leading to more accurate value estimations. Additionally, TD3 introduces target policy smoothing, clipped double Q-learning, and delayed policy updates contributing to its better performance compared to DDPG.

Long Short Term Memory. LSTM, a type of Recurrent Neural Network (RNN), was introduced to overcome challenges associated with capturing long-term dependencies in sequential data [15]. Its unique architecture incorporates a memory cell, input gate, forget gate, and output gate, effectively addressing the vanishing gradient problem, and enabling the network to sustain gradients across extended sequences. LSTMs have found broad applications in natural language processing, speech recognition, and time-series analysis, gaining recognition for their prowess in modeling intricate sequential dependencies.

3 Methods

3.1 Multi Agent LSTMTD3

In this study, we present an extension of memory-based actor-critic networks [16] tailored for the multi-agent setting, leveraging the MADDPG algorithm outlined in [17]. Our approach, illustrated in Fig. 1, adopts a framework of centralized training and decentralized execution. During the training phase, the critic network Q, receives additional

information regarding the policies of other agents, while the learned policies at execution time by actor network π, are restricted to accessing only local information. This differential access to information ensures that during training, the policy gains a comprehensive understanding of the environment and potential strategies. Our proposed method integrates a memory component wherein both the actor and critic are enhanced with an LSTM memory unit. This augmentation facilitates the extraction of relevant information from past history, which is then combined with the extracted features from the current observations.

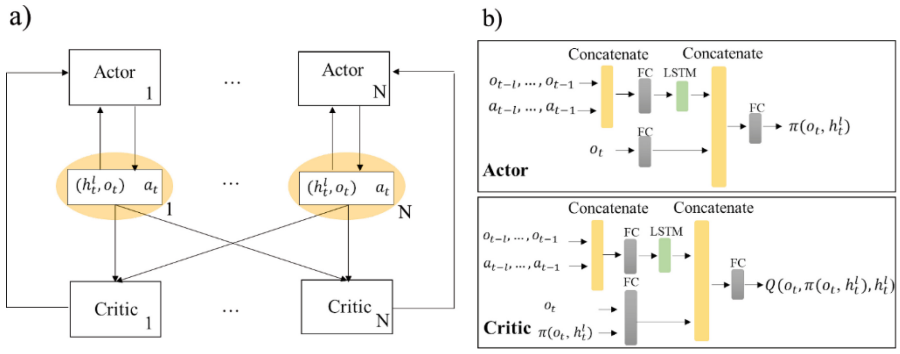

Fig. 1. Overview of the proposed method with decentralized actor centralized critic framework. a) During the training, the critic has access to the extra information from other agents while during execution each agent only has access to their own information. b) Both actor and critic incorporate and LSTM-based memory component in the architecture

In accordance with an off-policy learning setting, the experience replay mechanism is utilized to store the experienced trajectories, encompassing observations, actions, rewards, next observations and the done flag at time t. Additionally, this replay buffer is extended to include a history of paired observations and actions up to time t, with a predefined length of l as $h_t = o_{t-l}, a_{t-l}, \ldots, o_{t-1}, a_{t-1}$. During the training phase, a mini batch of S examples $\{h_t^l, o_t, a_t, r_t, o_{t+1}, d_t\}i = 1S\{h_t^l, o_t, a_t, r_t, o_{t+1}, d_t\}$ is uniformly sampled from replay buffer D, and the optimization follows that of TD3 algorithm. The pseudo code for the MA-LSTMTD3 algorithm is provided in Algorithm 1. Our implementation code is available at https://github.com/niktaaan/Multi-Agent-LSTMTD3.

Algorithm1: Pseudo-code for MA-LSTMTD3 for *N* Agents

Input: LSTM unit state-action history length L
for agent i=1 to N do
  Initialize critics $Q_i^{\theta_1}$, $Q_i^{\theta_2}$, and actor $Q_i^{\theta_1}$
  Initialize target networks $Q_i^{\acute{\theta}_1} \leftarrow Q_i^{\acute{\theta}_1}$, $Q_i^{\acute{\theta}_2} \leftarrow Q_i^{\theta_2}$, $\pi_i^{\acute{\mu}} \leftarrow \pi_i^{\mu}$
  Initialize state-action history buffer $H_i \leftarrow 0$
end for
Initialize replay buffer D
Initialize environment and obtain initial environment state $o_1 = env.reset()$
for $t = 1$ to T do
  for each agent i, select action i
  Agents take actions and observe next state, reward and done flag:
  $(o_{t+1}, r_t, d_t) = \text{env.step}(a_t)$
  Store experience tuple $(o_t, a_t, r_t, o_{t+1}, d_t)$ in D
  if d then:
    Reset environment $o_{t+1} = env.reset()$ and history buffers $H_{1\ldots N} \leftarrow 0$
  else
    for each agent i, update history h_{t+1}^l from H_i:
    $h_{t+1}^l = (h_t^l - (o_{t-1}, a_{t-1}^i)) \cup (o_t, a_t^i)$
  end if
  for agent $i = 1$ to N do
    Sample a mini-batch of S experiences, with agent i's corresponding history h_t^l from H_i :
    $\{h_t^l, o_t, a_t, r_t, o_{t+1}, d_t\}_{i=1}^{S}$ from D
    Optimize critics $Q_i^{\theta_1}$, $Q_i^{\theta_2}$
    if t mod $delay = 0$ then
      optimize actor π_i^{μ}
      update target networks:
      $\pi_i^{\acute{\mu}} \leftarrow \tau\pi_i^{\mu} + (1-\tau)\pi_i^{\acute{\mu}}$
      $Q_i^{\acute{\theta}_1} \leftarrow \tau Q_i^{\theta_1} + (1-\tau)\, Q_i^{\acute{\theta}_1}$, $Q_i^{\acute{\theta}_2} \leftarrow \tau Q_i^{\theta_2} + (1-\tau)\, Q_i^{\acute{\theta}_2}$
    end if
  end for
end for

The baselines utilized for comparing the proposed MA-LSTMTD3 are two deterministic algorithms: MA-DDPG and MA-TD3, where policies were evaluated every 1000 episodes, by playing 100 episodes, and averaging the resulting scores. In all experiments, hyperparameters were determined through a random search process to ensure robustness and generalizability. The following hyperparameters were consistent across all experiments: training proceeded for 300,000 steps, with the training phase initiated after an initial 25,000 steps to allow for stabilization. A replay buffer size of 600,000 was utilized to store past experiences for efficient learning. Each training iteration involved a batch size of 1024 samples drawn from the replay buffer to update the neural network parameters. The discount factor was set to 0.99 and tau as the soft update rate was set to 0.001. Also, the Adam optimizer was employed for optimization in every training step, utilizing a learning rate of 0.0001. Delay interval of 2 was used for MA-TD3 and MA-LSTMDTD3, and history length was set to 5 for MA-LSTMTD3 algorithm. Other

hyperparameters, including the details for network architecture in actor and critic are provided in Table 1. While DDPG and TD3 utilize fully connected (FC) layers exclusively, the architecture of LSTMTD3 incorporates a memory extraction block composed of both FC and LSTM layers, along with separate blocks for current feature extraction and post-combination. These components are indicated in three rows of layer size specifications in the table, representing the respective parts of the network architecture.

Table 1. Network parameters.

Layer size	Computation Block	MA-DDPG	MA-TD3	MA-LSTMTD3
Actor Network	Memory Extraction [FC], [LSTM]	–	–	[128], [128]
	Current Feature [FC]	[256, 256]	[256, 256]	[128]
	Post Combination [FC]	–	–	[128]
Critic Network	Memory Extraction [FC], [LSTM]	–	–	[128], [128]
	Current Feature [FC]	[256, 256]	[256, 256]	[128]
	Post Combination [FC]	–	–	[128]

4 Experimental Settings and Results

4.1 Experiments

To assess the efficacy of the proposed MA-LSTMTD3 algorithm, we employed Multi-agent Particle Environments (MPE) from the Petting Zoo library [18]. MPE offers a communication-oriented environment where particle agents interact with fixed landmarks, capable of movement and communication. Our experiments focused on the simple spread environment, a cooperative navigation setting comprising N agents and N landmarks, illustrated in Fig. 2, where the agents' task is to cover all landmarks while avoiding collisions. Each agent receives a global reward based on its proximity to the nearest landmark and incurs penalties for collisions with other agents. Our evaluation encompassed both MDP and POMDP conditions, with a varying number of agents to account for a range of environmental complexities.

POMDP Versions of the Environment. In the aforementioned environment, we introduced modifications to the observation space, resulting in partial observability versions of the tasks. Below are detailed explanation of the two versions implemented:

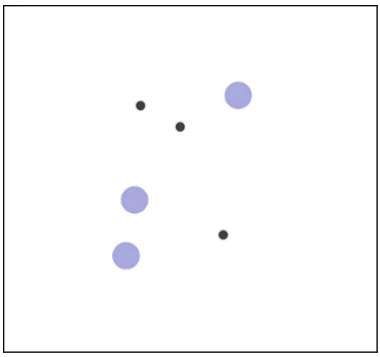

Fig. 2. Illustration of the simple_spread_v3 task from MPE, featuring cooperative agents depicted in blue and static landmarks depicted in black. Agents interact with the environment, while landmarks remain fixed and cannot be controlled.

Random Sensor Missing. In this version, we introduced partial observability by randomly setting the values of a specified number of sensors in the observation space to zero. This approach simulates scenarios where agents experience sensor failures or have access to only a subset of the environment's sensory inputs.

Random Noise. In this variant, we introduced random noise into the observations simulating the inherent uncertainty and unpredictability often present in real-world sensor reading. We conducted experiments in two distinct scenarios each characterized by varying levels of noise standard deviation. In the first scenario, we incorporated a relatively low noise standard deviation (std $= 0.1$), representing minor disturbances or sensor inaccuracies. In contrast, the second scenario featured a higher noise standard deviation (std $= 0.2$), simulating more significant disruptions or uncertainties in the sensor data.

4.2 Results

We evaluated the performance of the MA-LSTMTD3 algorithm in comparison to MA-DDPG and MA-TD3 across a range of multi-agent settings. This analysis encompassed fully observable scenarios, as well as various partial observable environments. Results are reported on different environment sizes with an increase in the number of agents, presenting an overview of performance ranging from simpler to more complex versions.

The mean rewards at the final training step averaged across four different seeds, $\pm$ the standard error of the mean (SEM), are presented in Table 2 and Table 3 for MDP and POMDP versions of the environment, respectively. Across all experiments, MA-LSTMTD3 outperforms MA-TD3 and MA-DDPG for the number of agents of five and higher.

Table 2. Mean performance ± SEM for MA-LSTMTD3, MA-DDPG, and MA-TD3 on the original MDP version of the simple_spread_v3 environment with varying numbers of agents. Numbers in bold denote cases where our model performed the best, which were usually for most challenging cases with most agents.

# of Agents	MA-DDPG	MA-TD3	MA-LSTMTD3
1	-2.9 ± 0.1	-2.7 ± 0.1	-3.7 ± 0.1
3	-74.9 ± 7	-47 ± 2	-58.4 ± 1.2
5	-279 ± 6.3	-151.1 ± 1.6	$\mathbf{-149.8 \pm 0.5}$
8	-863 ± 116.2	-357.5 ± 2.9	$\mathbf{-347.1 \pm 6.6}$
10	-990 ± 27.8	-555.2 ± 6.8	$\mathbf{-537.2 \pm 6.5}$

Table 3. Mean performance ± SEM for MA-LSTMTD3, MA-DDPG, and MA-TD3 on the POMDP versions of the simple_spread_v3 environment with varying numbers of agents. Numbers in bold denote cases where our model performed the best, which were usually for most challenging cases with most agents.

POMDP version	# of Agents	MA-DDPG	MA-TD3	MA-LSTMTD3
Random Sensor Missing (N = 2)	1	-2.7 ± 0.1	-2.6 ± 0.1	-3.6 ± 0.1
	3	-72.1 ± 2.2	-47.6 ± 2.1	-57 ± 0.8
	5	-292 ± 13.5	-152.6 ± 1.3	$\mathbf{-149.1 \pm 1.4}$
	8	-717 ± 75.7	-369.5 ± 8.8	$\mathbf{-340.1 \pm 3.6}$
	10	-1014 ± 70	-553.1 ± 6.4	$\mathbf{-513 \pm 4.4}$
Random Noise (std = 0.1)	1	-2.8 ± 0.1	-2.8 ± 0.1	-3.6 ± 0.1
	3	-68.8 ± 1.8	-45.1 ± 1.2	-59 ± 1
	5	-310.8 ± 33	-151.1 ± 1.3	$\mathbf{-149.5 \pm 3.4}$
	8	-737 ± 58.7	-364.3 ± 2.9	$\mathbf{-340.5 \pm 5.6}$
	10	-1384 ± 115	-557.6 ± 5.3	$\mathbf{-515.3 \pm 9}$
Random Noise (std = 0.2)	1	-2.8 ± 0.1	-2.8 ± 0.1	-3.6 ± 0.1
	3	-69.2 ± 1.8	-48.6 ± 2.2	-59.2 ± 2.7
	5	-290.2 ± 8	-152 ± 3.8	$\mathbf{-147.3 \pm 1.3}$
	8	-708.6 ± 16	-365.5 ± 8.9	$\mathbf{-336.7 \pm 1.3}$
	10	-939.7 ± 53	-550.4 ± 5.4	$\mathbf{-519.1 \pm 5.7}$

Figure 3 illustrates an example of comparison learning curves across four experimental conditions, with a sample number of agents set to five. The plots display learning curves from timestep 30,000, excluding the effect of initial steps where agents took random actions solely to fill the experience buffer without any gradient descent optimizations of the networks.

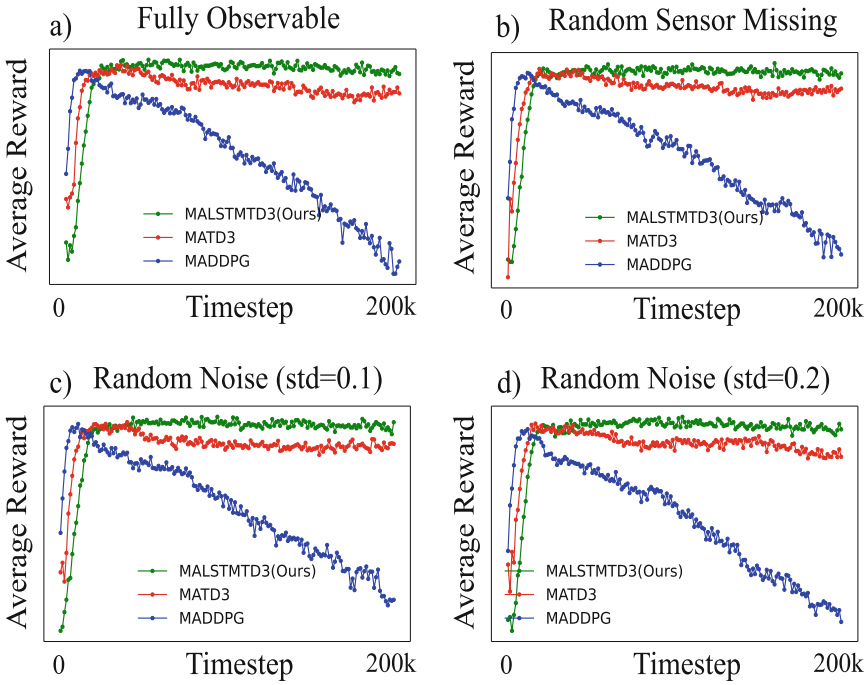

Fig. 3. Example learning curves for the purpose of visualization with number of agents = 5 across all experiments. a) fully observable, b) random sensor missing, c) random noise with std = 0.1 and d) random noise with std = 0.2.

5 Conclusions

In this study, we have illustrated the efficacy of augmenting the actor-critic architecture of the TD3 algorithm with LSTM units to enhance performance within cooperative multi-agent environments. Through experimentation across varying numbers of cooperating agents and different levels of partial observability, we have consistently observed enhancements in performance compared to the baseline methods in the more intricate settings featuring a higher number of agents, in specific for those exceeding three. These findings underscore the capacity of LSTM units to adeptly capture temporal dependencies, thereby facilitating improved anticipation and coordination of actions among agents over time.

Moving forward, there are a number of opportunities for further exploration and refinement of this approach. Future investigations may involve extending the analysis to encompass non-homogeneous agent populations, such as those encountered in adversarial environments, and by implementing in the model biologically inspired learning rules [19–23]. Furthermore, conducting empirical analyses to examine the dynamics of encoded information by LSTM units, and elucidating their contributions to enhancing overall environmental adaptability, represents a valuable direction for future research and development endeavors.

Acknowledgments. This work was supported by Compute Canada, the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Canadian Institutes of Health Research (CIHR) grants to Artur Luczak.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
2. Eßer, J., et al.: Guided reinforcement learning: a review and evaluation for efficient and effective real-world robotics [Survey]. IEEE Robot. Autom. Mag. **30**(2), 67–85 (2022)
3. Zhou, Z., Liu, G., Tang, Y.: Multi-agent reinforcement learning: methods, applications, visionary prospects, and challenges. arXiv preprint arXiv:2305.10091 (2023)
4. Li, Y.: Deep reinforcement learning: an overview. arXiv preprint arXiv:1701.07274 (2017)
5. Gronauer, S., Diepold, K.: Multi-agent deep reinforcement learning: a survey. Artif. Intell. Rev., 1–49 (2022)
6. Nguyen, T.T., Nguyen, N.D., Nahavandi, S.: Deep reinforcement learning for multiagent systems: a review of challenges, solutions, and applications. IEEE Trans. Cybern. **50**(9), 3826–3839 (2020)
7. Zhou, Y., et al.: On memory mechanism in multi-agent reinforcement learning. arXiv preprint arXiv:1909.05232 (2019)
8. Pesce, E., Montana, G.: Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication. Mach. Learn. **109**(9–10), 1727–1747 (2020)
9. Wierstra, D., et al.: Solving deep memory POMDPs with recurrent policy gradients. In: Artificial Neural Networks–ICANN 2007: 17th International Conference, Porto, Portugal, 9–13 September 2007, Proceedings, Part I 17. Springer, Heidelberg (2007)
10. Hausknecht, M., Stone, P.: Deep recurrent q-learning for partially observable MDPS. In: 2015 AAAI Fall Symposium Series (2015)
11. Bellman, R.: A Markovian decision process. J. Math. Mech., 679–684 (1957)
12. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Machine Learning Proceedings 1994, pp. 157–163. Morgan Kaufmann (1994)
13. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
14. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning. PMLR (2018)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
16. Meng, L., Gorbet, R., Kulić, D.: Memory-based deep reinforcement learning for POMDPs. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE (2021)
17. Lowe, R., et al.: Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
18. Terry, J., et al.: PettingZoo: gym for multi-agent reinforcement learning. Adv. Neural. Inf. Process. Syst. **34**, 15032–15043 (2021)

19. Kubo, Y., Chalmers, E., Luczak, A.: Combining backpropagation with equilibrium propagation to improve an actor-critic reinforcement learning framework. Front. Comput. Neurosci. **16**, 980613 (2022)
20. Kubo, Y., Chalmers, E., Luczak, A.: Biologically-inspired neuronal adaptation improves learning in neural networks. Commun. Integr. Biol. **16**(1), 2163131 (2023)
21. Chalmers, E., Gruber, A.J., Luczak, A.: HippoCluster: an efficient, hippocampus-inspired algorithm for graph clustering. Inf. Sci. **639**, 118999 (2023)
22. Chalmers, E., Luczak, A.: Reinforcement learning with brain-inspired modulation improves adaptation to environmental changes. In: International Conference on Artificial Intelligence and Soft Computing. Springer, Cham (2023)
23. Luczak, A., McNaughton, B.L., Kubo, Y.: Neurons learn by predicting future activity. Nat. Mach. Intell. **4**(1), 62–72 (2022)