DRQN, Multi-agent RL

Bio-Medical Computing Laboratory

TAEHEUM CHO

7th June, 2018

Multi-agent RL

RESEARCH ARTICLE

Multiagent cooperation and competition with deep reinforcement learning

Ardi Tampuu[1☯], Tambet Matiisen[1☯], Dorian Kodelja[1¤], Ilya Kuzovkin[1], Kristjan Korjus[1], Juhan Aru[2], Jaan Aru[1], Raul Vicente[1]*

1 Computational Neuroscience Lab, Institute of Computer Science, University of Tartu, Tartu, Estonia,
2 Department of Mathematics, ETH Zürich, Zürich, Switzerland

Multi-agent RL

- When two or more agents share an environment the problem of reinforcement learning is notoriously more complex.
- Indeed, most of game theory problems deal with multiple agents taking decisions to maximize their individual returns in a static environment.
- **Instead of a single agent playing** against a hardcoded algorithm, **we explore how multiple agents controlled** by autonomous DQNs learn to **cooperate and compete** while **sharing a high-dimensional environment.**
- We show that the agents develop successful strategies for both competition and cooperation, depending on the incentives provided by rewarding schemes.

Multi-agent RL - Materials and methods

- Q-learning algorithm
- Deep Q-learning algorithm
- Game selection
  - Pong game
- Rewarding schemes
  - We adjust the rewarding scheme by simply changing the reward ρ a player receives when putting the ball past the opponent (when scoring).

Table 1. Rewarding schemes to explore the transition from competitive to the cooperative strategy.

	L player scores	R player scores
L player reward	ρ	−1
R player reward	−1	ρ

For the cases we study $\rho \in [-1, 1]$. Example: with $\rho = -0.5$, when the left player scores, it receives −0.5 points and the right player receives -1 points.

Multi-agent RL - Materials and methods

- Training procedure
  - we let the agents learn for 50 epochs, 250000 time steps each.
  - the exploration rate decreases from an initial 1.0 to 0.05
  - Rewards are the signal that agents use to evaluate their performance and that they learn from.
- Collecting the game statistics
  - Average paddle-bounces per point
  - Average wall-bounces per paddle-bounce
  - Average serving time per point

Multi-agent RL - Results

Emergence of competitive agents

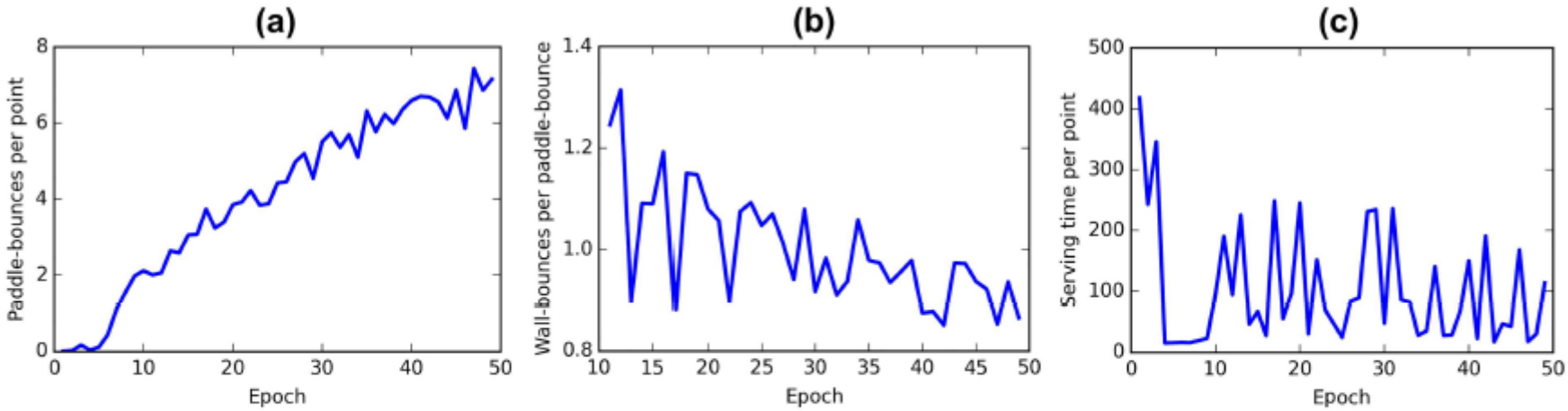

Fig 1. Evolution of the behavior of the competitive agents during training. (a) The number of paddle-bounces increases indicating that the players get better at catching the ball. (b) The frequency of the ball hitting the upper and lower walls decreases slowly with training. The first 10 epochs are omitted 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.

Multi-agent RL - Results

- Emergence of collaborative agents

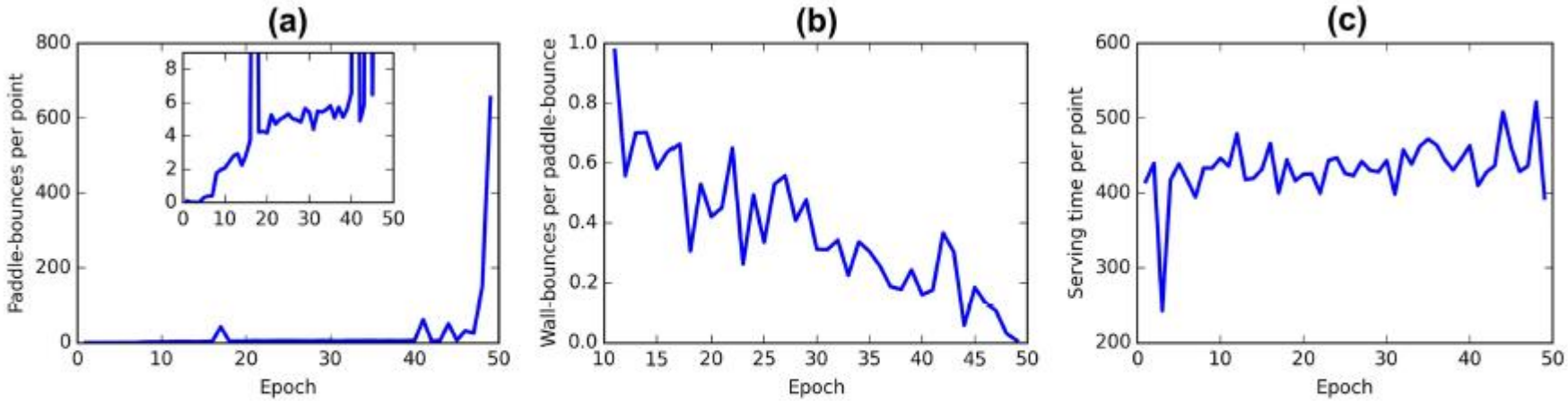

Fig 3. Evolution of the behavior of the collaborative agents during training. (a) The number of paddle-bounces increases as the players get better at reaching the ball. (b) The frequency of the ball hitting the upper and lower walls decreases significantly with training. The first 10 epochs are omitted from the plot as very few paddle-bounces were made by the agents and the metric was very noisy. (c) Serving takes a long time—the agents learn to postpone putting the ball into play.

Multi-agent RL - Results

- Progression from competition to cooperation

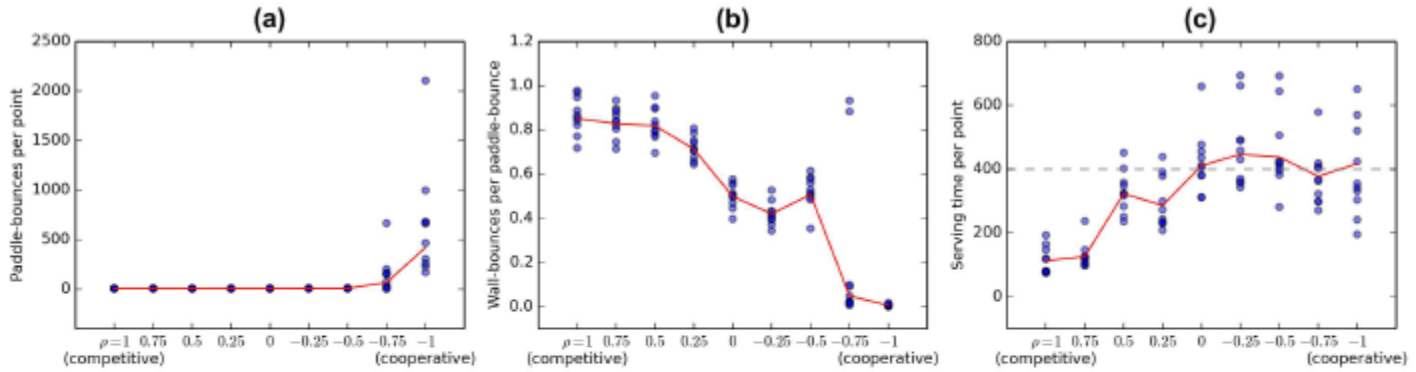

Fig 5. Progression of behavioral statistics when passing from competitive to collaborative rewarding scheme. Each blue dot corresponds to the average of one game. Red line depicts the average across games (also given in S2 Table). (a) The game lasts longer when the agents have a strong incentive to collaborate. (b) Forcing the agents to collaborate decreases the proportion of angled shots that bounce off the walls before reaching the opposite player. Notice the two aberrant values for $\rho = -0.75$ correspond to games where the agents never reach the collaborative strategy of keeping the ball alive by passing it horizontally. (c) Serving time decreases when agents receive stronger positive rewards for scoring.

Multi-agent RL - Results

- Comparison of multiplayer and single-player modes

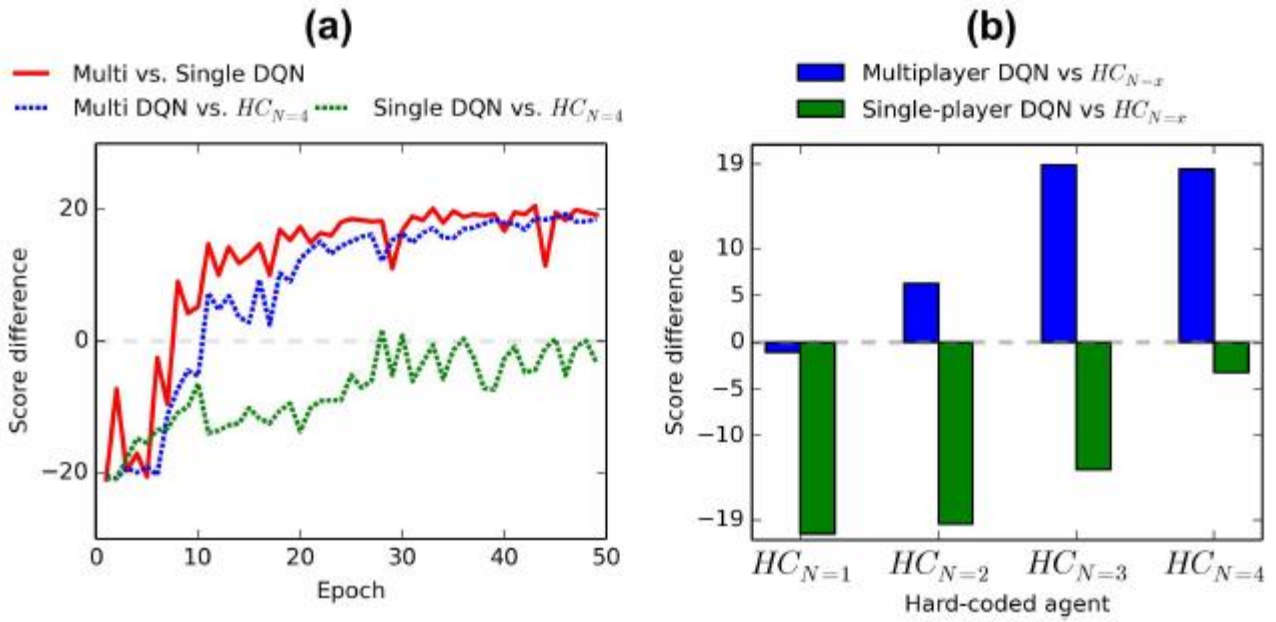

Fig 6. Results of games between multiplayer DQN, single-player DQN and four hand-coded algorithms. The values correspond to an average of 10 games with different random seeds. Score difference means the points scored by the agent mentioned first minus the points of the agent mentioned second. (a) Multi and Single DQN's performance against each other and against $HC_{N=4}$ in function of training time. (b) Scores of Single DQN and Multi DQN agents against 4 versions of a handcoded agent trying to keep the center of the paddle level with the ball. N refers to the number of frames a selected action is repeated by the algorithm before selecting a new action (the smaller the better).

DRQN

Deep Recurrent Q-Learning for Partially Observable MDPs

Matthew Hausknecht and Peter Stone
Department of Computer Science
The University of Texas at Austin
{mhauskn, pstone}@cs.utexas.edu

DRQN - Introduction

- DQNs learn to estimate the Q-Values (or long-term discounted returns) of selecting each possible action from the current game state.
- Instead of a Markov Decision Process (MDP), the game becomes a Partially-Observable Markov Decision Process (POMDP).
- Real-world tasks often feature incomplete and noisy state information resulting from partial observability.
- Therefore we introduce the Deep Recurrent Q-Network (DRQN), a combination of a Long Short Term Memory (LSTM) and a Deep Q-Network.
- We demonstrate that DRQN is capable of handing partial observability.
- When trained with full observations and evaluated with partial observations, DRQN better handles the loss of information than does DQN.

DRQN - Introduction

- Deep Q-Learning
- Partial Observability
  - In real world environments it's rare that the full state of the system can be provided to the agent or even determined.
  - A Partially Observable Markov Decision Process (POMDP) better captures the dynamics of many real world environments.
  - Our experiments show that adding recurrency to Deep Q-learning allows the Q-network network to better estimate the underlying system state

DRQN – DRQN Architecture

- we minimally modify the architecture of DQN, replacing only its first fully connected layer with a recurrent LSTM layer of the same size.

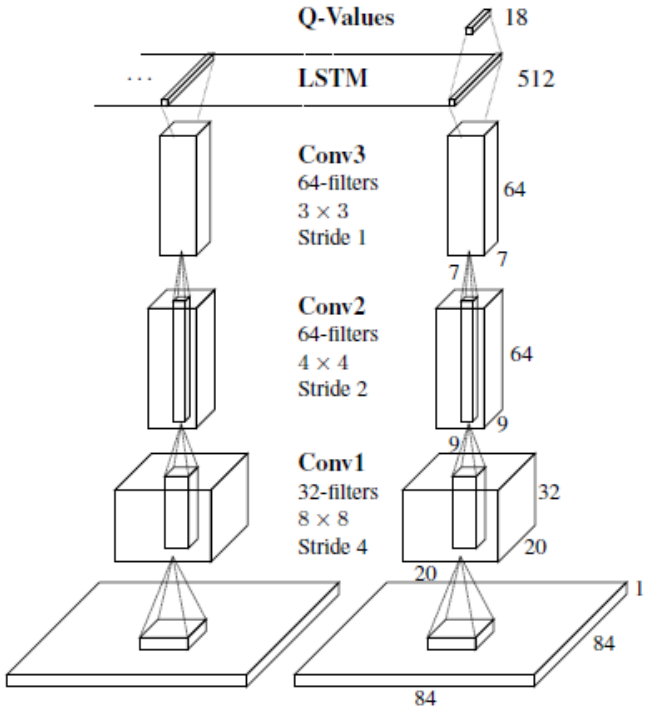

Figure 2: DRQN convolves three times over a single-channel image of the game screen. The resulting activations are processed through time by an LSTM layer. The last two timesteps are shown here. LSTM outputs become Q-Values after passing through a fully-connected layer. Convolutional filters are depicted by rectangular sub-boxes with pointed tops.

DRQN – Stable Recurrent Updates

- Bootstrapped Sequential Updates:
  - **Episodes are selected randomly** from the replay memory and updates begin at the **beginning of the episode** and proceed forward through time to the conclusion of the episode.
- Bootstrapped Random Updates:
  - Episodes are selected randomly from the replay memory and updates begin at random points in the episode and proceed for only unroll iterations timesteps
- Sequential updates have the advantage of carrying the LSTM's hidden state forward from the beginning of the episode.
- However, by sampling experiences sequentially for a full episode, they violate DQN's random sampling policy.

DRQN – Stable Recurrent Updates

- Experiments indicate that both types of updates are viable and yield convergent policies with similar performance across a set of games.
- All results herein use the randomized update strategy.
- We expect that all presented results would generalize to the case of sequential updates.

DRQN – Flickering Atari Games

- The screen is either fully revealed or fully obscured with probability p = 0.5
- In order to succeed at the game of Flickering Pong, it is necessary to integrate information across frames to estimate relevant variables such as the location and velocity of the ball and the location of the paddle.
- Perhaps the most important opportunity presented by a history of game screens is the ability to convolutionally detect object velocity.
- DRQN performs well at this task even when given only one input frame per timestep.
- Instead, the higher-level recurrent layer must compensate for both the flickering game screen and the lack of convolutional velocity detection.
- Thus both the non-recurrent 10-frame DQN and the recurrent 1-frame DRQN(for the last ten timesteps) have access to the same history of game screens

DRQN – Evaluation on Standard Atari Games

- Given the last four frames of input, all of these games are MDPs rather than POMDPs. Thus there is no reason to expect DRQN to outperform DQN.
- Indeed, results in **Table 1** indicate that on average, DRQN does roughly as well DQN.
- Specifically, our re-implementation of DQN performs similarly to the original.

	DRQN $\pm std$	DQN $\pm std$	
Game		Ours	Mnih et al.
Asteroids	1020 (±312)	1070 (±345)	1629 (±542)
Beam Rider	3269 (±1167)	**6923** (±1027)	6846 (±1619)
Bowling	62 (±5.9)	72 (±11)	42 (±88)
Centipede	3534 (±1601)	3653 (±1903)	8309 (±5237)
Chopper Cmd	2070 (±875)	1460 (±976)	6687 (±2916)
Double Dunk	**-2** (±7.8)	-10 (±3.5)	-18.1 (±2.6)
Frostbite	**2875** (±535)	519 (±363)	328.3 (±250.5)
Ice Hockey	-4.4 (±1.6)	-3.5 (±3.5)	-1.6 (±2.5)
Ms. Pacman	2048 (±653)	2363 (±735)	2311 (±525)

Table 1: On standard Atari games, DRQN performance parallels DQN, excelling in the games of Frostbite and Double Dunk, but struggling on Beam Rider. Bolded font indicates statistical significance between DRQN and our DQN.[5]

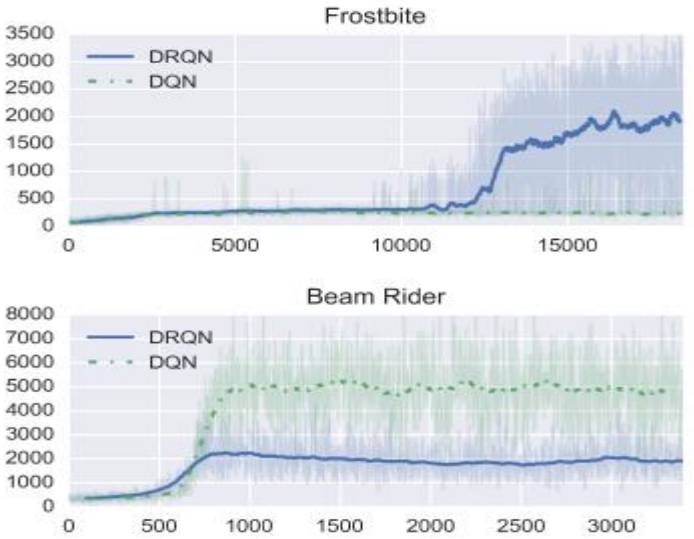

Figure 4: Frostbite and Beam Rider represent the best and worst games for DRQN. Frostbite performance jumps as the agent learns to reliably complete the first level.

DRQN – MDP to POMDP Generalization

- Figure 5 shows that while both algorithms incur significant performance decreases on account of the missing information, DRQN captures more of its previous performance than DQN across all levels of flickering.

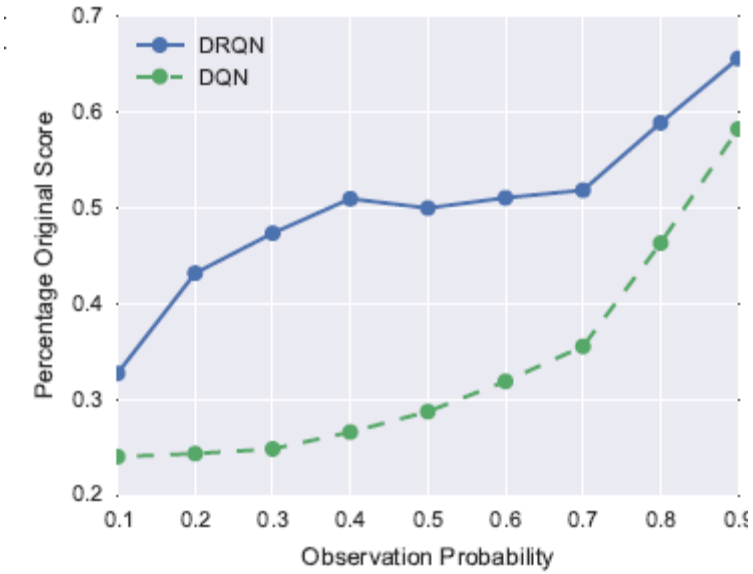

Figure 5: When trained on normal games (MDPs) and then evaluated on flickering games (POMDPs), DRQN's performance degrades more gracefully than DQN's. Each data point shows the average percentage of the original game score over all 9 games in Table 1.

DRQN – Discussion and Conclusion

- Real-world tasks often feature incomplete and noisy state information, resulting from partial observability.
- The resulting Deep Recurrent Q-Network (DRQN), despite seeing only a single frame at each step, is still capable integrating information across frames to detect relevant information such as velocity of on-screen objects.
- DRQN is better equipped than a standard Deep Q-Network to handle the type of partial observability induced by flickering game screens.
- **DRQN's performance generalizes better than DQN's at all levels of partial information.**
- across non-flickering Atari games, there are few significant differences between the recurrent and non-recurrent player.