Multiagent Cooperation and Competition with Deep Reinforcement Learning

Ardi Tampuu* **Tambet Matiisen***

Dorian Kodelja **Ilya Kuzovkin** **Kristjan Korjus**

Juhan Aru[†] **Jaan Aru** **Raul Vicente**[✉]

Computational Neuroscience Lab, Institute of Computer Science, University of Tartu
[†] Department of Mathematics, ETH Zurich
ardi.tampuu@ut.ee, tambet.matiisen@ut.ee, raul.vicente.zafra@ut.ee

* *these authors contributed equally to this work*

Abstract

Multiagent systems appear in most social, economical, and political situations. In the present work we extend the Deep Q-Learning Network architecture proposed by Google DeepMind to multiagent environments and investigate how two agents controlled by independent Deep Q-Networks interact in the classic videogame *Pong*. By manipulating the classical rewarding scheme of Pong we demonstrate how competitive and collaborative behaviors emerge. Competitive agents learn to play and score efficiently. Agents trained under collaborative rewarding schemes find an optimal strategy to keep the ball in the game as long as possible. We also describe the progression from competitive to collaborative behavior. The present work demonstrates that Deep Q-Networks can become a practical tool for studying the decentralized learning of multiagent systems living in highly complex environments.

Introduction

In the ever-changing world biological and engineered agents need to cope with unpredictability. By learning from trial-and-error an animal or a robot can adapt its behavior in a novel or changing environment. This is the main intuition behind reinforcement learning [SB98, PM10]. A reinforcement learning agent modifies its behavior based on the rewards that it collects while interacting with the environment. By trying to maximize the reward during these interactions an agent can learn to implement complex long-term strategies.

Due to the astronomic number of states of any realistic scenario, for a long time algorithms implementing reinforcement learning were either limited to simple environments or needed to be assisted by additional information about the dynamics of the environment. Recently, however, the Swiss AI Lab IDSIA [KCSG13] and Google DeepMind [MKS$^+$13, MKS$^+$15] have produced spectacular results in applying reinforcement learning to very high-dimensional and complex environments such as video games. In particular, DeepMind demonstrated that AI agents can achieve superhuman performance in a diverse range of Atari video games. Remarkably, the learning agent only used raw sensory input (screen images) and the reward signal (increase in game score). The proposed methodology, so called Deep Q-Network, combines a convolutional neural network for feature representation with Q-learning training [Lin93]. It represents the state-of-the-art approach for model-

free reinforcement learning problems in complex environments. The fact that the same algorithm was used for learning very different games suggests its potential for general purpose applications.

The present article builds on the work of DeepMind and explores how multiple agents controlled by autonomous Deep Q-Networks interact when sharing a complex environment. Multiagent systems appear naturally in most of social, economical and political scenarios. Indeed, most of game theory problems deal with multiple agents taking decisions to maximize their individual returns in a static environment [BBDS08]. Collective animal behavior [Sum10] and distributed control systems are also important examples of multiple autonomous systems. Phenomena such as cooperation, communication, and competition may emerge in reinforced multiagent systems.

The goal of the present work is to study emergent cooperative and competitive strategies between multiple agents controlled by autonomous Deep Q-Networks. As in the original article by DeepMind, we use Atari video games as the environment where the agents receive only the raw screen images and respective reward signals as input. We explore how two agents behave and interact in such complex environments when trained with different rewarding schemes.

In particular, using the video game *Pong* we demonstrate that by changing the rewarding schemes of the agents either competitive or cooperative behavior emerges. Competitive agents get better at scoring, while the collaborative agents find an optimal strategy to keep the ball in the game for as long as possible. We also tune the rewarding schemes in order to study the intermediate states between competitive and cooperative modes and observe the progression from competitive to collaborative behavior.

1 Methods

1.1 The Deep Q-Learning Algorithm

The goal of reinforcement learning is to find a policy – a rule to decide which action to take in each of the possible states – which maximizes the agent's accumulated long term reward in a dynamical environment. The problem is especially challenging when the agent must learn without the explicit information about the dynamics of the environment or the rewards. In this case perhaps the most popular learning algorithm is Q-learning [Wat89]. Q-learning allows one to estimate the value or quality of an action in a particular state of the environment.

Recently Google DeepMind trained convolutional neural networks to approximate these so-called Q-value functions. Leveraging the powerful feature representation of convolutional neural networks the so-called Deep Q-Networks have obtained state of the art results in complex environments. In particular, a trained agent achieved superhuman performance in a range of Atari video games by using only raw sensory input (screen images) and the reward signal [MKS$^+$15, SQAS15].

When two or more agents share an environment the problem of reinforcement learning is much less understood. The distributed nature of the learning offers new benefits but also challenges such as the definition of good learning goals or the convergence and consistency of algorithms [Sch14, BBDS08]. For example, in the multiagent case the environment state transitions and rewards are affected by the joint action of all agents. This means that the value of an agent's action depends on the actions of the others, and hence each agent must keep track of each of the other learning agents, possibly resulting in an ever-moving target. In general, learning in the presence of other agents requires a delicate trade-off between the stability and adaptive behavior of each agent.

There exist several possible adaptations of the Q-learning algorithm for the multiagent case. However, this is an open research area and theoretical guarantees for multiagent model-free reinforcement learning algorithms are scarce and restricted to specific types of tasks [Sch14, BBDS08]. In practice the simplest method consists of using an autonomous Q-learning algorithm for each agent in the environment, thereby using the environment as the sole source of interaction between agents. In this work we use this method due to its simplicity, decentralized nature, computational speed, and ability to produce consistent results for the range of tasks we report. Therefore, in our tests each agent is controlled by an independent Deep Q-Network with architecture and parameters as reported in [MKS$^+$15].

1.2 Adaptation of the Code for the Multiplayer Paradigm

The original code published with [MKS+15] does not provide the possibility to play multiplayer games. Whereas the agents are independent realizations of Deep Q-Networks and many of the Atari games allow multiple players, the communication protocol between the agents and the emulator restricts one to use a single player. To solve this issue we had to modify the code to allow transmitting actions from two agents, as well as receiving two sets of rewards from the environment. The game screen is fully observable and is shared between the two agents, hence no further modifications were needed to provide the state of the game to multiple agents simultaneously.

1.3 Game Selection

Atari Learning Environment (ALE) [BNVB12] currently supports 61 games, but only a handful on them have a two player mode. In order to choose a suitable game for our multiplayer experiments we used following criteria:

1. The game must have real-time two-player mode. Many games (e.g. *Breakout*) alternate between two players and are therefore less suitable for our multiagent learning experiments.
2. Deep Q-learning algorithm must be able to play the game above human level in single player mode. For example *Wizard of Wor* has a two-player mode, but requires extensive labyrinth navigation, which current deep Q-learning algorithm is not able to master.
3. The game must have a natural competitive mode. In addition, we were interested in games, where we can switch between cooperation and competition by a simple change of reward function.

After having considered several other options we finally chose to work within the *Pong* game environment because it satisfies all the criteria, it was supported by existing code and can be learned relatively quickly. It also has the advantage of being easily understood by the reader due to its simplicity and its role in the video game history.

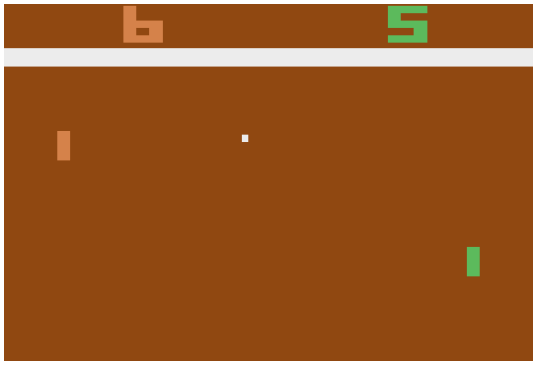


Figure 1: The *Pong* game. Each agent corresponds to one of the paddles.

In *Pong* each agent corresponds to one of the paddles situated on the left and right side of the screen (see Figure 1). There are 4 actions that each of the two agents can take: move up, move down, stand still, and fire (to launch the ball or to start the game). Which action is taken is decided by the corresponding Deep Q-Network for both agents separately.

While this is outside the scope of the present work, we would like to report on two other games which might mix well with interesting scientific questions on competition and collaboration. Firstly, *Outlaw* is a simple shooting game that in a restricted game mode can be seen as a real-time approximation of prisoner's dilemma. Secondly, *Warlords* is a game with up to four players where the emergence of collaboration in the face of an adversary could be tested.

1.4 Rewarding Schemes

A central aim of this work is to study the emergence of a diversity of coordinated behaviours depending on how the agents are rewarded. Here we describe the different rewarding schemes we set for the agents. We adjust rewarding schemes by simply changing the reward both players get when the ball goes out of game. In such a way we can create several very different games within the same *Pong* environment.

1.4.1 Score More than the Opponent (Fully Competitive)

In the traditional rewarding scheme of Pong, also used in [MKS+15], the player who last touches the outgoing ball gets a plus point, and the player losing the ball a minus point. This makes it essentially a zero-sum game, where a positive reward for the left player means negative reward for right player and vice versa. We call this fully competitive mode as the aim is to simply try to put the ball behind your opponent.

	Left player scores	Right player scores
Left player reward	$+1$	-1
Right player reward	-1	$+1$

Table 1: Rewarding scheme for the classical case of *Pong*. This rewarding scheme leads to a competitive strategy.

1.4.2 Loosing the Ball Penalizes Both Players (Fully Cooperative)

In this setting we want the agents to learn to keep the ball in the game for as long as possible. To achieve this, we penalize both of the players whenever the ball goes out of play. Which of the players let the ball pass does not matter and no positive rewards are given.

	Left player scores	Right player scores
Left player reward	-1	-1
Right player reward	-1	-1

Table 2: Rewarding scheme to induce cooperative strategy.

Notice that another possible collaborative mode would be to reward both players on each outgoing ball, but we expected this mode not to show any interesting behaviour.

1.4.3 Transition Between Cooperation and Competition

The fully competitive and fully cooperative cases both penalize loosing the ball equally. What differentiates the two strategies are the values on the main diagonal of the reward matrix, i.e. the rewards agents get for putting the ball past the other player. If one allows this reward value to change gradually from -1 to $+1$, one can study intermediate scenarios that lie between competition and cooperation. We ran a series of experiments to explore the changes in the behaviour when the reward for "scoring" a point, ρ, changes from -1 to $+1$ while the penalty for losing the ball is kept fixed at -1.

	Left player scores	Right player scores
Left player reward	ρ	-1
Right player reward	-1	ρ

Table 3: Rewarding scheme to explore the transition from competitive to the cooperative strategy where $\rho \in [-1, 1]$.

1.5 Training Procedure

In all of the experiments we let the agents learn for 50 epochs [1], 250000 time steps each. Due to using a frame skipping technique the agents see and select actions only on every 4th frame. In the following we use "visible frame", "frame" and "time step" interchangeably.

During the training time , as in [MKS+15], the exploration rate (percentage of actions chosen randomly) decreases from an initial 1.0 to 0.05 in million time steps and stays fixed at that value. Here, selecting a random action simply means that we draw a command randomly from amongst all the possible actions in the game instead of using the prediction that would have been made by the Deep Q-Network. A more detailed description of the training procedure and parameters can be found in [MKS+15].

The convergence of Q-values is an indicator of the convergence of the deep Q-network controlling the behaviour of the agent. Hence, we monitor the *average maximal Q-values* of 500 randomly selected game situations, set aside before training begins. We feed these states to the networks after each training epoch and record the maximal value in the last layer of each of the Deep Q-Networks. These maximal values correspond to how highly the agent rates its best action in each of the given states and thus estimates the quality of the state itself.

After each epoch snapshots of the Deep Q-Networks are stored to facilitate the future study of the training process.

1.6 Collecting the Game Statistics

To obtain quantitative measures of the agents' behaviour in the *Pong* environment, we identified and counted specific events in the game, e.g. bouncing of the ball against the paddle or the wall. We used Stella [MA03] integrated debugger to detect these events. Stella makes it possible to identify the exact memory locations that hold the numbers of bounces made by the players as well as several other game events.

To quantitatively evaluate the emergence of cooperative or competitive behaviour we collected the measures after each training epoch for all of the rewarding schemes. After the end of each epoch we took the deep Q-networks of both players in their current state and had them play 10 games [2] using different random seeds to obtain the statistics on the measures. During the testing phase the exploration rate was set to 0.01. The behavioral measures we used are the following:

- **Average *paddle-bounces* per point** counts how many times the ball bounces between two players before one of them scores a point. Randomly playing agents almost never hit the ball. Well trained agents hit the ball multiple times in an exchange. Hereafter we refer to this statistic as *paddle-bounces*.

- **Average *wall-bounces* per paddle-bounce** quantifies how many times the ball bounces from top and bottom walls before it reaches the other player. It is possible to hit the ball in an acute angle so that it bounces the walls several times before reaching the other player. Depending on the strategy, players might prefer to send the ball directly to the other player or use the wall bounces. Hereafter we refer to this statistic as *wall-bounces*.

- **Average *serving time* per point** shows how long it takes for the players to restart the game after the ball was lost (measured in frames). To restart the game, the agent who just scored has to send a specific command (fire). Depending on the rewarding scheme the agents might want to avoid restarting the game. Hereafter we refer to this statistic as *serving time*.

[1] We limit the learning to 50 epochs because the Q-values predicted by the network have stabilized (see Supplementary Materials Figure 8)

[2] In *Pong* one game consists of multiple exchanges and lasts until one of the agents reaches 21 points

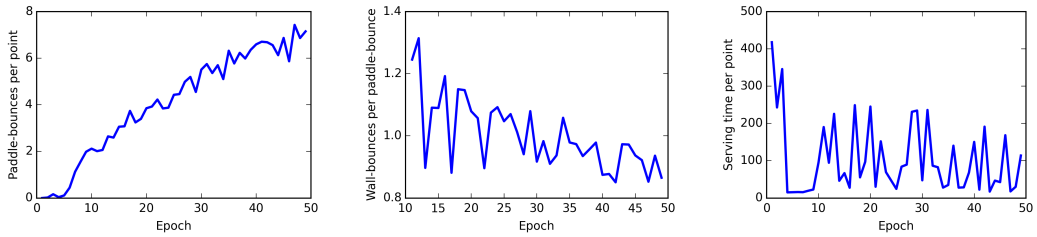
2 Results

In this section we explain our main result: the emergence of competitive and collaborative behaviors in the playing strategies of the learning agents.

2.1 Emergence of Competitive Agents

In the full competitive (zero-sum) rewarding scheme each agent obtains an immediate reward when the ball gets past the other agent and an immediate punishment when it misses the ball.

Initially the agents fail to hit the ball at all but with training both agents become more and more proficient. The learning of both agents progresses continuously. Figure 2 summarizes the evolution of the quantitative descriptors of behaviour during training.



(a) Paddle-bounces per point

(b) Wall-bounces per paddle-bounce

(c) Serving time per point

Figure 2: Evolution of the behaviour of the competitive agents during training. (a) The number of paddle-bounces increases indicating that the players get better at catching the ball. (b) The frequency of the ball hitting the upper and lower walls decreases slowly with training. The first 10 epochs are omitted from the plot as very few paddle-bounces were made by the agents and the metric was very noisy. (c) Serving time decreases abruptly in early stages of training- the agents learn to put the ball back into play. Serving time is measured in frames.

Qualitatively we can report that by the end of training both agents are capable of playing the game reasonably well. First of all, both players are capable of winning regardless of who was the one to serve. Secondly, the exchanges can last for a considerable amount of touches (Figure 2a) even after the speed of the ball has increased. Thirdly we observe that the agents have learned to put the ball back in play rapidly (Figure 2c).

Figure 3 illustrates how the agents' predictions of their rewards evolve during an exchange. A first observation is that the Q-values predicted by the agents are optimistic, both players predicting positive future rewards in most situations. The figure also demonstrates that the agents' reward expectations correlate well with game situations. More precisely, one can observe that whenever the ball is travelling towards a player its reward expectation drops and the opponent's expectation increases. This drop occurs because even a well trained agent might miss the ball (at least 5% of the actions are taken randomly during training) resulting in -1 and $+1$ rewards for the two players respectively.

We also note that the Q-values correlate with the speed of the ball (data not shown). The faster the ball travels, the bigger are the changes in Q-values - possibly because there is less time to correct for a bad initial position or a random move.

Interestingly, one can also notice that as soon as the serving player has relaunched the ball its reward expectation increases slightly. This immediate increase in Q-value makes the agents choose to serve the ball as soon as possible and thus explains the decrease in serving time (Figure 2c).

See section 4.3 for a short example video displaying the behavior of the agents in different game situations and the corresponding Q-values.

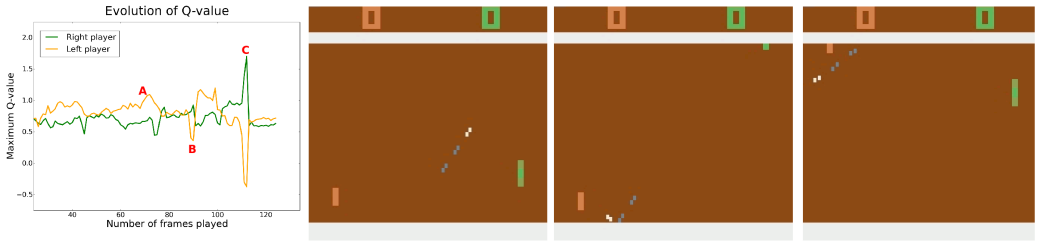
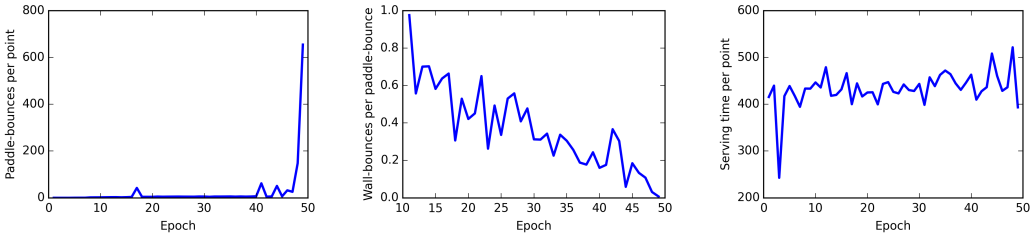


Figure 3: A competitive game - game situations and the Q-values predicted by the agents. A) The left player predicts that the right player will not reach the ball as it is rapidly moving upwards. B) A change in the direction of the ball causes the left player's reward expectation to drop. C) Players understand that the ball will inevitably go out of the play. See section 4.3 for videos illustrating other game situations and the corresponding agents' Q-values.

2.2 Emergence of Collaborative Agents

In the fully cooperative rewarding scheme each agent receives an immediate punishment whenever the ball get past either of the agents. Thus, the agents' are motivated to keep the ball alive. The agents get no positive rewards and the best they can achieve is to minimize the number of times the ball is lost. The evolution of the quantitative descriptors of behaviour during cooperative training is



(a) Paddle-bounces per point

(b) Wall-bounces per paddle-bounce

(c) Serving time per point

Figure 4: Evolution of the behaviour of the collaborative agents during training. (a) The number of paddle-bounces increases as the players get better at reaching the ball. (b) The frequency of the ball hitting the upper and lower walls decreases significantly with training. The first 10 epochs are omitted from the plot as very few paddle-bounces were made by the agents and the metric was very noisy. (c) Serving time increases - the agents learn to postpone putting the ball into play. Serving time is measured in frames.

shown on Figure 4. The emergent strategy after 50 epochs of training can be characterized by three observations: (i) the agents have learned to keep the ball for a long time (Figure 4a); (ii) the agents take a long time to serve the ball (Figure 4c) because playing can only result in negative rewards; and (iii) the agents prefer to pass the ball horizontally across the field without touching the walls (Figure 4b). Notice that the frequency of wall-bounces decreases gradually with training, whereas the number of paddle-bounces increases abruptly.

On Figure 5 an exchange between collaborative agents is illustrated. Just like the competitive agents, the collaborative agents learn that the speed of the ball is an important predictor of future rewards - faster balls increase the risk of mistakes. The clear drop in the predicted Q-values in situation B compared to situation A is caused by the ball travelling faster in situation B.

In the exchange illustrated on Figure 5 the agents eventually miss the ball. In some exchanges, however, the players apply a coordinated strategy where both agents place themselves at the upper border of the playing field and bounce the ball between themselves horizontally (Figure 6).

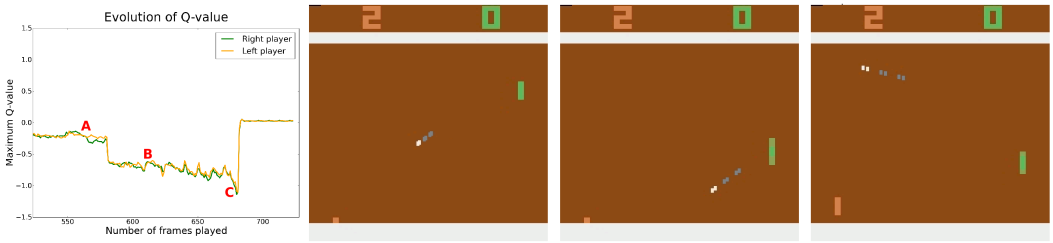


Figure 5: Cooperative game. A) The ball is moving slowly and the future reward expectation is not very low - the agents do not expect to miss the slow balls. B) The ball is moving faster and the reward expectation is much more negative - the agents expect to miss the ball in the near future. C) The ball is inevitably going out of play. Both agents' reward expectations drop accordingly. See section 4.3 for videos illustrating other game situations and the corresponding agents' Q-values.

Whenever a random action takes them away from the edge, they move back towards the edge in the next time step. Note that being at the edge of the field minimizes the effect of random actions - random movements to only one of the two directions are possible. Arriving to the stable situation happens only in some exchanges and might depend on the way the ball is served. It is made harder by the fact that agents choose actions every 4th frame.

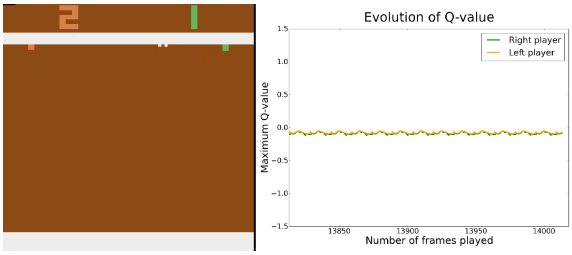


Figure 6: In the cooperative setting the agents sometimes reach a strategy that allows them to keep the ball in the game for a very long time.

See section 4.3 for a video illustrating the evolution of the agents' learning progression towards the final converged behaviour and coordinated strategy.

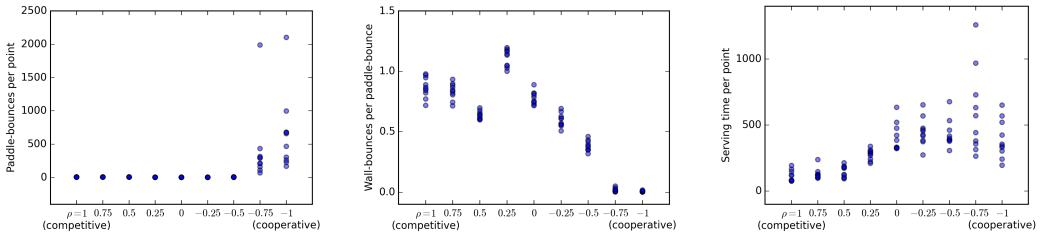
2.3 Progression from Competition to Collaboration

Besides the two cases described above, we also ran a series of simulations with intermediate rewarding schemes. Our aim here is to describe the transition from competitive to collaborative behaviour when the immediate reward received for scoring a point (ρ) is decreased.

On Figure 7, the quantitative behavioural metrics are plotted for decreasing values of ρ in order to give a better overview of the trends. Table 4 summarises these results numerically. The statistics are collected after agents have been trained for 50 epochs and are averaged over 10 game runs with different random seeds.

For $\rho = 1$, $\rho = 0.75$, and $\rho = 0.5$ the number of paddle-touches per point stays the same, indicating that the learning algorithm is not very sensitive to the exact size of the reward in that range. However, we observe a significant decrease in the number of touches when we lowered ρ from 0.5 to 0.25 (Figure 7a, Table 4).

Also the number of wall-bounces per paddle-bounces is the same for reward values 1 and 0.75 (Figure 7b, Table 4). Lowering ρ to 0.5, the wall-bounces become significantly less frequent indicating some change of behaviour that was not captured by the number of touches. Interestingly, $\rho = 0.25$ does not continue the decreasing trend and instead results in the highest number of wall-bounces per paddle-bounce among all the tested rewarding schemes.



(a) Paddle-bounces per point

(b) Wall-bounces per paddle-bounce

(c) Serving time per point

Figure 7: Progression of behavioural statistics when passing from competitive to collaborative rewarding schemes. (a) The number of touches increases when the agents have a strong incentive to collaborate. (b) Forcing the agents to collaborate decreases the amount of angled shots that bounce off the walls before reaching the opposite player. (c) Serving time is significantly shorter when agents receive positive rewards for scoring.

Also, for $\rho > 0$, weaker positive reward leads to a longer serving time (Figure 7c, Table 4). Let us remind that to restart the game, the agent who just scored has to send a specific command (fire). If the agent never chooses to send this command, the game may still restart via actions randomly selected by the ϵ-greedy exploration. In such case serving takes on average a bit more than 400 frames. This means that the agents trained with $\rho > 0$ actively choose to restart the game at least in some game situations.

Conversely, in all the rewarding schemes with $\rho \leq 0$, the agents learn to avoid relaunching the ball. The average serving times are around 400 frames and correspond to those obtained when agents only start the game due to randomly chosen actions (Figure 7c, Table 4).

For $\rho \leq 0$ we also observe that increasingly negative rewards leads to keeping the ball alive longer (Figure 7a, Table 4) and to hitting the walls less often (Figure 7b). The increased dispersion in statistics for $\rho = -1$ and $\rho = -0.75$ is the result of agents sometimes reaching the strategy of keeping the ball infinitely.

Agent	Average paddle-bounces per point	Average wall-bounces per paddle-bounce	Average serving time per point
Competitive $\rho = 1$	7.15 ± 1.01	0.87 ± 0.08	113.87 ± 40.30
Transition $\rho = 0.75$	7.58 ± 0.71	0.83 ± 0.06	129.03 ± 38.81
Transition $\rho = 0.5$	6.93 ± 0.49	0.64 ± 0.03	147.69 ± 41.02
Transition $\rho = 0.25$	4.49 ± 0.43	1.11 ± 0.07	275.90 ± 38.69
Transition $\rho = 0$	4.31 ± 0.25	0.78 ± 0.05	407.64 ± 100.79
Transition $\rho = -0.25$	5.21 ± 0.36	0.60 ± 0.05	449.18 ± 99.53
Transition $\rho = -0.5$	6.20 ± 0.20	0.38 ± 0.04	433.39 ± 98.77
Transition $\rho = -0.75$	409.50 ± 535.24	0.02 ± 0.01	591.62 ± 302.15
Cooperative $\rho = -1$	654.66 ± 542.67	0.01 ± 0.00	393.34 ± 138.63

Table 4: Behavioural statistics of the agents as a function of their incentive to score.

3 Discussion

Multiagent reinforcement learning has an extensive literature in the emergence of conflict and cooperation between agents sharing an environment [Tan93, CB98, BBDS08]. However, most of the reinforcement learning studies have been conducted in either simple grid worlds or with agents already equipped with abstract and high-level sensory perception.

In the present work we demonstrated that agents controlled by autonomous Deep Q-Networks are able to learn a two player video game such as *Pong* from raw sensory data. This result indicates that Deep Q-Networks can become a practical tool for the decentralized learning of multiagent systems living a complex environments.

In particular, we described how the agents learned and developed different strategies under different rewarding schemes, including full cooperative and full competitive tasks. The emergent behavior of the agents during such schemes was robust and consistent with their tasks. For example, under a cooperative rewarding scheme the two *Pong* agents (paddles) discovered the coordinated strategy of hitting the ball parallel to the x-axis, which allowed them to keep the ball bouncing between them for a large amount of time. It is also interesting to note that serving time, i.e. the time taken by the agent to launch the first ball in a game, was one of the behavioral variables modified during the learning.

3.1 Limitations

We observe that in the fully competitive (zero-sum) rewarding scheme, the agents overestimate their future discounted rewards. The realistic reward expectation of two equally skillful agents should be around zero, but in most game situations both of our Deep Q-Networks predict rewards near 0.5 (Figure 3, supplementary videos). Overestimation of Q-values is a known bias of the Q-learning algorithm and could potentially be remedied by using the novel Double Q-learning algorithm proposed by [Has10].

In addition, with a more accurate estimation of Q-values we might obtain clear collaborative behaviour already with higher values of ρ (less incentive to collaborate) and already earlier in the training process.

In this work we have used the simplest adaptation of deep Q-learning to the multiagent case, i.e., we let an autonomous Deep Q-Network to control each agent. In general, we expect that adapting a range of multiagent reinforcement algorithms to make use of Deep Q-Networks will improve our results and pave the way to new applications of distributed learning in highly complex environments.

A larger variety of metrics might have helped us to better describe the behaviour of different agents, but we were limited to the statistics extractable from the game memory. More descriptive statistics such as average speed of ball and how often the ball is hit with the side of the paddle would have required analyzing the screen images frame by frame. While probably useful descriptors of behaviour, we decided that they were not necessary for this preliminary characterization of multiagent deep reinforcement learning.

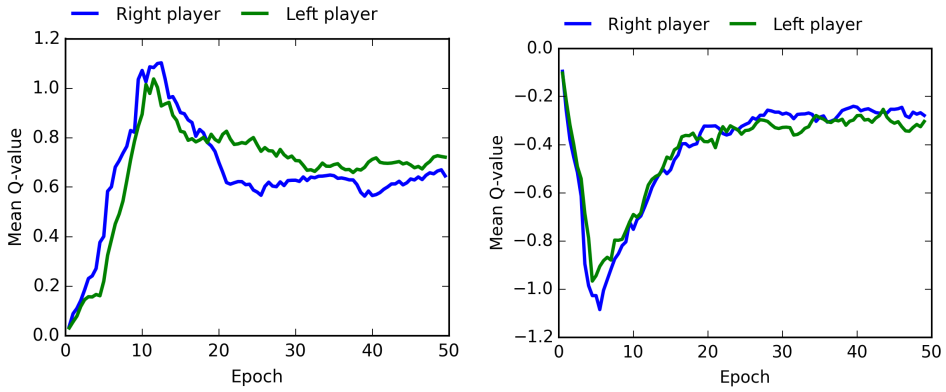
3.2 Future Work

In the present work we have considered two agents interacting in an environment such as *Pong* with different rewarding schemes leading them towards competition or collaboration. Using other games such as *Warlords* we plan to study how a larger number of agents (up to four) organize themselves to compete or cooperate and form alliances to maximize their rewards while using only raw sensory information. It would certainly be interesting to analyse systems with tens or hundreds of agents in such complex environments. This is currently not feasible with the system and algorithms used here.

A potential future application of the present approach is the study of how communication codes and consensus can emerge between interacting agents in complex environments without any a priori agreements, rules, or even high-level concepts of themselves and their environment.

4 Supplementary Materials

4.1 Evolution of Deep Q-Networks



(a) Q-value estimated by the competitive agents (b) Q-value estimated by the collaborative agents

Figure 8: Evolution of the Q-value and reward of cooperative and competitive agents over the training time.

4.2 Code

The version of the code adapted to the multiplayer paradigm together with the tools for the visualization can be accessed at our Github repository: `https://github.com/NeuroCSUT/DeepMind-Atari-Deep-Q-Learner-2Player`.

4.3 Videos

Please see the videos illustrating the core concepts and the results of the current work on the dedicated YouTube playlist `https://www.youtube.com/playlist?list=PLfLv_F3r0TwyaZPe50OOUx8tRf0HwdR_u`.

Author Contributions

A.T., T.M., K.K. and I.K. explored the background technical framework. D.K., A.T. and T.M. developed and tested the technical framework, and analyzed the data. A.T., T.M., D.K., I.K., Ja.A. and R.V. wrote the manuscript. All authors designed research, discussed the experiments, results, and implications, and commented on the manuscript. R.V. and Ja.A. managed the project. R.V. conceptualized the project.

Acknowledgements

We would like to thank Andres Laan and Taivo Pungas for fruitful early discussions about the project. We gratefully acknowledge the support of NVIDIA Corporation with the donation of one GeForce GTX TITAN X GPU used for this research. We also thank the financial support of the Estonian Research Council via the grant PUT438.

References

[BBDS08] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156–172, 2008.

[BNVB12] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *arXiv preprint arXiv:1207.4708*, 2012.

[CB98] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI/IAAI*, pages 746–752, 1998.

[Has10] Hado V Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[KCSG13] Jan Koutník, Giuseppe Cuccu, Jürgen Schmidhuber, and Faustino Gomez. Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 1061–1068. ACM, 2013.

[Lin93] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, DTIC Document, 1993.

[MA03] BW Mott and S Anthony. Stella: a multiplatform atari 2600 vcs emulator. 2003.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[PM10] David L Poole and Alan K Mackworth. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[SB98] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[Sch14] Howard M Schwartz. *Multi-Agent Machine Learning: A Reinforcement Approach*. John Wiley & Sons, 2014.

[SQAS15] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[Sum10] David JT Sumpter. *Collective animal behavior*. Princeton University Press, 2010.

[Tan93] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[Wat89] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.