Investigation of Independent Reinforcement Learning Algorithms in Multi-Agent Environments

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Independent reinforcement learning algorithms have no theoretical guarantees for finding the best policy in multi-agent settings. However, in practice, prior works have reported good performance with independent algorithms in some domains and bad performance in others. Moreover, a comprehensive study of the strengths and weaknesses of independent algorithms is lacking in the literature. In this paper, we carry out an empirical comparison of the performance of independent algorithms on four PettingZoo environments that span the three main categories of multi-agent environments, i.e., cooperative, competitive, and mixed. We show that in fully-observable environments, independent algorithms can perform on par with multi-agent algorithms in cooperative and competitive settings. For the mixed environments, we show that agents trained via independent algorithms learn to perform well individually, but fail to learn to cooperate with allies and compete with enemies. We also show that adding recurrence improves the learning of independent algorithms in cooperative partially observable environments.

1 Introduction

One of the simplest ways to apply reinforcement learning in multi-agent settings is to assume that all agents are independent of each other. In other words, every other agent is seen as part of the environment from any agent's perspective. Independent algorithms (i.e., single-agent algorithms) face the issue of non-stationarity in the multi-agent domain due to the violation of the Markovian property in a Markov Decision Process [1]. As a result, independent algorithms lack convergence guarantees, and are not theoretically sound in the multi-agent setting [2]. Despite these shortcomings, independent algorithms have the advantage of requiring lower computational resources and being easier to scale to large environments than traditional multi-agent algorithms which perform exact opponent modelling of each agent. In practice, prior works have reported mixed performance for independent algorithms in different multi-agent domains [3]–[9]. However, a study of the strengths and weaknesses of independent algorithms across various categories within the multi-agent domain is lacking in the literature.

In this paper, we investigate the empirical performance of independent algorithms in multi-agent settings, and compare them to various multi-agent algorithms under the Centralized Training and Decentralized Execution scheme [10], [11]. We evaluate these algorithms on 4 multi-agent environments from the PettingZoo library [12], which span the 3 main categories of multi-agent environments (i.e., cooperative, competitive and mixed) [13]–[16]. We show that independent algorithms can perform on par with multi-agent algorithms in the cooperative, fully-observable setting, and adding recurrence allows them to perform well compared to multi-agent algorithms in partially observable environments. In the competitive setting, we show that parameter sharing alongside the addition of agent indicators allow independent algorithms to outperform some multi-agent algorithms, such as Multi-Agent Proximal Policy Optimization [17], and Multi-Agent Deep Deterministic Policy Gradient [8], in fully-observable environments. For the mixed setting, we show that agents of independent algorithms learn to perform well individually, but fail in learning to cooperate with allies and compete against enemies.

2 Background Information

In this section, we provide readers with a brief overview of the various concepts and algorithms that are used throughout the paper.

2.1 Reinforcement Learning

In Reinforcement Learning (RL), an agent interacts with the environment by making sequential decisions [18]. At every time step, denoted as t, the agent observes a state s_t from the environment, and takes an action a_t. This action is executed in the environment, which returns a reward r_t and the next state s_{t+1} that are determined by the reward function $R(s_t, a_t)$ and the transition probability, $P(s_{t+1}|s_t, a_t)$, respectively. Critically, $R(s_t, a_t)$ and $P(s_{t+1}|s_t, a_t)$ are part of the environment, and are usually unknown to the agent of a model-free RL algorithm. Since the transition probability $P(s_{t+1}|s_t, a_t)$ conditions the next state s_{t+1} purely on the current state s_t and action a_t, it satisfies the Markov property [19]. This interaction between the agent and the environment is called a Markov Decision Process (MDP) [20]. The objective of an RL agent is to learn a policy $\pi(a_t|s_t)$, which maps a state to an action that maximizes the expected cumulative reward it receives from the environment. This is written as $\sum_t \gamma^t r_t$, where $\gamma \in [0, 1)$ represents a discount factor on future rewards.

2.2 Multi-Agent Reinforcement Learning

The single-agent MDP framework is extended to the Multi-Agent Reinforcement Learning (MARL) setting in the form of stochastic games [21]. In an N-agent stochastic game, at every time step, each of the n agents, identified by $j \in \{1, 2, \ldots, n\}$ across all agents, takes an action u_t^j. The joint action $\boldsymbol{u_t} \triangleq \{u_t^1, \ldots, u_t^N\}$ determines the rewards obtained by each agent. State transitions of the environment are determined by the transition probability $P(s_{t+1}|s_t, u_t)$, which conditions on the state and the joint action at timestep t.

2.3 Centralized Training and Decentralized Execution

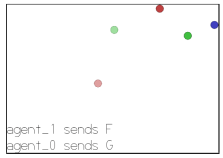
While it is technically possible to learn a centralized controller that maps a state to a distribution over the joint action space, the number of possible combinations of actions grows exponentially with the number of agents. This makes centralized control intractable for environments with many agents. Therefore, this paper is mainly focused on multi-agent algorithms which correspond to a Centralized Training and Decentralized Execution (CTDE) scheme [10], [11]. A CTDE algorithm has two phases. During the control phase, where policies are deployed in the environment, rather than using a centralized controller to take actions for all agents, decentralized agents make decisions based on their individual observations. During the prediction phase, centralized training is performed, which allows for extra information (e.g. the state) to be utilized, as long as this is not required during the control phase.

2.4 Cooperative, Competitive and Mixed

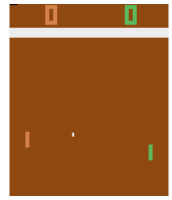
This paper follows the convention of classifying every multi-agent algorithm and environment studied into one of three categories – cooperative, competitive, or mixed (cooperative-competitive) [13]–[16].

In the cooperative setting, agents collaborate with each other to achieve a common goal. As a result, it is very common for all agents to share the same reward function (i.e., a team goal) [22]. Also known as the multi-agent credit assignment problem, every agent has to deduce its own contributions from the team reward [22]. Algorithms studied in this paper that explicitly address the multi-agent credit-assignment problem include QMIX [9] and Counterfactual Multi-Agent Policy Gradients (COMA) [7]. Additionally, the CommNet [23] extension on top of COMA is utilized for specific cooperative environments. Other multi-agent algorithms that are considered for the cooperative scenario include Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [8] and Multi-Agent Proximal Policy Optimization (MAPPO) [17].

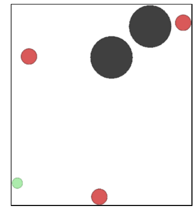
(a) Space Invaders (Atari, cooperative)



(b) Simple Reference (MPE, cooperative)



(c) Pong (Atari, competitive)



(d) Simple Tag (MPE, mixed)

Figure 1: The four PettingZoo environments used in the experiments. All figures were obtained from `https://pettingzoo.ml/`

In the competitive setting, agents play a zero-sum game, where one agent's gain is another agent's loss. In other words, $\sum_a r(s, u, a) = 0 \, \forall s, u$. Algorithms that are studied specifically in this paper include Deep Reinforcement Opponent Network (DRON) [24], MADDPG and MAPPO. MADDPG and MAPPO learn a separate critic for every agent, which gives the algorithms flexibility to learn different behaviours for agents with different reward functions.

In a mixed or cooperative-competitive setting, environments are neither zero-sum (competitive) nor cooperative, and they do not necessarily need to be general-sum either. A common setting would be environments that require every agent to cooperate with some agents, and compete with others [13]–[15]. MADDPG and MAPPO are used here for the same reason as the competitive setting.

2.5 Independent Algorithms and Non-Stationarity

One naive approach for applying single-agent RL to the multi-agent setting would be the use of independent learners, where each agent treats every other agent as part of the environment, and learns purely based on individual observations. In a multi-agent setting, the transition probability P and reward function R are conditioned on the joint action u. Since all agents in the environment are learning, their policies constantly change. Therefore, from each independent learner's perspective, the transition probability and reward function appear non-stationary, due to the lack of awareness of other agents' actions. This violates the Markovian property of an MDP, which then causes independent algorithms to be slow to adapt to other agents' changing policies, and as a result, face difficulties in converging to a good policy [24]–[26].

In this paper, we chose to use a popular off-policy algorithm, Deep Q-Network (DQN) [27], and an on-policy algorithm, Proximal Policy Optimization (PPO) [28]. In specific partially observable environments, Deep Recurrent Q-Network (DRQN) [29] is also utilized. Following the work of Gupta et al. [30], parameter sharing is utilized for all independent algorithms, such that experiences from all agents are trained simultaneously using a single network. This allows the training to be more efficient [30]. The aforementioned independent algorithms are tested in all 3 categories of multi-agent environments.

3 Experimental Setup

In this section, we introduce the environments used for the experiments, specify the various preprocessing that were applied, and other relevant implementation details.

3.1 Environments

The experiments were performed on multiple multi-agent environments from the PettingZoo library [12], which contains the Multi-Agent Particle Environments (MPE) [8], [31] and multi-agent variants of the Atari 2600 Arcade Learning Environment (ALE) [32], [33].

For the cooperative setting, experiments were performed on a modified version of the 2-player Space Invaders [32], [33], and the Simple Reference MPE environment [8], [31]. In Space Invaders (Fig. 1a), both agents share the common goal of shooting down all aliens. To make Space Invaders cooperative, we removed the (positive) reward that is given to a player whenever the other player gets hit. Additionally, the environment rewards every agent individually by default. Since a number of cooperative multi-agent algorithms (e.g., QMIX and COMA) assume that only a team reward is given, we modified the reward function such that a team reward is given instead (i.e., both agents receive the sum of their individual rewards). This setup exemplifies the multi-agent credit assignment problem, the effect of which is studied more closely in the Section 4.1.1. On the other hand, in the Simple Reference environment (Fig. 1b), two agents are rewarded by how close they are to their target landmark. However, the target landmark of an agent is only known by the other agent, as a result communication is required for both agents to navigate successfully to their target landmarks.

For the competitive setting, we performed experiments on the 2-player variant of the original Atari Pong environment (Fig. 1c). For the mixed setting, we opted for the Simple Tag MPE environment (Fig. 1d), which is a Predator-Prey environment [31]. This environment consists of 4 agents – 3 predators and a prey. The prey travels faster and has to avoid colliding with the predators, while the 3 predators travel slower and have to work together to capture the prey. The rewards received by the prey and a predator sum to 0 (i.e., the prey gets a negative reward for collision, while the predators get rewarded positively), and all predators receive the same reward. The prey is also negatively rewarded if it strays away from the predefined area (a 1×1 unit square). This environment is general-sum because it contains 3 predators and a single prey.

3.2 Preprocessing

For the MPE environments, no preprocessing was done, and default environment-parameters were used for all MPE experiments.

For the Atari environments, following the recommendations of Marlos et al. [34], we performed the following preprocessing - reward clipping, sticky actions, frame skipping, and no-op resets. The number of steps per episode was also set to a limit of 200 for both Atari environments, as that yielded the best results in general. Furthermore, the action spaces for both Atari environments were shrunk to their effective action spaces in order to improve learning efficiency. For Pong specifically, we also concatenated a one-hot vector of the agent's index to the observations so that independent algorithms can differentiate one from the other when parameter sharing is utilized. Further details of the preprocessing performed can be found in Appendix A.

3.3 Implementation

Implementations of all algorithms were based on open-sourced libraries/reference implementations. Default hyperparameters were used for all algorithms, and no hyperparameter tuning was performed. Details of implementations, alongside their hyperparameters, can be found in Appendix C.

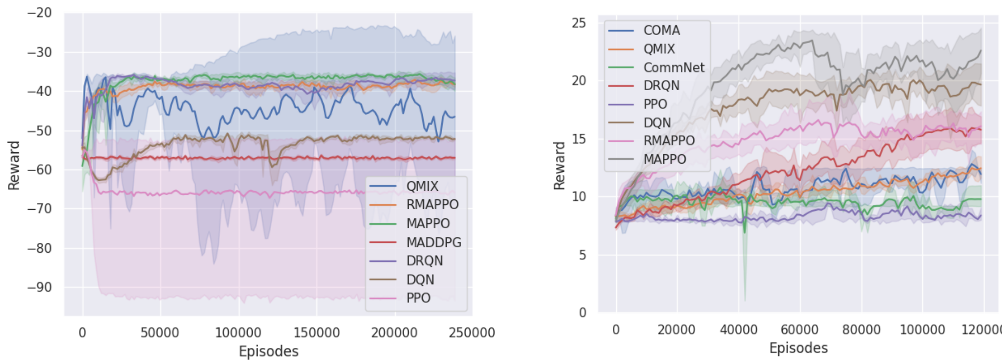
All experiments were performed across 5 different seeds. Parameter sharing was utilized for all algorithms throughout the experiments for all environments with homogeneous state and action spaces. For multi-agent algorithms that perform centralized training (e.g., QMIX, COMA, MADDPG etc.), the global states were represented by the concatenation of all agents' local observations. We also used the 128-byte Atari RAM as state inputs, rather than visual observations. This allows the algorithms to focus their learning on control rather than on both control and perception, improving learning efficiency.

4 Experiment Results

In this section, we highlight the experiments performed on the four multi-agent environments (i.e., Simple Reference, Space Invaders, Pong and Simple Tag), and provide discussions about the obtained results.

4.1 Cooperative

Simple Reference We ran the various algorithms on the Simple Reference environment for 240k episodes (6×10^6 steps). From Fig. 2a, it could be observed that all independent algorithms converged to a lower score, except for DRQN, whose recurrence allowed it to vastly outperform DQN and converge to a score on par with multi-agent algorithms. However, this trend was not observed when comparing MAPPO to its recurrent variant (i.e., RMAPPO), as MAPPO performs equally



(a) Simple Reference

(b) Space Invaders

Figure 2: Training curves of various algorithms in two cooperative environments. For every algorithm, the solid line represents the mean reward per episode, while the shaded region represents the 95% confidence interval around the mean.

well as RMAPPO. We hypothesize that since MAPPO's centralized critic learns based on the joint observation and action of both agents, this minimizes the amount of partial observability of every agent, and allows each agent to learn to communicate with other agents effectively without recurrence. In contrast, for independent algorithms, such as DQN, where the interactions between the agents are not explicitly learned (since all other agents are treated as part of the environment), adding recurrence could help mitigate some resulting partial observability, hence improving their performance, as described above.

Space Invaders Unlike the Simple Reference environment, the Space Invaders environment seemed to favour non-recurrent variants of algorithms (Fig. 2b). MAPPO vastly outperformed RMAPPO, and similarly DQN outperformed DRQN. This is also likely the underlying reasoning behind the comparatively poorer performance of the multi-agent algorithms, such as QMIX, COMA and CommNet, all of which were implemented with recurrent neural networks under the CTDE scheme.

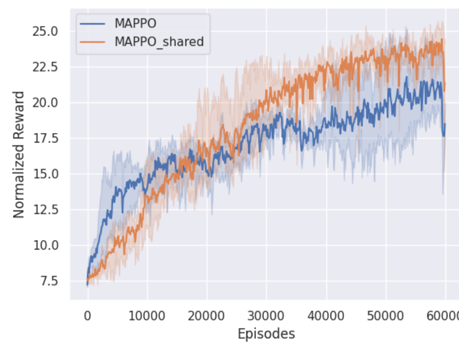
Additionally, since there is no unit collision in the Space Invaders environment (i.e., agents can move past each other without being blocked), they do not have to coordinate between themselves to achieve a high score in the environment; a good policy can be learned solely by having agents maximize their individual rewards. This explains the strong performance that was achieved by DQN. Also, since this is a cooperative task with both agents having identical goals, learning separate representations for individual agents is not very important; the learning of both agents assist each other. This is shown in Fig. 6b in Appendix B, where the addition of an agent indicator did not yield any performance improvement for DQN on Space Invaders.

Given such circumstances, it is interesting to observe the stronger performance of MAPPO compared to the independent algorithms. By conditioning on the joint action, MAPPO's critic has full observability into the joint action that resulted in the team reward. Therefore, the observed reward is unbiased, which allows the learning process to be more efficient. In contrast, independent algorithms have to learn from a noisy team reward signal, where an agent could receive a large positive team reward even when it did nothing. This relates to the problem of credit assignment in MARL, noted in prior works [35].

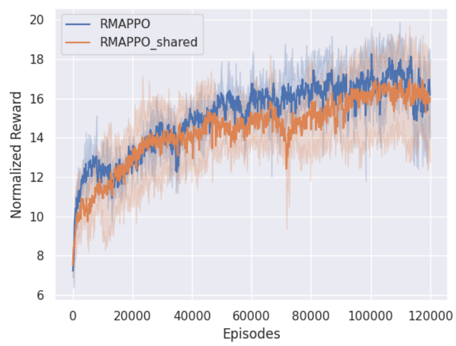
4.1.1 Multi-Agent Credit Assignment Problem in Fully Observable Settings

In this section, we attempt to study the effect of using a team reward signal, rather than individual reward signals on various independent and multi-agent algorithms in a fully observable environment. When team rewards are the only rewards given, these reward signals are noisy for independent algorithms because the agent, which treats every other agent as part of the environment, does not know the actions taken by other agents. This makes it difficult for independent algorithms' agents to learn how their individual actions contribute to the team reward signal. We performed the experiments on Space Invaders, in which the default agents receive individual rewards from the environment. To

(a) DQN



(b) MAPPO



(c) RMAPPO

Figure 3: Training curves of various algorithms in Space Invaders, comparing when individual rewards are given (blue) to when team rewards are given (orange).

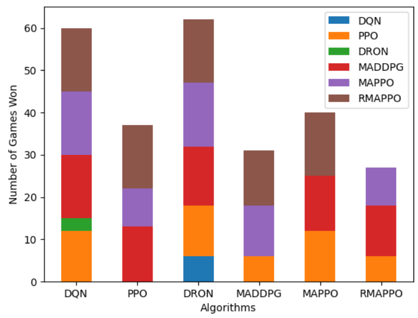
study the effect of the multi-agent credit assignment problem, we performed two runs per algorithm, one with team rewards only, and the other with individual rewards only (i.e., agents are rewarded independently by the environment).

For multi-agent algorithms, such as MAPPO (Fig. 3b) and RMAPPO (Fig. 3c), having a team reward does not have a large effect on the performance of the algorithms. This is expected because these algorithms have critics that learn from the joint action, which allow them to implicitly learn the estimated contribution of every agent without factorization.

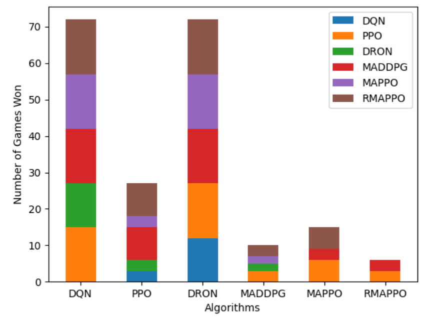
On similar lines, regarding independent algorithms, we observe that having team rewards instead of individual ones do not impact their performance adversely (Fig. 3a). A plausible explanation could be that since all agents receive the same reward for a given joint action, this allows the independent algorithms to correlate actions from different agents that produced similar (high) rewards.

4.2 Competitive

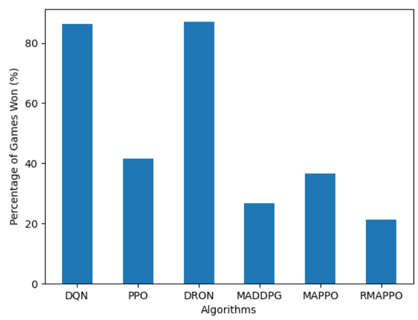
The 2-player Pong environment was used for the competitive setting. All algorithms were first trained using parameter sharing with the addition of agent indicators (the effect of which is detailed in Appendix B) for 60k episodes (1.2×10^6 steps), their network parameters were then saved. Since Pong is a zero-sum game, we evaluated them by putting them head-to-head against each other for 3 episodes for all possible combinations. After that, their positions were swapped, and the entire process was repeated. Swapping their positions is crucial for evaluation, because the first player (playing the right paddle) is always the serving player, therefore the first player always has an advantage over the second player (which plays the left paddle). This advantage is further exacerbated because the winning side always gets to serve subsequent openings. The entire evaluation process was repeated across all 5 seeds.



(a) Number of games won as the first player



(b) Number of games won as the second player



(c) Overall win rate percentage

Figure 4: Performance of various algorithms when playing against other algorithms in Pong.

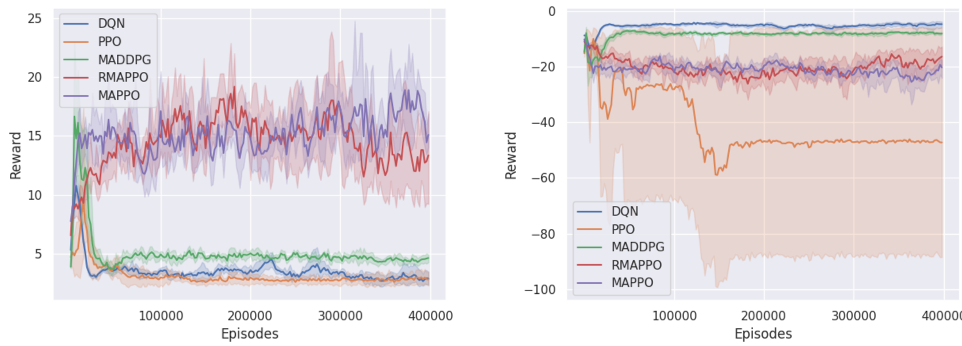
From the stacked bar charts shown in Fig. 4, a similar trend across the number of games won as the first and second player can be observed (Fig. 4a and 4b). DRON is consistently the best player, closely followed by DQN. Both of these algorithms were also the only algorithms to have a win rate of greater than 50% for the games they have played (Fig. 4c).

An interesting observation that can be made is the strong performance of independent algorithms, compared to other multi-agent algorithms. Since Pong is fully observable, critics that learn based on the joint observation of both agents do not necessarily provide any new information. Furthermore, since Pong is a highly reactive environment, an agent can learn a good policy solely by understanding how to position its paddle according to the trajectory of the ball (towards the agent). While learning on the joint action could allow agents to learn to better predict the incoming trajectory of the ball, it can be observed that the additional layer of complexity causes the sample efficiency to decrease and only yields diminishing returns.

In addition to the above factors, it is possible that parameter sharing benefited agents of independent algorithms by allowing them to learn better representations of both players, since they were trained to play as both players simultaneously. Had these algorithms trained without parameter sharing, there would likely be a larger performance difference between independent algorithms and opponent modelling algorithms such as DRON. Instead of treating other agents as part of the environment, opponent modelling allows agents to adapt more quickly to the opponent's changing strategies [24]. However, the minimal improvement DRON has over DQN suggests that in the Pong environment, an agent's policy may not be significantly affected by changes in the opponent agent's policy (i.e., individual agents can play the same way regardless of how their opponent played).

4.3 Mixed

In the Simple Tag (i.e., Predator-Prey) environment, the predators are incentivized to cooperate together to trap the prey, while the prey is incentivized to dodge the predators while staying within a predefined area. For our method of evaluation, we plot the training curves of the prey (Fig. 5b),



(a) Reward of a predator; all predators obtain the same reward

(b) Reward of the prey

Figure 5: Training curves of various algorithms in the Simple Tag, a Predator-Prey environment

and one of the predators (Fig. 5a), since all predators receive the same reward. Since the observation and action spaces differ between the predators and the prey, none of the agents have their parameters shared. We chose not to share the parameters of the predators to ensure that bias towards the predators was not introduced (since they would have 3 times the amount of data to work with compared to the prey).

In the case of DQN, the prey successfully learned to minimize the number of collisions with the predators, which can be observed by the strong performance achieved by the prey (Fig. 5b). However, similar to PPO, since the predators were trained completely independently (i.e., their parameters were not shared), they did not manage to learn how to cooperate with one another to capture the prey (Fig. 5a). It is interesting to observe that MADDPG converged to a policy similar to DQN, with the difference being that its predators have learned to cooperate better, thus getting slightly higher rewards compared to DQN's predators (Fig. 5a). Subsequently, as a result of the higher rewards obtained by the predators, MADDPG achieves a slightly lower score for its prey (Fig. 5b).

MAPPO and RMAPPO, on the other hand, learned a different strategy. As we can observe from the comparatively noisier curves obtained from their predators and preys (Fig. 5a and 5b), there is a constant tug-of-war between the prey and the predators - as the predators learn how to cooperate better, their scores increase, which subsequently causes the prey to learn how to dodge, decreasing the predators' scores, and vice versa. Since the predators of MAPPO and RMAPPO achieves a much higher score compared to all other algorithms, this is indicative that the predators have successfully learned to cooperate to trap the prey.

5 Conclusion

Cooperative In the cooperative setting, for environments where individual agents have full observability such as Space Invaders, we showed that independent algorithms can perform even better than certain multi-agent algorithms. Furthermore, we showed that independent algorithms are able to cope well with the multi-agent credit assignment problem in environments that are fully observable with a relatively small number of agents, and where every agent has the same task. On the other hand, in the Simple Reference environment where the need for agents to communicate induces partial observability, adding recurrence allowed independent algorithms to perform as well as other multi-agent algorithms. We also discussed the significance of learning on the joint observation and action, rather than individual ones, and showed that MAPPO performs as well as DRQN in the Simple Reference environment, without the need for an RNN. Moreover, in Space Invaders, MAPPO was able to consistently achieve the highest score amongst all other algorithms.

Competitive In the Pong environment, we saw that DRON and DQN were able to outperform all other algorithms. We argued that this is due to the fully observable nature of the Pong environment, in addition to the diminishing returns that learning from joint actions could yield. Furthermore, we showed that with the use of agent indicators, independent algorithms were able to learn robust policies for both competing agents using parameter sharing.

Mixed In the Predator-Prey environment, we saw that since there were no parameter sharing to induce cooperation, predators from independent algorithms were unable to learn how to cooperate with each other to capture the prey. Conversely, in DQN we saw that its prey was able to achieve the highest score consistently, showing that the prey has learned to dodge the predators effectively while staying within the predefined area. Interestingly, we also saw how MADDPG's training curve for its predators and prey shows resemblance to that of DQN, suggesting that it also faced difficulties in learning strategies for the predators to coordinate and capture the prey. MAPPO and RMAPPO, on the other hand, were the only algorithms that managed to achieve high scores for their predators, suggesting that their predators have learned how to collaborate with each other to hunt the prey. The noisiness of their graphs suggest that there is a constant tug-of-war between the prey and the predators, as one tries to outsmart the other.

6 Future Work

In this section, we highlight some future work that could potentially bring more insights into having a broader understanding of dealing with non-stationarity and partial observability for independent algorithms, both of which are common in the multi-agent setting. In the Space Invaders environment, we observed that independent algorithms were able to learn well with just a team reward. Future work could be done to determine if this was only the case for fully observable environments, or under what conditions would independent algorithms still be able to cope with the multi-agent credit assignment problem. It would also be interesting to study the performance of non-recurrent variants of multi-agent algorithms such as QMIX and COMA in fully observable environments. Since the experiments performed in this paper only included fully-observable competitive and mixed environments, future work can also include a more diverse set of environments, such as partially observable competitive and mixed environments.

References

[1] S. Choi, D.-Y. Yeung, and N. Zhang, "An environment model for nonstationary reinforcement learning," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12, MIT Press, 2000. [Online]. Available: `https://proceedings.neurips.cc/paper/1999/file/e8d92f99edd25e2cef48eca48320a1a5-Paper.pdf`.

[2] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *ICML*, 1993.

[3] E. Zawadzki, A. Lipson, and K. Leyton-Brown, *Empirically evaluating multiagent learning algorithms*, 2014. arXiv: `1401.8074 [cs.GT]`.

[4] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PLOS ONE*, vol. 12, no. 4, pp. 1–15, Apr. 2017. DOI: `10.1371/journal.pone.0172395`. [Online]. Available: `https://doi.org/10.1371/journal.pone.0172395`.

[5] Y. Shoham and K. Leyton-Brown, *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[6] OpenAI, : C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, *Dota 2 with large scale deep reinforcement learning*, 2019. arXiv: `1912.06680 [cs.LG]`.

[7] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: `https://ojs.aaai.org/index.php/AAAI/article/view/11794`.

[8] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, *Multi-agent actor-critic for mixed cooperative-competitive environments*, 2020. arXiv: `1706.02275 [cs.LG]`.

[9] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, Jul. 2018, pp. 4295–4304. [Online]. Available: `http://proceedings.mlr.press/v80/rashid18a.html`.

[10] L. Kraemer and B. Banerjee, "Multi-agent reinforcement learning as a rehearsal for decentralized planning," *Neurocomputing*, vol. 190, pp. 82–94, 2016, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2016.01.031. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231216000783.

[11] F. A. Oliehoek, M. T. J. Spaan, and N. A. Vlassis, "Optimal and approximate q-value functions for decentralized pomdps," *CoRR*, vol. abs/1111.0062, 2011. arXiv: 1111.0062. [Online]. Available: http://arxiv.org/abs/1111.0062.

[12] J. K. Terry, B. Black, M. Jayakumar, A. Hari, R. Sullivan, L. Santos, C. Dieffendahl, N. L. Williams, Y. Lokesh, C. Horsch, *et al.*, "Pettingzoo: Gym for multi-agent reinforcement learning," *arXiv preprint arXiv:2009.14471*, 2020.

[13] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008. DOI: 10.1109/TSMCC.2007.913919.

[14] L. Canese, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, and S. Spanò, "Multi-agent reinforcement learning: A review of challenges and applications," *Applied Sciences*, vol. 11, no. 11, 2021, ISSN: 2076-3417. DOI: 10.3390/app11114948. [Online]. Available: https://www.mdpi.com/2076-3417/11/11/4948.

[15] K. Zhang, Z. Yang, and T. Başar, *Multi-agent reinforcement learning: A selective overview of theories and algorithms*, 2021. arXiv: 1911.10635 [cs.LG].

[16] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: A survey," *Artificial Intelligence Review*, pp. 1–49, 2021.

[17] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, *The surprising effectiveness of mappo in cooperative, multi-agent games*, 2021. arXiv: 2103.01955 [cs.LG].

[18] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[19] A. A. Markov, "The theory of algorithms," *Trudy Matematicheskogo Instituta Imeni VA Steklova*, vol. 42, pp. 3–375, 1954.

[20] R. BELLMAN, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, 1957, ISSN: 00959057, 19435274. [Online]. Available: http://www.jstor.org/stable/24900506.

[21] L. S. Shapley, "Stochastic games," *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, pp. 1095–1100, 1953, ISSN: 0027-8424. DOI: 10.1073/pnas.39.10.1095. eprint: https://www.pnas.org/content/39/10/1095.full.pdf. [Online]. Available: https://www.pnas.org/content/39/10/1095.

[22] Y.-h. Chang, T. Ho, and L. Kaelbling, "All learning is local: Multi-agent learning in global reward games," in *Advances in Neural Information Processing Systems*, S. Thrun, L. Saul, and B. Schölkopf, Eds., vol. 16, MIT Press, 2004. [Online]. Available: https://proceedings.neurips.cc/paper/2003/file/c8067ad1937f728f51288b3eb986afaa-Paper.pdf.

[23] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16, Barcelona, Spain: Curran Associates Inc., 2016, pp. 2252–2260, ISBN: 9781510838819.

[24] H. He, J. Boyd-Graber, K. Kwok, and H. Daumé, "Opponent modeling in deep reinforcement learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16, New York, NY, USA: JMLR.org, 2016, pp. 1804–1813.

[25] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with non-stationarity in multi-agent deep reinforcement learning," *CoRR*, vol. abs/1906.04737, 2019. arXiv: 1906.04737. [Online]. Available: http://arxiv.org/abs/1906.04737.

[26] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote, *A survey of learning in multiagent environments: Dealing with non-stationarity*, 2019. arXiv: 1707.09183 [cs.MA].

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[29] M. Hausknecht and P. Stone, *Deep recurrent q-learning for partially observable mdps*, 2017. arXiv: 1507.06527 [cs.LG].

[30] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Autonomous Agents and Multiagent Systems*, G. Sukthankar and J. A. Rodriguez-Aguilar, Eds., Cham: Springer International Publishing, 2017, pp. 66–83, ISBN: 978-3-319-71682-4.

[31] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," *arXiv preprint arXiv:1703.04908*, 2017.

[32] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, Jun. 2013.

[33] J. K. Terry and B. Black, "Multiplayer support for the arcade learning environment," *arXiv preprint arXiv:2009.09341*, 2020.

[34] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. J. Hausknecht, and M. Bowling, "Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents," *CoRR*, vol. abs/1709.06009, 2017. arXiv: `1709.06009`. [Online]. Available: `http://arxiv.org/abs/1709.06009`.

[35] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, vol. 33, no. 6, pp. 750–797, 2019.

[36] J. K. Terry, B. Black, and A. Hari, "Supersuit: Simple microwrappers for reinforcement learning environments," *arXiv preprint arXiv:2008.08932*, 2020.

[37] M. Li, *Machin*, `https://github.com/iffiX/machin`, 2020.

[38] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, *Stable baselines3*, `https://github.com/DLR-RM/stable-baselines3`, 2019.

[39] starry-sky6688, *Starcraft*, `https://github.com/starry-sky6688/StarCraft`, 2019.

[40] H. van Hasselt, A. Guez, and D. Silver, *Deep reinforcement learning with double q-learning*, 2015. arXiv: `1509.06461 [cs.LG]`.

[41] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, *Dueling network architectures for deep reinforcement learning*, 2016. arXiv: `1511.06581 [cs.LG]`.

[42] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, *Prioritized experience replay*, 2016. arXiv: `1511.05952 [cs.LG]`.