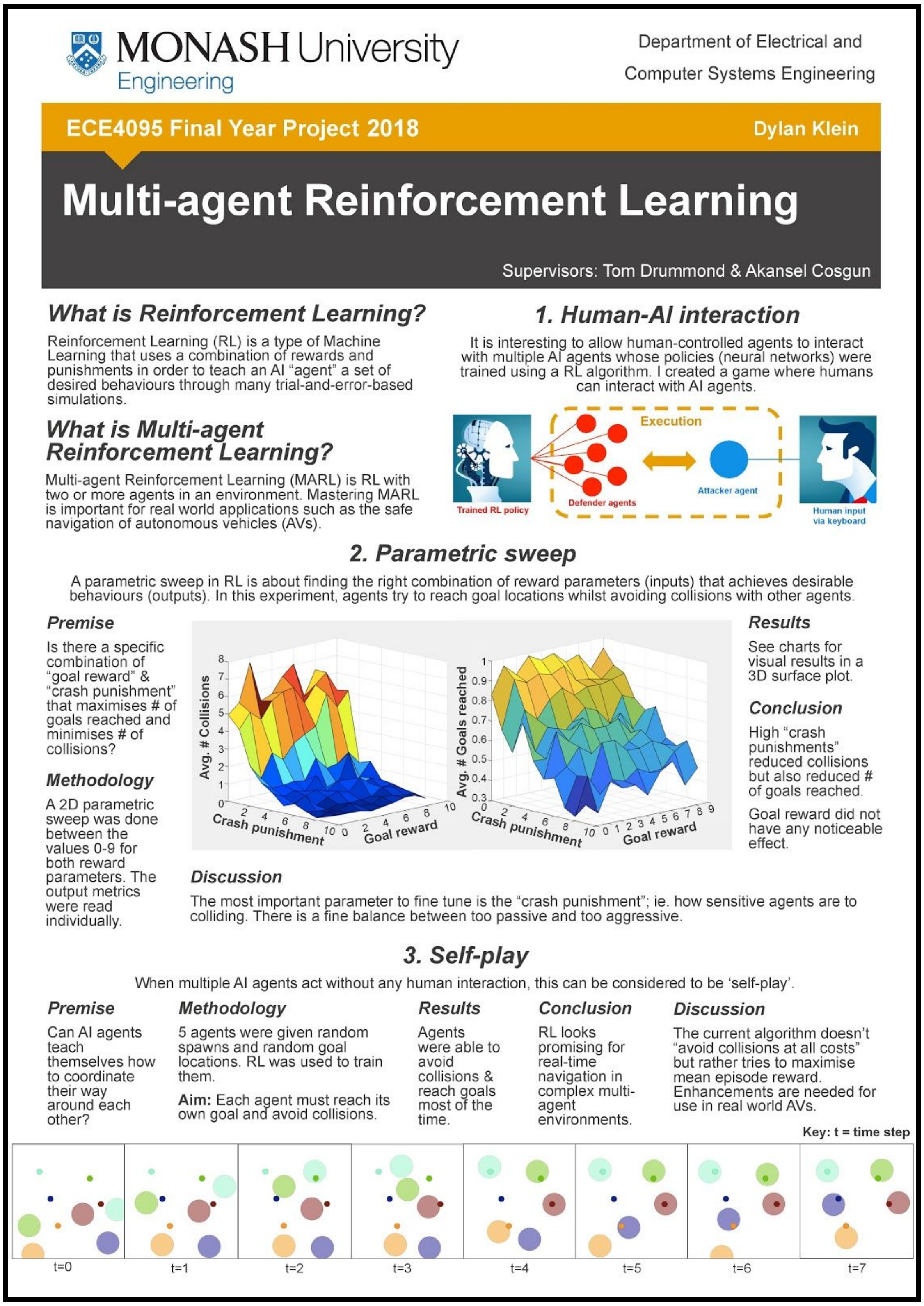
FYP 2018 Final Report
“Multi-agent Reinforcement Learning”

Dylan Klein
25117033

1. Significant Contributions

- Demonstrated that Q-table based reinforcement-learning can, when the state space is relatively small (less than ~10,000 possible states) deliver reliable negotiation behaviour.
- Discovered that the level of aggressiveness or defensiveness exhibited by agents can be determined by fine tuning the “collision punishment” in the reward function.
- Discovered that the ‘discount factor’ in the Bellman Equation is responsible for causing ‘catastrophic forgetting’ if set too low.
- Demonstrated that, in adversarial games, agents whose adversary is a policy-trained agent display qualitatively similar behaviour to that of agents whose adversary is human-controlled.
- Developed an interactive game where a human can interact with agents that were previously trained on a policy using reinforcement learning.

2. Poster



MONASH University
Engineering

Department of Electrical and Computer Systems Engineering

ECE4095 Final Year Project 2018 — **Dylan Klein**

Multi-agent Reinforcement Learning

Supervisors: Tom Drummond & Akansel Cosgun

What is Reinforcement Learning?

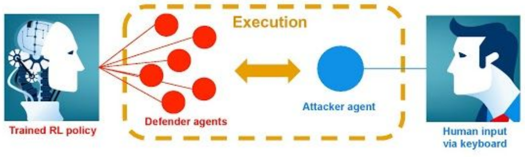
Reinforcement Learning (RL) is a type of Machine Learning that uses a combination of rewards and punishments in order to teach an AI "agent" a set of desired behaviours through many trial-and-error-based simulations.

What is Multi-agent Reinforcement Learning?

Multi-agent Reinforcement Learning (MARL) is RL with two or more agents in an environment. Mastering MARL is important for real world applications such as the safe navigation of autonomous vehicles (AVs).

1. Human-AI interaction

It is interesting to allow human-controlled agents to interact with multiple AI agents whose policies (neural networks) were trained using a RL algorithm. I created a game where humans can interact with AI agents.



2. Parametric sweep

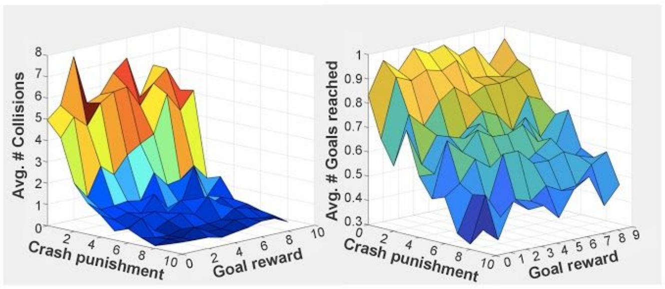
A parametric sweep in RL is about finding the right combination of reward parameters (inputs) that achieves desirable behaviours (outputs). In this experiment, agents try to reach goal locations whilst avoiding collisions with other agents.

Premise

Is there a specific combination of "goal reward" & "crash punishment" that maximises # of goals reached and minimises # of collisions?

Methodology

A 2D parametric sweep was done between the values 0-9 for both reward parameters. The output metrics were read individually.



Results

See charts for visual results in a 3D surface plot.

Conclusion

High "crash punishments" reduced collisions but also reduced # of goals reached.

Goal reward did not have any noticeable effect.

Discussion

The most important parameter to fine tune is the "crash punishment"; ie. how sensitive agents are to colliding. There is a fine balance between too passive and too aggressive.

3. Self-play

When multiple AI agents act without any human interaction, this can be considered to be 'self-play'.

Premise

Can AI agents teach themselves how to coordinate their way around each other?

Methodology

5 agents were given random spawns and random goal locations. RL was used to train them.

Aim: Each agent must reach its own goal and avoid collisions.

Results

Agents were able to avoid collisions & reach goals most of the time.

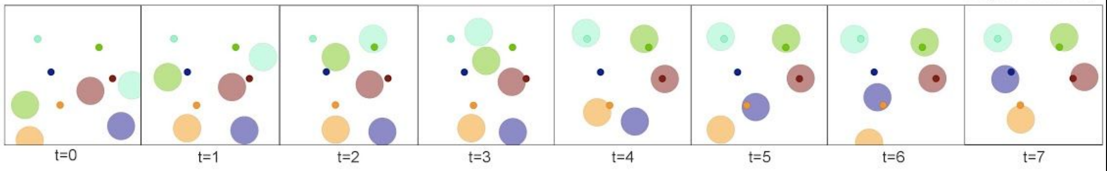
Conclusion

RL looks promising for real-time navigation in complex multi-agent environments.

Discussion

The current algorithm doesn't "avoid collisions at all costs" but rather tries to maximise mean episode reward. Enhancements are needed for use in real world AVs.

Key: t = time step



3. Executive Summary

My project investigated the use of Reinforcement Learning (RL) in a multi-agent context where agents need to cooperate or compete with each other. Multi-agent Reinforcement Learning (MARL) for Autonomous Vehicles (AVs) is of particular interest to me, however MARL can be applied in many other domains as well. For this project, my approach has been to:

1. Implement Q-table learning in a multi-agent context by cloning and modifying a repository created by Siraj Raval*.
2. Implement Deep Q-Network (DQN) learning in a multi-agent context by cloning and modifying the two repositories created by Open AI**.

One of my objectives was to demonstrate that multiple agents could reach respective goals whilst avoiding collisions by negotiating with each other. A key finding is that the behaviour of agents can be reduced down to one tunable parameter, which dictates their level of defensiveness or aggressiveness. This has significant implications for the design of Autonomous Vehicles (AVs), whereby a balance between passiveness and assertiveness must be found. My second objective was to investigate the interaction between human-controlled agents with AI-trained agents. A key finding is that, in adversarial games, agents whose adversary is a policy-trained agent display qualitatively similar behaviour to that of agents whose adversary is human-controlled. This suggests that if an AV were to encounter an unpredictable human driver on the road, it should behave with similar behaviour to when it faced its AI counterparts in training simulations.

*Siraj Raval's repository:
https://github.com/llSourcell/q_learning_demo
**Open AI's repositories:
1. https://github.com/openai/maddpg
2. https://github.com/openai/multiagent-particle-envs

4. Nomenclature

I would like to acknowledge Tom Drummond and Akansel Cosgun for their ongoing support and guidance throughout this project.

4.1. Table of Contents

4.2. List of Figures

5. Literature Review

5.1. Abstract

Deep Reinforcement Learning has recently become popular. The field has attracted the interest of experts in the field of robotics and automation in order to solve problems that require real-time autonomous decision making where the solution may not be obvious. This is the case with the problem of AVs. This literature review evaluates the benefits and disadvantages of learning based methods (such as RL) and rules-based methods. RL for real-time decision-making is explored since this is required in real-time robotic control systems. A specific area of interest for this study is multi-agent domains. How RL functions in multi-agent domains is critical to the viability of using RL in the context of AVs for example, where many vehicles obeying their own policies need to interact with each other by virtue of occupying the same roads at the same time. Further, adversarial games, where multiple agents have different objectives, is interesting since it adds further complexity to an agent who needs to discover that not all other agents are similar to itself. This is especially important when dealing with an unpredictable human-controlled adversary. This scenario can be compared to a driving scenario where an AV needs to work with other human drivers on the road in order to reach an objective. Finally the topic of Imitation Learning (IL) is explored since AV researchers have shown interest in order to reduce the space of possible solutions.

5.2. Learning-based vs Rules-based methods

Rules-based methods are when agents' policies are based on a set of mathematical equations (rules). Rules-based methods differ in nature from learning-based methods where agents' policies are learnt implicitly via training data. In an environment similar to the one I am implementing in this study, Berg, J., Lin, M., & Manocha, D. [1] concluded that Reciprocal Velocity Obstacles, a rules-based method, was successful in achieving collision-free navigation for a large number of agents (250 agents). Although the navigation was very robust, the downside of rules-based methods is the lack of domain generalisation that is inherent to the demonstration. Domain generalisation is something that learning-based methods aim to achieve.

Furthermore, any policy where the dynamics of agents can be described by a system of ordinary differential equations (ODEs) or partial differential equations (PDEs) can too be considered a rules-based method [2]. Again, many agents (500 microscopic agents) have been shown to coordinate and navigate successfully, especially in the presence of obstacles in the environment. However the disadvantage of such rules-based methods is that these environments need to be predictable. Such is not the case in real world applications, such as autonomous driving, where drivers have high unpredictability and the optimal solution is not obvious. This point further supports the implementation of learning-based policies (such as reinforcement learning or imitation learning) for such tasks.

Rules-based methods such as auction-based methods have proven successful candidates for multi-agent coordination tasks and routing [3]. In the context of robot teams where the team has common objectives, robots are required to dynamically delegate tasks to one another without the explicit definition of a leader agent. In this scenario, an auction system means that each robot places bids on the ability to take up certain tasks. This is where a type of negotiation is required to take place. There are a variety of methods to structure the bidding system, each with its own unique advantages and disadvantages. However these rules, such as with previous studies, need to be explicitly defined. Whereas a learning-based method need not be explicitly defined, leading to greater scalability. For the above reasons, I have decided to investigate the viability of Reinforcement Learning in multi-agent coordination, navigation and adversarial games.

5.3. Decision making using Reinforcement Learning

Reinforcement Learning (RL) is a type of learning that uses trial-and-error in order to train a policy for action-based environments. The rewards are compared to labels in classification problems. In RL, the rewards are said to be labels that are "time-delayed". A simple example of the promise of Reinforcement Learning is provided by Open AI's Gym project. Using a Markov Decision Process, Open AI demonstrates that an agent is able to navigate simple environments such as the Frozen Lake [4] using only the current and the most recent previous state. The agent's policy is stored in a Q-table which, to put it simply, is a matrix whose rows represent the entire state space of the game and whose columns represent all the available actions.

RL lends itself well to training agents to play video games. This is because video games are limited in their action space as well as in their state space. Specifically, Atari 2600 games are well defined in this regard. For this reason, Deepmind [5] focussed on testing the ability of Deep-Q Network (DQN) to play a set of 49 Atari 2600 games. Deepmind used a convolutional neural network in order to process 84 x 84 x 4 images of the on-screen pixels. This was done in order to give the AI minimal prior knowledge, that is to restrict the inputs to visual data only and the number of actions available (but not their correspondences). The reinforcement-learning based artificial intelligence successfully managed to surpass the performance of all previous algorithms (such as Q-tables) and achieved a level comparable to that of a professional human games tester, using the same algorithm, network architecture and hyperparameters.

Sutton, R., et al. [6] made significant contributions to the field of Reinforcement Learning by developing a Policy Gradient Method with Function Approximation. The development proved for the first time that a version of policy iteration with arbitrary differentiable function approximation is convergent to a locally optimal policy. Shalev-Shwartz, S., Shammah, S., Shashua, A. [7] used policy gradients in a multi-agent context in order to prove that it can work without the Markovian assumption. Because policy gradients is cited as a successful algorithm in multi-agent environments, I have an interest in using policy gradients in my study which focuses on multi-agent environments. Furthermore, Open AI's baseline Multi-agent Deep Deterministic Policy Gradients (MADDPG) [8] is an Actor-Critic method which is based on a policy gradient method. It was shown to perform well in multi-agent particle environments and this is the reason I have decided to use the MADDPG algorithm in my work.

5.4. Multi-agent Reinforcement Learning

Multi-agent environments, by their nature, introduce a layer of unpredictability over single-agent environments. A multi-agent environment is considered unpredictable when each agent is unaware of the policies of its counterpart agents, whether those counterparts be either AI or human. In a 2017 study, agents were trained under partial observability in a decentralised manner and their Deep Recurrent Q-Networks (DRQNs) were distilled (merged) after the training [9]. Overall, the merged DRQN was found to have superior performance than each individual DRQN. This is different to centralised methods of training, where a single policy is continuously updated during training time.

In the use case of Reinforcement Learning based Autonomous Vehicles, the primary aim is for an AV to drive in a safe, yet human-like, manner on the road with inevitably unpredictable human drivers. In the problem of real life autonomous driving, an agent needs to take into consideration more than just the current and one most recent state. In order to think like a human, the autonomous agent requires to take into account things like the 'mood of other drivers' and other information that occurred further back in the past. Hence it is necessary to remove the assumption of Markovity from the RL algorithm [7]. Additionally, the algorithm that Mobileye (Intel) is interested in implementing in their AVs is based on Policy Gradients. Policy Gradients have appeared in multiple literatures in the topic of multi-agent Reinforcement Learning.

Lowe, R., et al. [8] at Open AI compared two approaches to multi-agent unpredictable environments. One approach is to apply a single-agent actor-critic method to a multi-agent environment, in a decentralised fashion, called Deep Deterministic Policy Gradients (DDPG). The second is to use centralised learning method in order to train a multi-agent actor-critic-based policy MADDPG. In MADDPG, a policy is augmented in training time with the policies of all agents in the environment. In testing time, the execution is decentralised. MADDPG was found to overall outperform DDPG. The study made use of a particle environment in order to focus on the performance of agents in a multi-agent environment, without confusing the results with any potential effects of image processing.

5.5. Adversarial Reinforcement Learning

Self-play in competitive multi-agent environments is not a new idea – it has already been explored in TD-gammon [10] and refined in AlphaGo [11] and Dota 2 [12]. In both cases, the resulting behavior was far more complex than the environment itself, and the self-play approach provided the agents with a perfectly tuned curriculum for each task. In another Open AI study [13], the concept of adversarial self-play was brought to other domains: specifically, in the domain of continuous control, where balance, dexterity, and manipulation are the key skills. Agents were trained in a distributed manner with a Policy Gradient algorithm.

Adversarial play in multi-agent reinforcement learning has been extended to the domain of control and robotics. Bowling, M. and Veloso, M. [14] developed and general-purpose, scalable algorithm (GraWOLF) to teach robots to play adversarial games. The algorithm was demonstrated to work both in simulation and in the real world. The adversarial task was such that a defender agent was required to block an attacker from reaching a goal destination. The findings displayed that the attacker agent was almost impossible to defend against. At best the defender could slow the attacker down. It would be interesting to investigate how the attacker would fare with multiple coordinated defenders, who have power in numbers.

Adversarial self-play has been tested in the domain of generalised domain tasks. For instance, Deepmind's AlphaGo algorithm that has been successful in defeating a world-class Go player was further generalised to defeat world-class champions in the games of chess and shogi (Japanese chess) [15]. Whilst the study focussed on board games, there is no reason this sort of adversarial intelligence could not be applied to a control or physical robot context. Moreover, perhaps this sort of generalisation could extend to cooperative tasks in addition to the adversarial tasks.

5.6. Imitation Learning

Imitation Learning is the idea is that a policy need not learn from scratch, but rather imitate a human's actions. Video games have served as an appropriate domain in order to test the viability Imitation Learning. The main advantage of Imitation Learning is that it reduces the search space of appropriate solutions [16]. This helps reduce training time in terms of computational time, however in reality it requires collecting many hours of human play which inevitably takes much longer. The study found that Imitation Learning was sufficient to train an artificial game character in a First Person Shooter (FPS) to a degree that outperformed bots that use finite state machines or other architectures. However, it was still unable to realise some more advanced and complex human behaviours and playing techniques. It would be interesting to see whether a mixed training would improve upon this (Imitation combined with Reinforcement Learning).

Remarkable results have been achieved when Imitation Learning is used to train a system in an end-to-end fashion, mapping visual data directly to output actions [17]. Without explicitly decomposing the problem of self-driving into human-defined systems such as lane detection, path planning and control, NVIDIA taught a car to drive in an end-to-end fashion. The visual input from a single camera was connected directly to a CNN which then outputs the desired angle of the steering wheel. The paper argues is that this type of learning will lead to better system performance than traditional self-driving engineering methods. However in Imitation Learning there exists a significant issue of robustness. If the vehicle comes across a scenario (image) that it has not experienced before in its dataset, it will likely act in an unpredictable manner. This is dangerous in a safe self-driving scenario where there are many unpredictable circumstances.

A particular area of interest is the harmonisation of Imitation Learning with Reinforcement Learning. One method that has been examined is the architectural integration of the two methods. Hamahata, K., et al [18] introduced an additional internal reward system, which is generated by the learner agent in order to achieve this. The integration was able to speed up the learning process. However, whether the method was able to produce a more theoretically effective policy than what could be achieved with either Reinforcement Learning or Imitation Learning separately was not clear. It would be interesting to investigate whether a more sequential approach would lead to a more effective policy. In other words, a policy is first trained via Imitation Learning, followed by further training via Reinforcement Learning.

This sequential approach was included in Mobileye's paper. Shalev-Shwartz, S., Shammah, S., Shashua, A. [7] observed that the RL algorithm can be initialised with Imitation Learning, and then updated using an iterative Policy Gradient approach without the Markovian assumption. The superior performance of a mixed training over regular RL training is of particular interest to my project.

6. Introduction

My project investigated the use of Reinforcement Learning in a multi-agent context. I am investigating this topic because multi-agent reinforcement learning is a developing area and a key area of interest in the field of AV research. Furthermore, I believe that the interaction between humans and RL agents is a rather understudied field of research.

NVIDIA has developed an algorithm based on Imitation Learning in order to train AVs. Mobileye has developed an Imitation + Reinforcement Learning-based algorithm for the purpose of safe autonomous driving, which has been demonstrated to achieve impressive results. Open AI has open-sourced an algorithm called Multi-Agent Deep Deterministic Policy Gradient (MADDPG) for use in multi-agent environments. Open AI has also evaluated multi-agent RL in adversarial play environments.

Whilst the agents that were investigated in this project are point particles, some of the learnings from this report can be applied to real world contexts. For example, in the specific case of Autonomous Vehicles, there are two primary objectives:

1. Reach a desired goal destination
2. Avoid collisions with obstacles or other vehicles

One of my objectives was to demonstrate that multiple agents could reach respective goals whilst avoiding collisions by negotiating with each other. My initial approach to the task has been to use Q-table learning (the most basic form of reinforcement-learning) in order to provide a MVP. Beyond the MVP, my approach was to fork and modify Open AI's Multi-agent Particle Environment (MPE) repository in order to solve the specific problem of interest.

My second objective was to investigate the interaction between human-controlled agents with AI-trained agents. In order to achieve this, I added interactive functionality to Open AI's MPE training script. This allowed for the combination of trained policies and keyboard inputs in the same environment. Several adversarial games were created in order to showcase the interactivity.

7. Overview

Two approaches were taken in this project. Initially, a tabular Q-Learning approach was taken. This approach worked well for very simple worlds with a relatively small state space. However in order to create environments with large state spaces, this approach breaks down since it requires a matrix (table) to be computed that contains as many rows as there exist possible states. Any matrix with over ~10,000 rows x 5 columns took a few minutes to initialise. With this approach, introducing multiple agents increased the state space exponentially. Each additional agent would raise the magnitude of the number of rows of the Q-table by an order of 10, thus potentially leading to matrices with millions or billions of rows which would take years to initialise!

For this reason, a DQN approach was then taken. The implementation of a DQN allowed for many agents to be introduced, and for more complex environments to be introduced. With DQN, a CPU could train 40+ agents simultaneously in a reasonable amount of time. This is because the network is not required to compute a row in a matrix for every possible state. Rather, the DQN attempts to learn a function which directly maps an observation to an action.

7.1. Tabular Q-Learning approach

Q-Learning methods that use tables as the policy are known as tabular Q-Learning [19]. Tabular Q-Learning attempts to learn the value of being in a given state, and taking a specific action there. In its simplest implementation, Q-Learning is a table of values for every state (row) and action (column) possible in the environment. Within each cell of the table, we learn a value for how good it is to take a given action within a given state. We make updates to our Q-table using something called the Bellman equation, which states that the expected long-term reward for a given action is equal to the immediate reward from the current action combined with the expected reward from the best future action taken at the following state. In this way, we reuse our own Q-table when estimating how to update our table for future actions. In equation form, the rule looks like this:

$$Eq\ 1.\ Q(s,a) = r + \gamma(max(Q(s',a'))$$

This says that the Q-value for a given state (s) and action (a) should represent the current reward (r) plus the maximum discounted (γ) future reward expected according to our own table for the next state (s') we would end up in. The discount variable allows us to decide how important the possible future rewards are compared to the present reward. By updating in this way, the table slowly begins to obtain accurate measures of the expected future reward for a given action in a given state.

7.2. Deep Q-Network approach

Open AI have developed an algorithm called Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [8]. MADDPG extends a reinforcement learning algorithm called DDPG, taking inspiration from actor-critic reinforcement learning techniques; other groups are exploring variations and parallel implementations of these ideas.

The algorithm treats each agent in the simulation as an “actor”, and each actor gets advice from a “critic” that helps the actor decide what actions to reinforce during training. Traditionally, the critic tries to predict the value (i.e. the reward we expect to get in the future) of an action in a particular state, which is used by the agent - the actor - to update its policy. This is more stable than directly using the reward, which can vary considerably. To make it feasible to train multiple agents that can act in a globally-coordinated way, critics are enhanced so they can access the observations and actions of all the agents, as the Figure 7-1 shows.

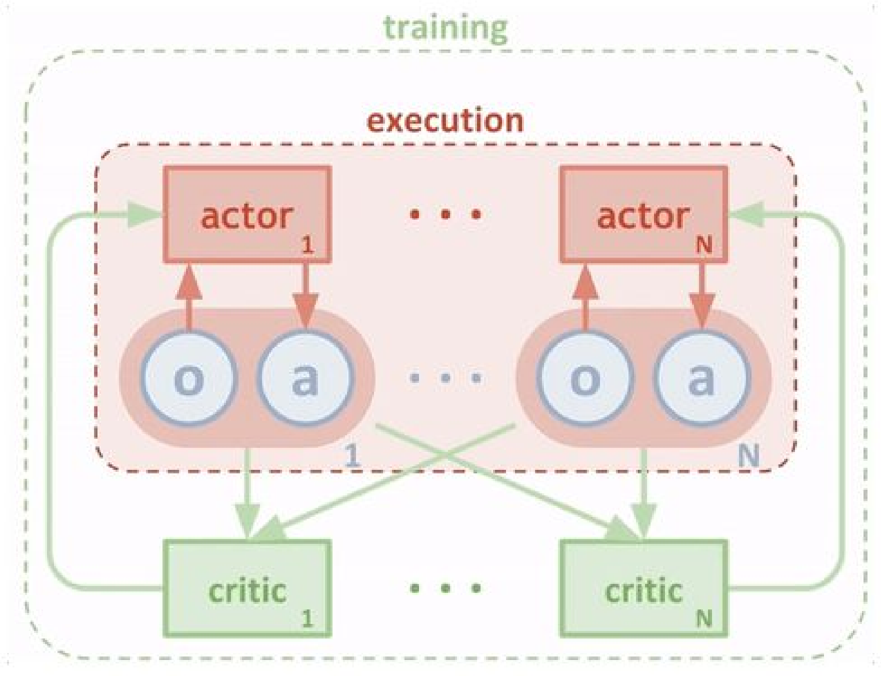


Figure 7-1: MADDPG algorithm architecture

Agents don’t need to access the central critic at test time; they act based on their observations in combination with their predictions of other agents behaviors’. Since a centralized critic is learned independently for each agent, this new approach can also be used to model arbitrary reward structures between agents, including adversarial cases where the rewards are opposing.

Several multi-agent particle environments were created in order to evaluate the above MADDPG algorithm's performance. The details of each environment is discussed in the following section.

8. Detailed Discussion

This section will discuss the particular details of the environments which were set up to be tested. Initially, environments were created to cater for a tabular Q-Learning approach. Later, environments were created for a DQN approach. The DQN experiments, which implement Open AI's MADDPG, include sections on self-play, a parametric sweep and human-AI interaction.

8.1. Tabular Q-Learning approach

The tests conducted with the tabular Q-Learning approach served as a proof-of-concept that agents, even with the most basic method of tabular Q-Learning, could exhibit negotiation and coordination behaviour. The code base from Siraj Raval's repository "Q Learning demo" was cloned and modified in order to allow for multiple agents in one environment. Tests were designed to mimic common driving scenarios. Tests that were conducted are described in Table 8-1 below. Actions available to agents are: Up, Down, Left, Right, No op. An important note to make is that the order in which agents took turns to act in any one step was randomised. This was done in order to remove any 'first-move' bias from occuring. Additionally, each agent took only one individual step in an environment step.

Learning parameters are:

- discount_factor=0.9
- alpha=0.1
- epsilon_decay=0.5

Reward function includes:

- Wall collision = -1
- Step cost = -0.04
- Goal reward = +5
- Agent-agent collision = -10

Table 8-1: Noteworthy Tabular Q-Learning Test Cases

Test ID	Description
a	T-intersection
b	Head-to-head cars on a narrow road
c	Roundabout
d	"Plus" (+) game

8.1.1. T-intersection

Premise	The most basic of environments is one where two agents contest one cell in order to reach a goal location. For a successful negotiation, one agent will need to stop and give way whilst the other agent takes way.
Methodology	Agent 1 spawns in the bottom left corner whilst agent 2 spawns in the bottom right. The bottom centre cell is contested. The top centre cell is the goal cell of both agents. Agents are not punished if they collide on the goal cell specifically, but are punished if they collide elsewhere.

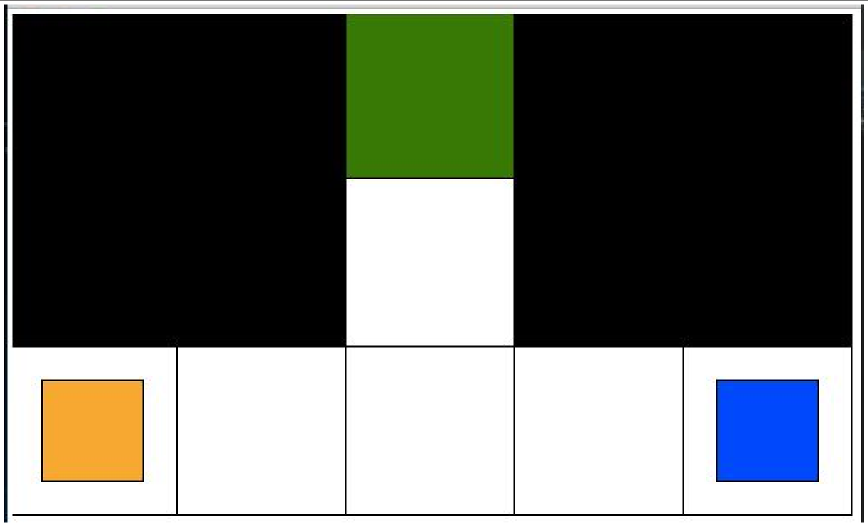


Figure 8-1: T-intersection environment

8.1.2. Head-to-head cars on a narrow road

Premise	In this scenario, a successful negotiation will require one agent to step aside in order to let the other agent pass, as if on a narrow road where only one vehicle can pass at any one time.
Methodology	Agent 1 spawns on the left, agent 2 spawns on the right. The centre bottom cell is contested. Each agent's' goal is located where its counterpart spawns.

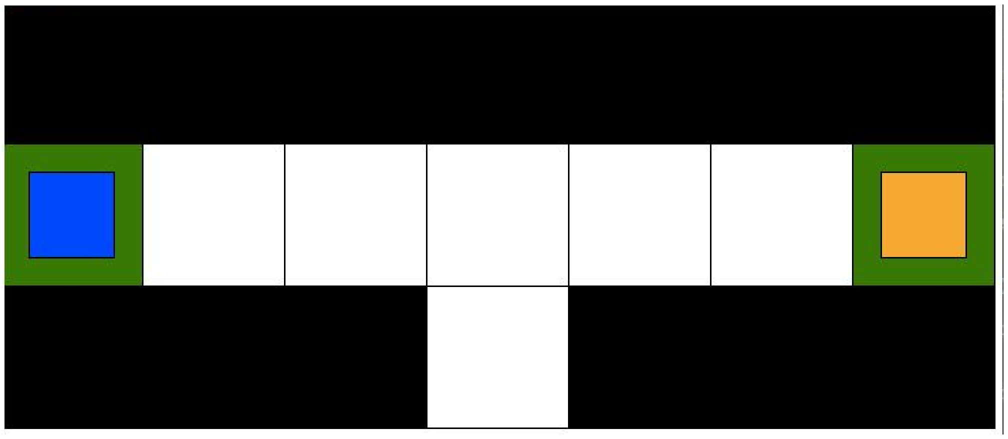


Figure 8-2: Head-to-head cars environment

8.1.3. Roundabout

Premise	In this scenario, there are four entries/exits. A successful negotiation will require all agents in the roundabout to move round the roundabout in the same direction.
Methodology	Between 2-4 agents are tested. Each agent spawns at its own cell either at the top, bottom, left, or right of the grid. Each agent's goal cell is located directly opposite from the cell at which they spawn.

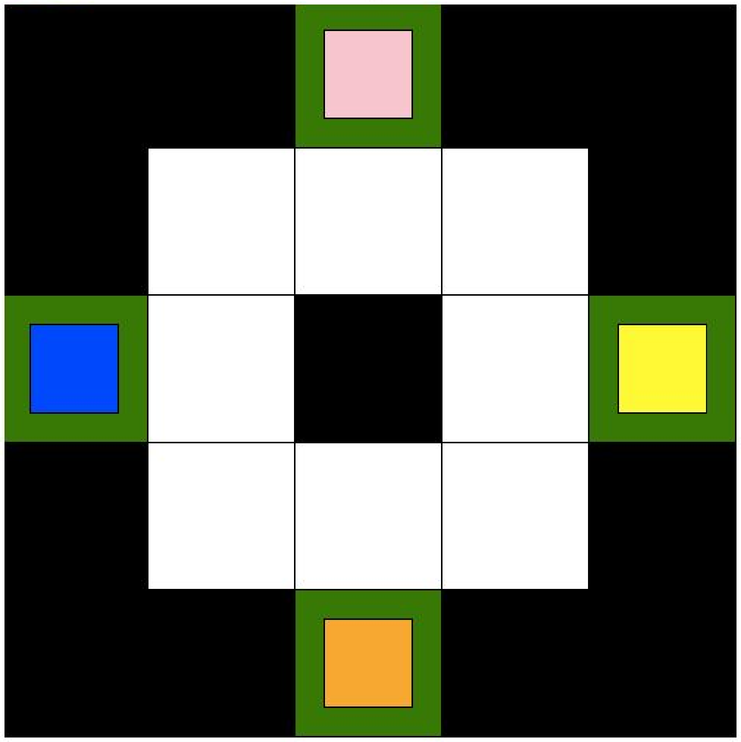


Figure 8-3: Roundabout environment

8.1.4. “Plus” (+) game

Premise	In this scenario, the goal destinations of each agent are constantly changing which causes randomised negotiations when the centre cell is contested.
Methodology	Between 2-3 agents are tested. Each agent spawns at its own cell either at the top, bottom, left, or right of the grid. Each agent’s goal cell is dynamically allocated. An agent can have multiple simultaneous goals. Each agent’s intended goals are displayed to the viewer in testing mode, however are hidden from all other agents.

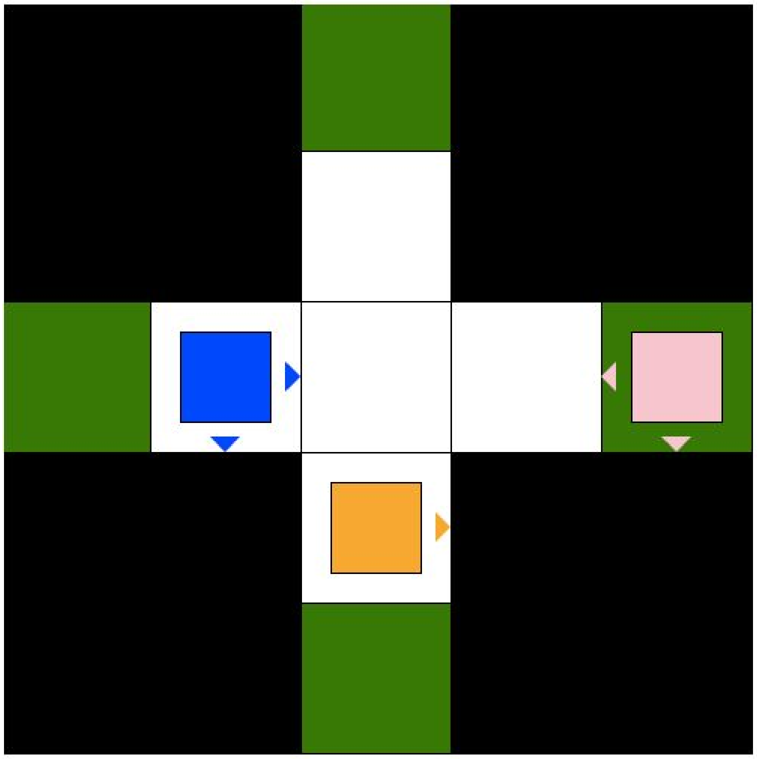


Figure 8-4: “Plus” environment

8.2. Deep Q-Network approach

Tests that were conducted are described in Table 8-2 below. The Test ID column refers to the Test Case spreadsheet (see Appendix A). Each test had a maximum episode length of 50 steps, with the exception of test 04, which had a maximum episode length of 25 steps. 25 steps allowed for faster training, however 50 steps was necessary to give agents enough time to find a reasonable solution in each episode.

8.2.1. Self-play

Self-play test cases involve agents being trained simultaneously, using the actor-critic-based MADDPG described above. Agents are tested as usual, by observing the mean-episode reward and by means of an "eye test", ie. looking at the visual playback which is rendered to screen.

Table 8-2: Noteworthy DQN Self-Play Test Cases

Test ID	Description
04	Simple point-to-point navigation (1 agent)
12	Simple point to point navigation with an obstacle (1 agent)
17	Navigation and collision avoidance (2 agents)
22	Navigation and collision avoidance (10 agents)
82	Navigation, collision avoidance and negotiation (2 agents)
93	Navigation, collision avoidance and negotiation (3 agents)
95	Navigation, collision avoidance and negotiation (4 agents)
96	Navigation, collision avoidance and negotiation (5 agents)

8.2.1.1. Simple point-to-point navigation (1 agent)

Premise	The simplest proof of concept was demonstrating that a single agent could navigate itself to a goal landmark, without any other agents or obstacles in the environment.
Methodology	The "Simple Spread" scenario from Open AI's MPE repo was modified. Number of agents was reduced to 1. Number of landmarks was reduced to 1. Initially the respective spawn locations of the agent and the goal were fixed. The spawn locations were later randomized in order to generalize the policy of the agent.

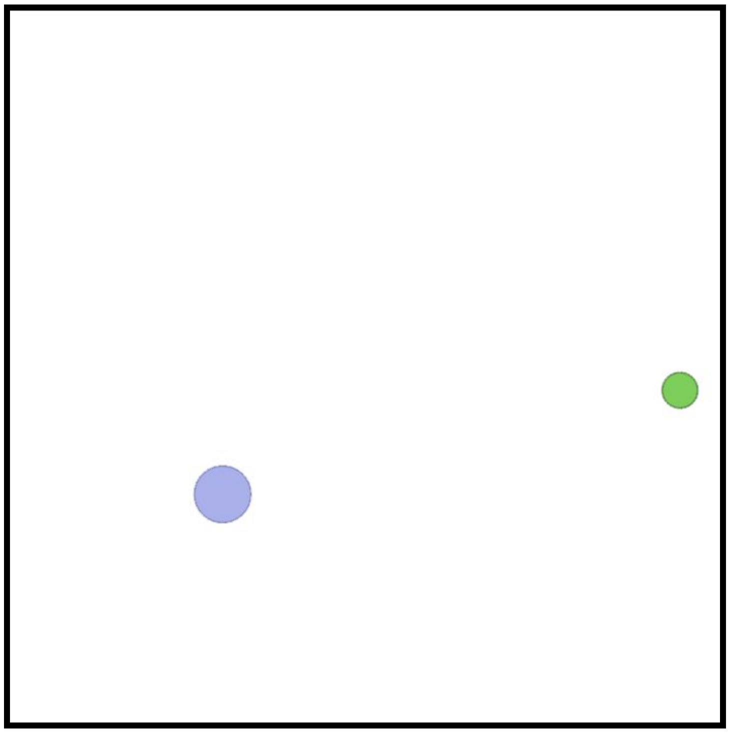


Figure 8-5

8.2.1.2. Simple point to point navigation with an obstacle (1 agent)

Premise	The aim is to demonstrate that an agent can navigate around the obstacle and continue onto its goal.
Methodology	The environment was modified from the previous test to include a stationary obstacle. Number of obstacles was set to 1. Number of agents remained at 1. Number of goals remained at 1. Initially the respective spawn locations of the agent and the goal were fixed. The spawn locations were later randomized in order to generalize the policy of the agent.

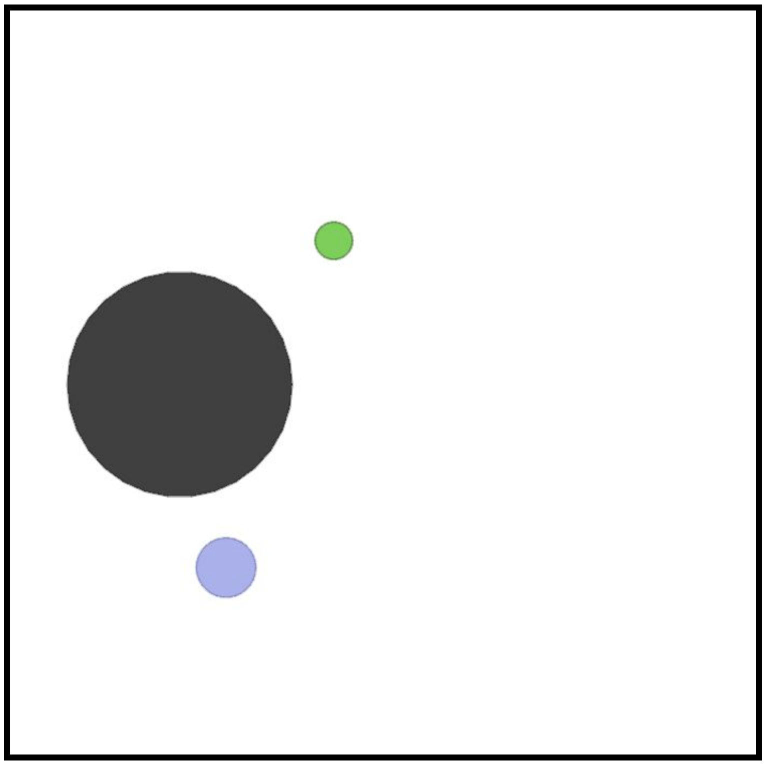


Figure 8-6

8.2.1.3. Navigation and collision avoidance (2 agents)

Premise	The aim is to demonstrate that 2 agents can avoid colliding with each other and continue onto their respective goals
Methodology	The environment was modified from the previous test to include a second agent. Number of obstacles was set to 0. Number of agents increased to 2. Number of goals increased to 2. A punishment was given when agents collide with each other. Initially the respective spawn locations of the agent and the goal were fixed. The spawn locations were later randomized in order to generalize the policy of the agent. Agents' reward functions were redefined to incentivise movement to an agent's unique goal using unique goal ID's (in one-hot encoding). This is different from Open AI's implementation where agents are incentivised to move to the closest goal.

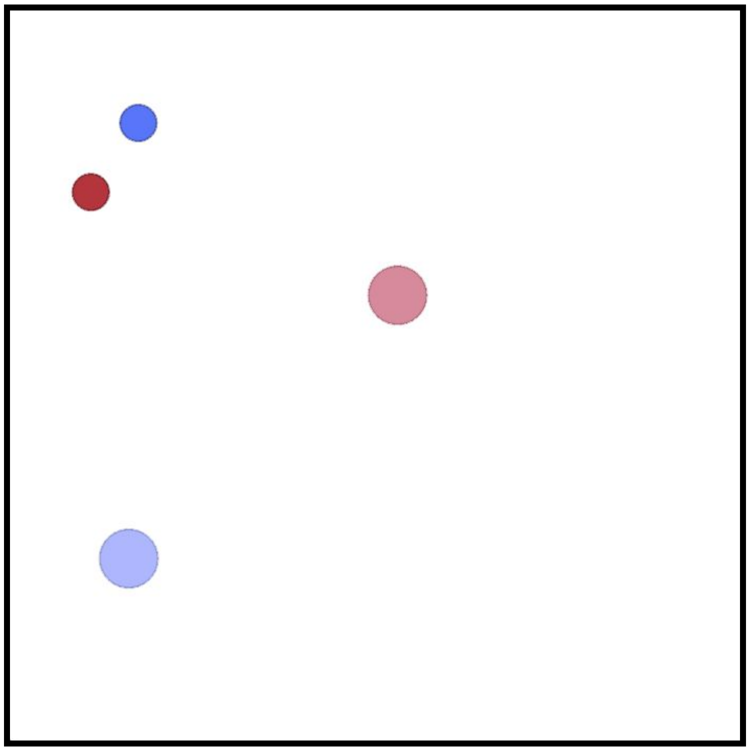


Figure 8-7

8.2.1.4. Navigation and collision avoidance (10 agents)

Premise	The aim is to demonstrate that many agents can avoid colliding with each other and continue onto their respective goals.
Methodology	The environment was modified from the previous test to include 10 agents. Number of goals increased to 10. Number of obstacles was set to 0.

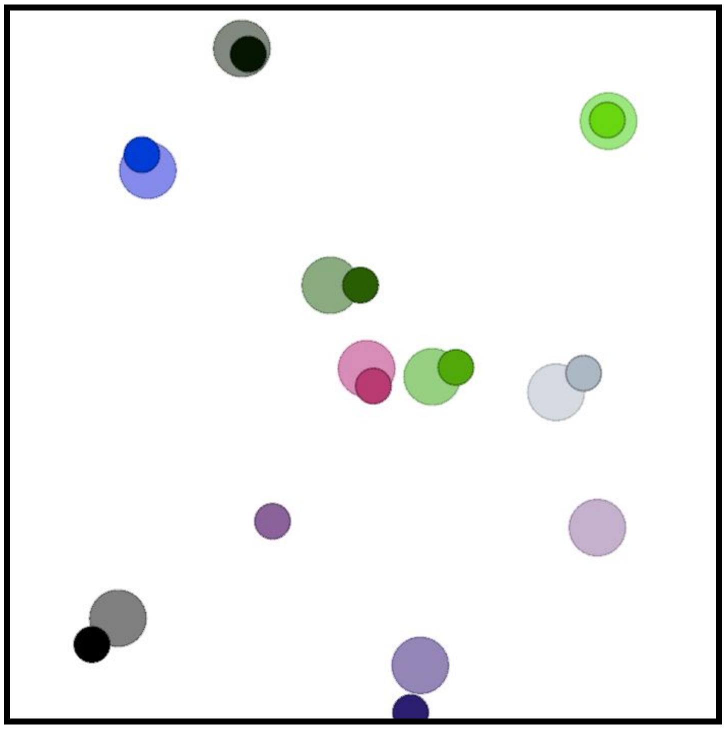


Figure 8-8

8.2.1.5. Navigation, collision avoidance and negotiation (2 agents)

Premise	The aim is to demonstrate that 2 agents can implicitly communicate via each other's movements in order to negotiate a contested space and avoid collision.
Methodology	A plus environment was created by introducing 12 static obstacles (see Figure 8-9). The agents and goals were given random spawn locations within the corridors of the plus structure.

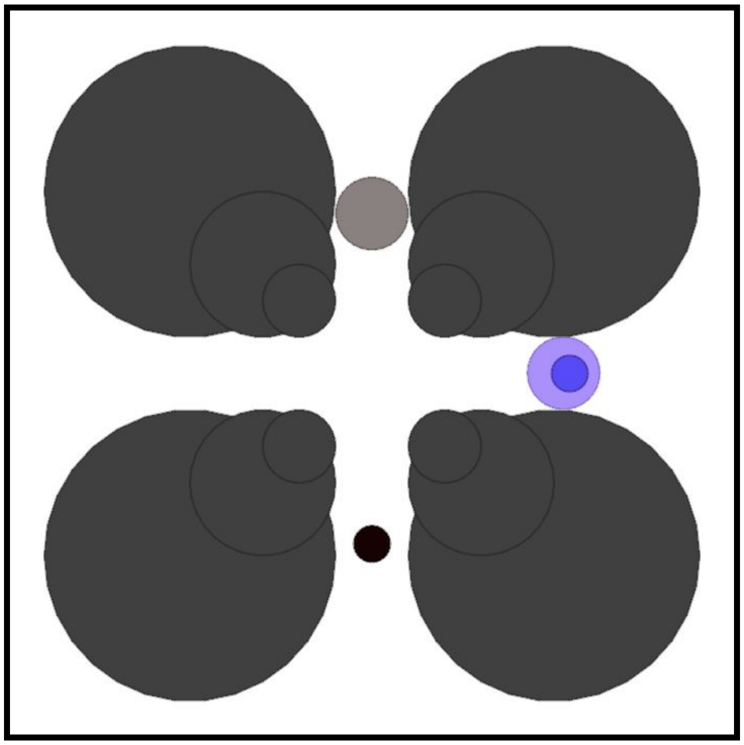


Figure 8-9

8.2.1.6. Navigation, collision avoidance and negotiation (3 agents)

Premise	The aim is to demonstrate that 3 agents can implicitly communicate via each other's movements in order to negotiate a contested space and avoid collision.
Methodology	The environment was modified from the previous test to include an additional agent and an additional goal.

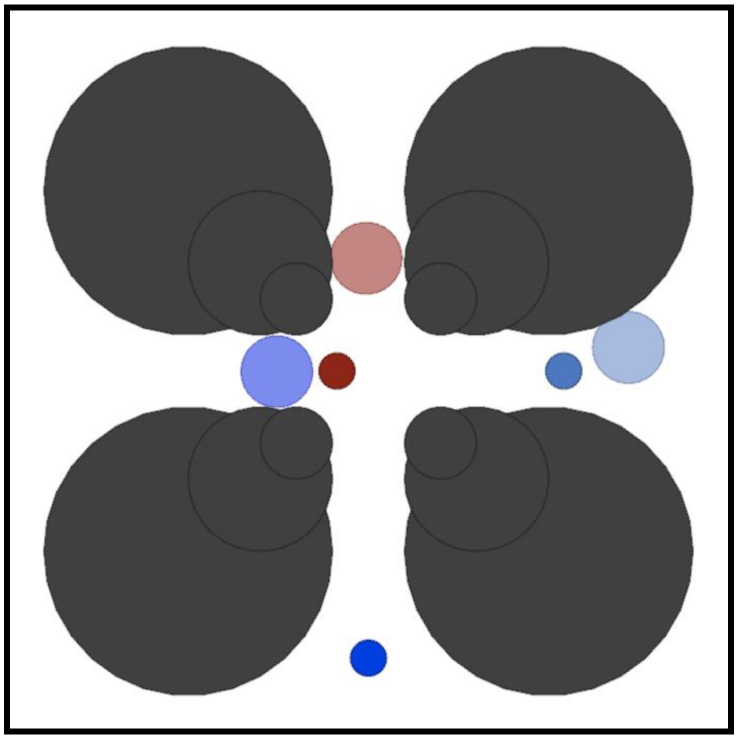


Figure 8-10

8.2.1.7. Navigation, collision avoidance and negotiation (4 agents)

Premise	The aim is to demonstrate that 4 agents can implicitly communicate via each other's movements in order to negotiate a contested space and avoid collision.
Methodology	The environment was modified from the previous test to include an additional agent and an additional goal.

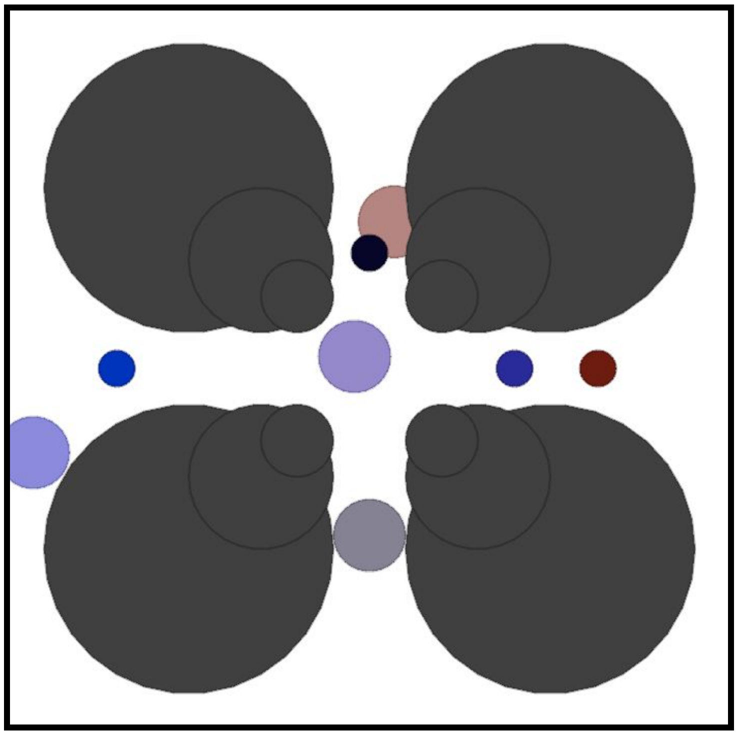


Figure 8-11

8.2.1.8. Navigation, collision avoidance and negotiation (5 agents)

Premise	The aim is to demonstrate that 5 agents can implicitly communicate via each other's movements in order to negotiate a contested space and avoid collision.
Methodology	The environment was modified from the previous test to include an additional agent and an additional goal.

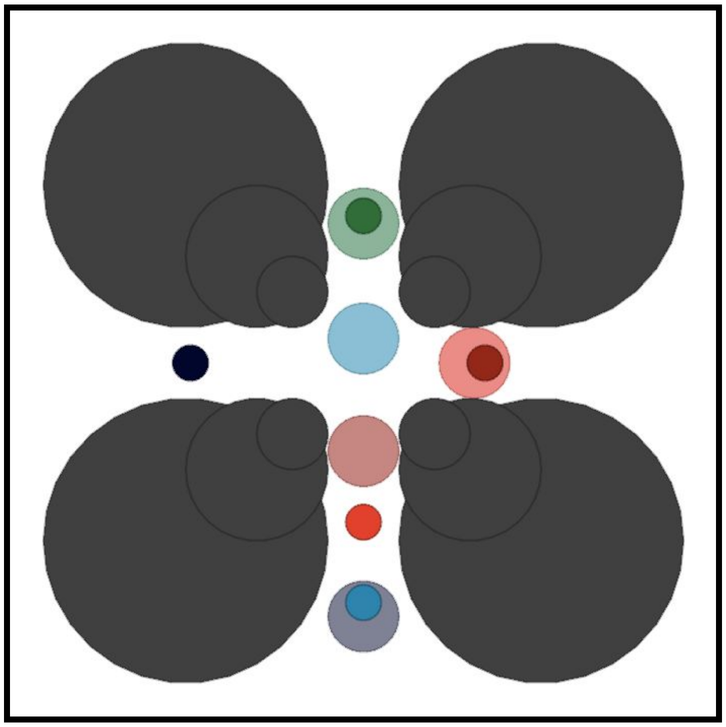


Figure 8-12

8.2.2. Parametric Sweep

A 2-dimensional parametric sweep on the same scenario ("plus world") as in section 8.2.1.6 (*Navigation, collision avoidance and negotiation - 3 agents)* was conducted in order to tease out the best parameters that would yield the most desired behaviour by agents. This scenario was chosen since agents' behaviours seemed to show promising results, however fine-tuning seemed that it may be able to offer additional benefits. It was hypothesised that there would be a fine balance between the following input parameters:

- Collision punishment: How badly agents are punished when two or more agents overlap any amount of space together.
- Cooperative reward: How well agents are rewarded when ALL agents reach their respective goals. To clarify, if any one agent has not reached its goal, no agents are rewarded. This setup was implemented to encourage agents to encourage cooperative behaviour.

Each parameter was swept between the values of 0-9. This range of values was chosen because the step cost (based on Euclidean distance to goal) was a value with magnitude less than sqrt(2) ≅ 1.414. The upper values of the range (5-9) would be high enough to significantly influence the reward function above the step cost. Whilst the lower values (0-4) would somewhat influence the reward function above the step cost. It was decided that the range 0-9 had enough resolution given the training time of 2 days.

The outputs of the experiment are the benchmarked metrics:

- Average number of collisions per episode per agent: This metric is an indicator of how many collisions have occurred between agents, adjusting for number of agents and number of episodes.
- Average number of goals reached per episode per agent (0-1): This metric is an indicator of how frequently agents reached their goal locations. For example, if 4 agents reached all 4 goals then this metric = 100% = 1 or if 3 out of 4 reached goals, this metric = 75% = 0.75.

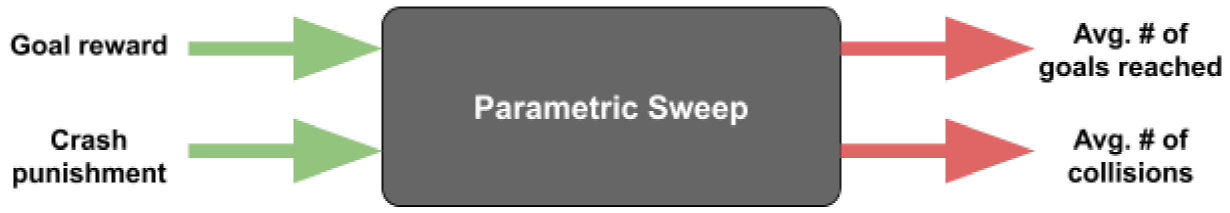


Figure 8-13

The desired behaviour would be an agent who could *minimise* its collisions whilst *maximising* the frequency in which it reaches its goal location. As a rough benchmark, the experiment would be deemed to be successful if a set of input parameters are found such that:

- Average number of collisions per episode per agent < 0.5
- Average number of goals reached per episode per agent > 0.9

8.2.3. Human-AI Interaction

In this section, all agents were trained simultaneously using the actor-critic-based method MADDPG mentioned above. In testing mode, the interactive.py script was modified in order to interface with trained policies. In this case, one agent's policies was overwritten by keyboard inputs (up, down, left, right) whilst the rest of the agents followed their learned policies.

8.2.3.1. "Rugby" game

Premise	The aim is to observe what behaviours emerge when a team of defender agents need to defend a goal location from an attacker agent. Another aim is to observe the difference in behaviours when the attacker agent is based on a learned policy compared with when it is based on human inputs.
Methodology	The environment consists of 6 defender agents (red) and a slightly larger attacker agent (blue). Each time the attacker reaches the goal, the location of the goal is reset to a random spawn.

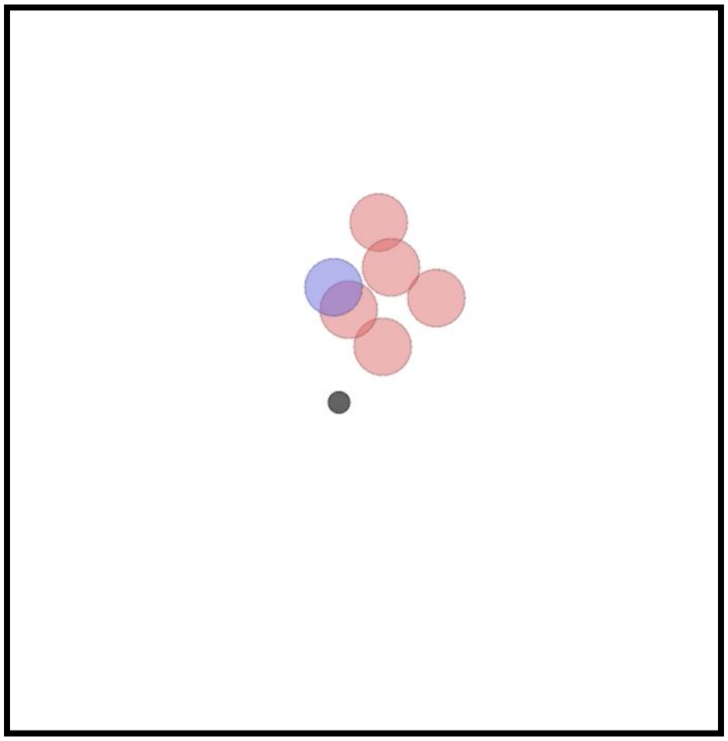


Figure 8-14

8.2.3.2. "FYP" structure

Premise	The aim is to observe whether a group of agents, who trained in an environment without any adversary, could continue to follow their objective function during testing with a human-controlled "destructor" adversary with unpredictable behaviours.
Methodology	The environment consists of 42 agents, 41 of which is designated to a specific location in order to collectively retain a specific structure. In this case, the structure spelled the acronym "FYP". The last agent was given no reward function in training, however in testing it is human-controlled. The human can experiment with crashing into the rest of the structure in order to disturb the structure. The agents are considered successful if they can recover to their designated locations

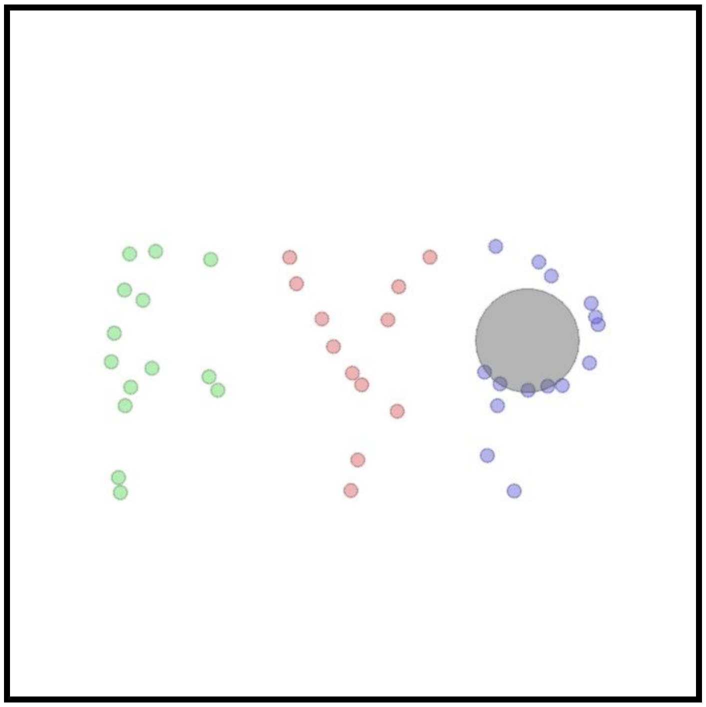


Figure 8-15

9. Experimental Results and Discussion

This section will discuss the results from experimenting with a tabular Q-Learning approach and with a DQN approach. The DQN experiments, which implement Open AI's MADDPG, include sections on self-play, a parametric sweep and human-AI interaction.

9.1. Tabular Q-Learning approach

9.1.1. T-intersection

Results	The agents' learning curves (Figure 9-1) suggest that whilst there is fight for dominance in the early episodes causes high volatility, eventually agents both learn to cooperate and hence simultaneously converge to a steady state. By way of an 'eye test', agents were observed to quickly flicker (oscillate) back and forth between cells whilst negotiating in order to communicate intent to the other agent. Figure 9-2 demonstrates that the number of collisions per episode falls at a rapid pace as both agents learn how to avoid each other.
Conclusion	Overall the agents were able to reliably negotiate in order to resolve the conflict of the contested cell.

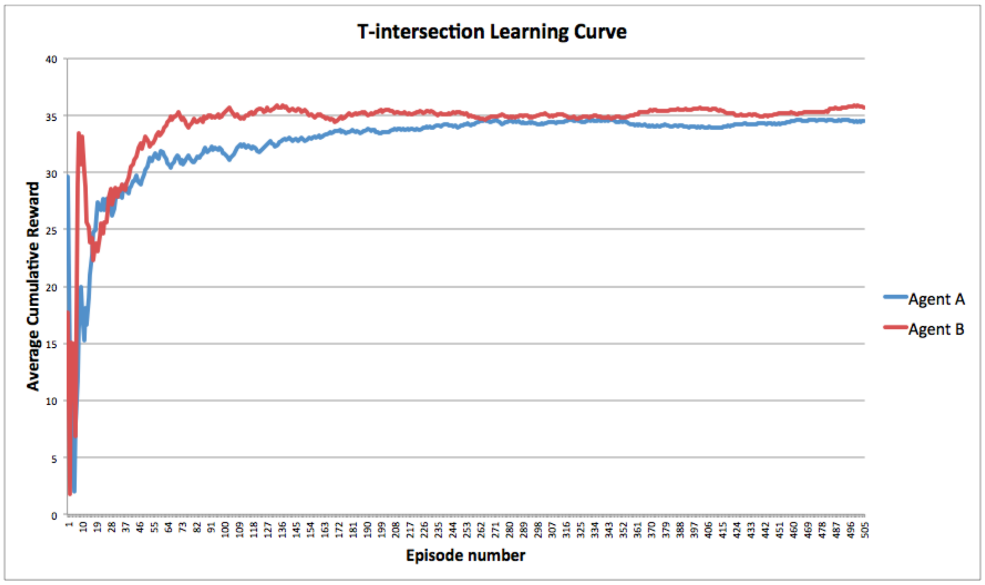


Figure 9-1

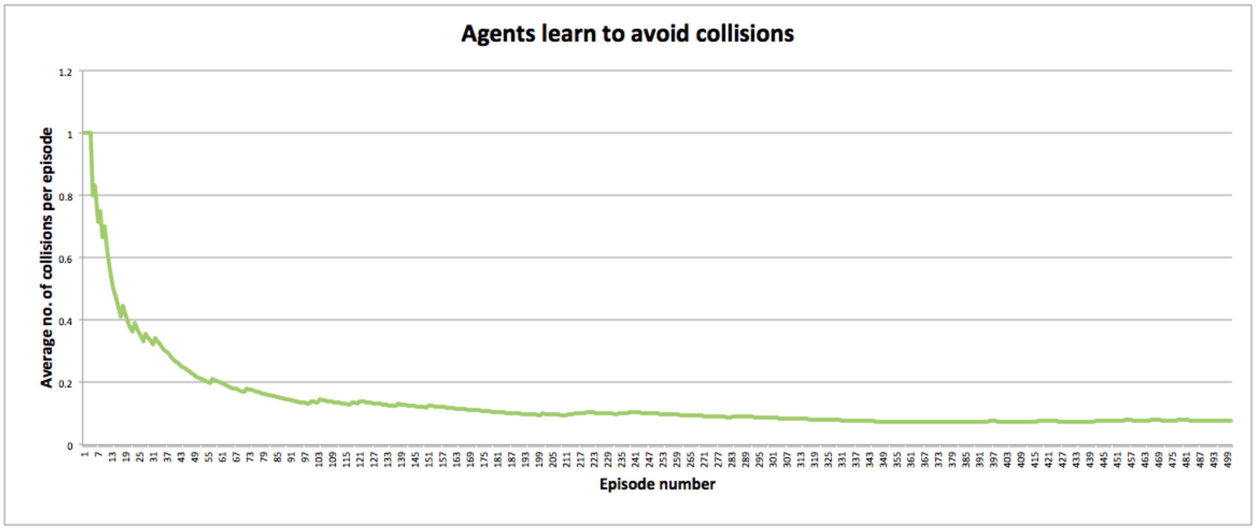


Figure 9-2

9.1.2. Head-to-head cars on a narrow road

Results	Similar to the above, the learning curve in Figure 9-2 displays simultaneous learning. The environment seemed to challenge agents more in the early episodes since Average Cumulative Reward (ACR) was negative. Since the ACR was negative to begin with, this indicates that agents were colliding often. However the agents were able to overcome the collision behaviour by episode ~30 according to the chart. By way of an 'eye-test', one agent was seen to move into the bottom cell in order to give way.
Conclusion	Agents were able to successfully give way and take way when appropriate in order to collaborate and this suggests that this kind of negotiation is teachable in RL and gives hope to scenarios where multiple AVs that are facing head-to-head on a narrow road need to find a solution.

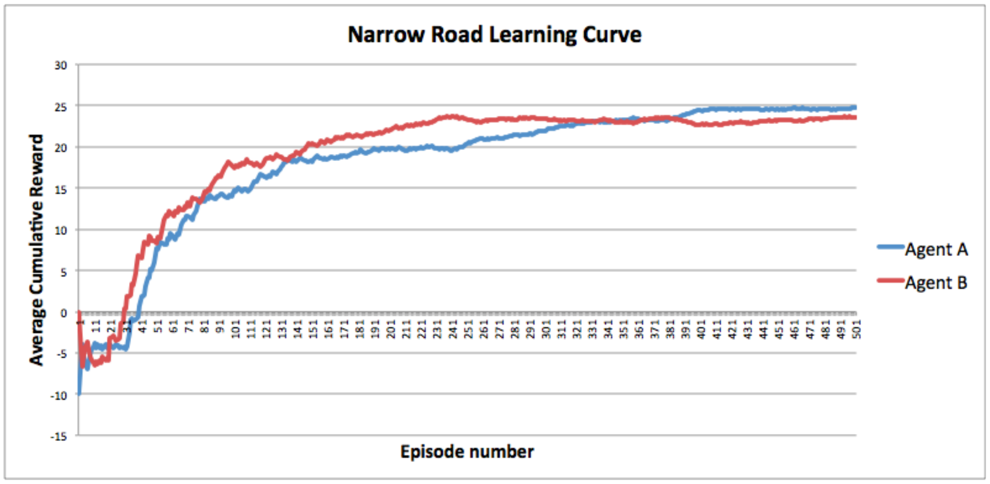


Figure 9-3

9.1.3. Roundabout

Results	The learning curve in Figure 9-3 displays poor ACR. Whilst it is promising that each agent's curve finds a steady state, it can be observed that this steady state value is approximately -5.
Conclusion	In the tabular Q-Learning environments, the reset function caused the entire episode to end when any 2 agents collided. This setup was not ideal, since it may have lead to sparse rewards. It may be interesting to modify the reset function such that collisions only punish agents, but do not reset the entire episode. Thus agents may have a better chance at receiving positive rewards, which could lead to better performance outcomes. With the current setup, agents were not able to negotiate the roundabout environment. By way of an 'eye-test' agents would take several steps before colliding with one another.

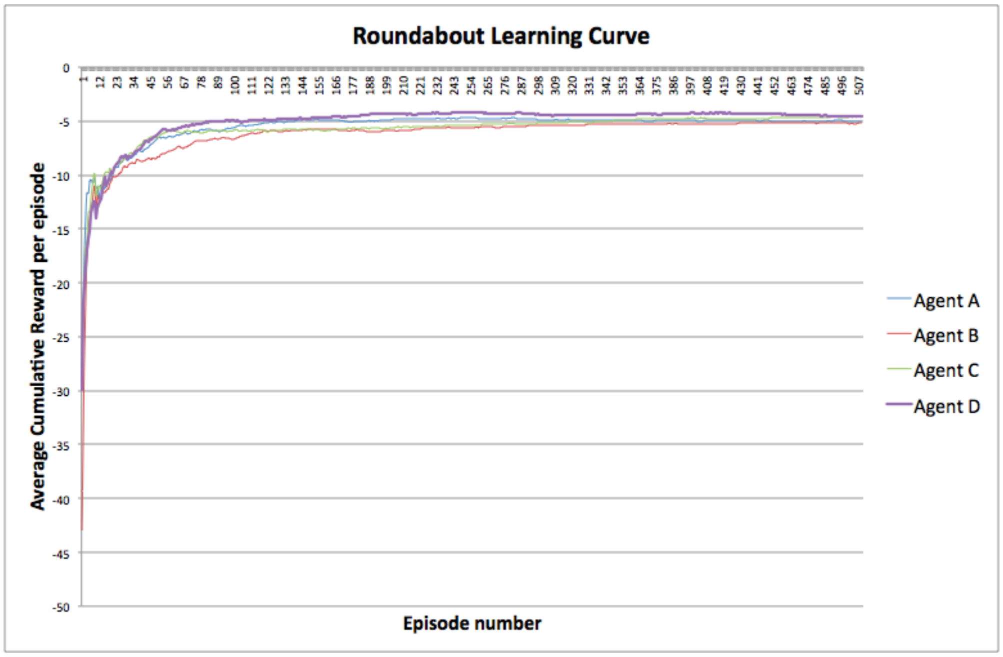


Figure 9-4

9.1.4. "Plus" (+) game

Results	The learning curves in Figure 9-4 display promising results. Though many more episodes of training are required when compared with previous requirements, the trend shows that given enough training time, the learning curves would yield positive values. By way of an eye-test, whilst each agent may take a while to reach its goal, it eventually does.
Conclusion	These are significant results since, even though a simplistic toy environment, there are a vast number of combinations of agent positions and hidden goal destinations. Coupled with the randomness of constantly changing hidden agent goals, the learned policy is quite general. The "plus" environment does not suffer from the sparse rewards problem that the "roundabout" environment suffers from, hence agents are able to learn. We can conclude that it is important for agents to achieve regular rewards for increased training performance.

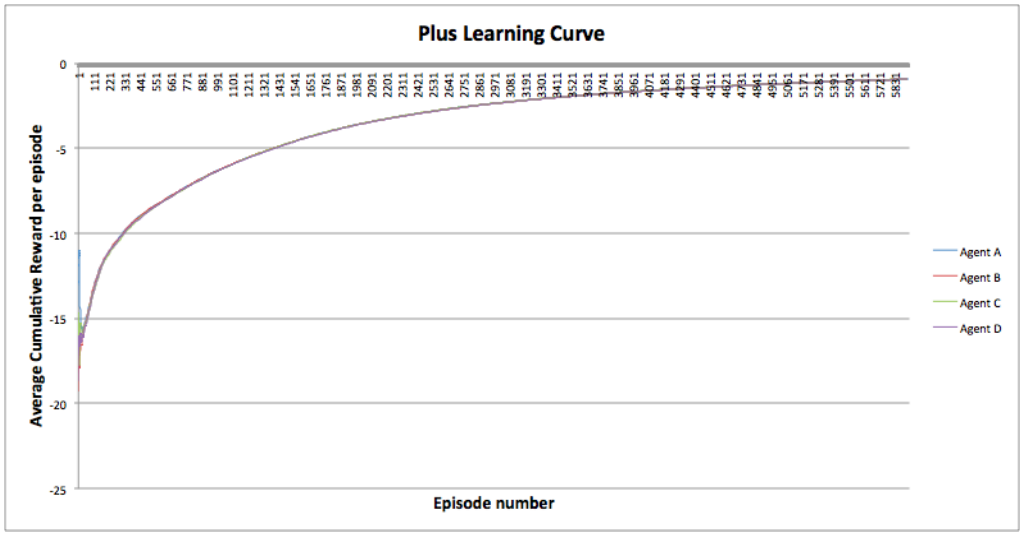


Figure 9-5

9.2. Deep Q-Network approach

Because the training uses a centralised critic, the Mean Episode Reward (MER) is the primary metric that is to be maximised in each of the environments. Each training outputs one centralised MER that encompasses the average reward between all agents.

9.2.1. Self-play

9.2.1.1. Simple point-to-point navigation (1 agent)

Results	According to the eye test, the agent was able to successfully navigate to its goal. See Figure 9-5 for the agent's learning curve.
Conclusion	The reward function, which was based on Euclidean distance between agent to goal, is sufficient for encouraging an agent to navigate towards its goal location.

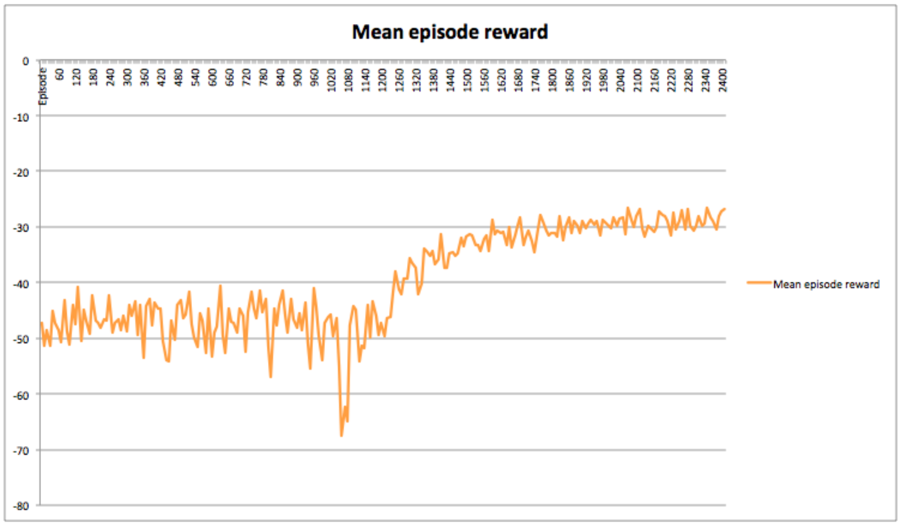


Figure 9-6

9.2.1.2. Simple point to point navigation with an obstacle (1 agent)

Results	According to the eye test, the agent was able to successfully navigate to its goal. However in some situations the agent first collided with the obstacle before navigating around it. In rare cases, the agent became stuck behind the obstacle and never reached its goal. As seen in Figure 9-6, the final MER when an obstacle is present (approx. -55) is lower than when there is no obstacle (approx. -28) from Figure 9-5.
Conclusion	Obstacles can be overcome by agents however they do negatively impact performance.

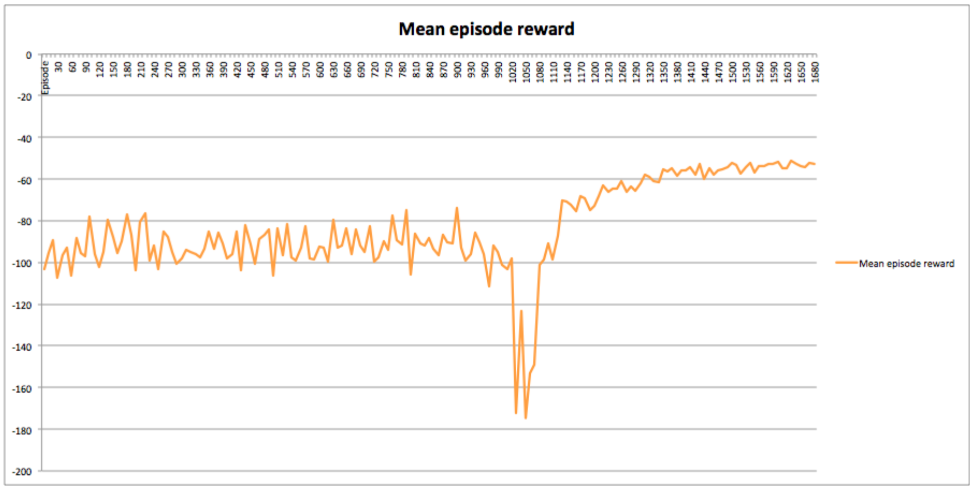


Figure 9-7

9.2.1.3. Navigation and collision avoidance (2 agents)

Results	According to the eye test, the agents were able to successfully navigate to their goals. Agents seemed to avoid each other when needed, however this occurs very seldomly in the environment's current setup.
Conclusion	Each additional agent increases the mean episode reward by approximately -50. From Figure 9-7, it can be observed that 2 agents yield a MER of approximately -100. It seems necessary to force agents into negotiation situations more often. This can be achieved either by adding constraints (walls) to the environment, or increasing the number of agents such that there is less free space to operate, or by increasing each agent's size for the same reason.

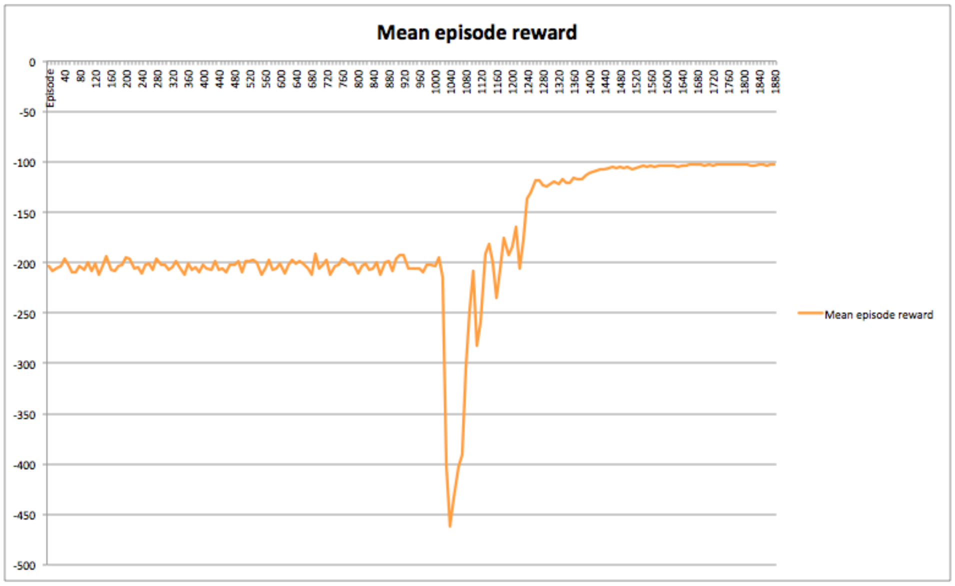


Figure 9-8

9.2.1.4. Navigation and collision avoidance (10 agents)

Results	Agents generally reach their goals, however training is very slow. According to an 'eye-test', it is ambiguous whether agents are truly avoiding one another. This is because agents seem to clip each other from time to time, as opposed to head-on collisions.
Conclusion	It seems that because agents' objectives are to maximise mean episode reward, this means that a calculated decision is made to, from time to time, brush past or collide with other agents. It is essentially a cost-benefit

	calculation for agents. Agents have no inherent understanding that avoiding collisions is a maximum priority, and collisions should be avoided at all costs. This implies that agents' behaviours are very sensitive to the fine tuning of reward function parameters.

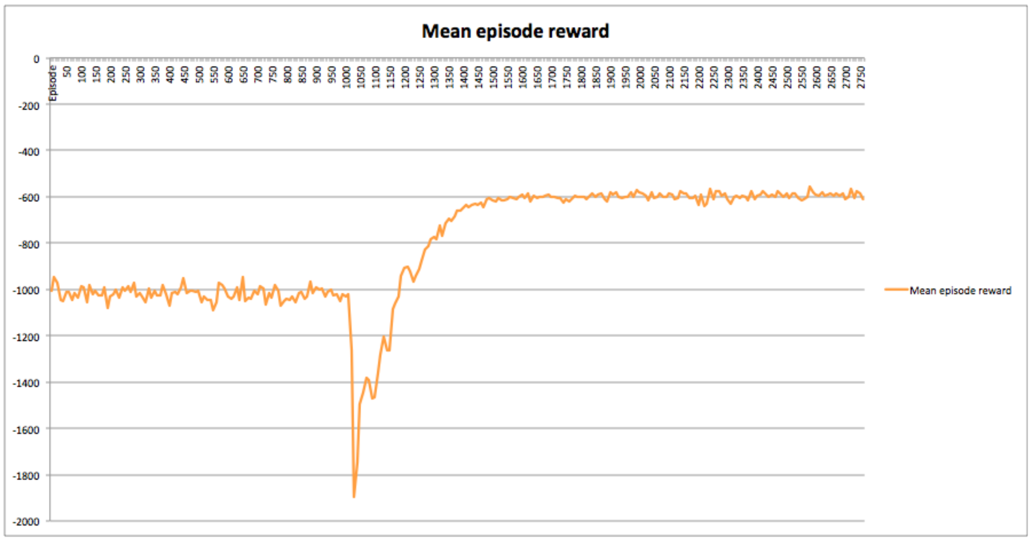


Figure 9-9

9.2.1.5. Navigation, collision avoidance and negotiation (2 agents)

Results	After 48300 episodes, the final MER was -126.7175. Compared with section 9.2.1.3, the MER was slightly lower (-100). According to an 'eye-test' agents were very successful at both reaching their goals and negotiating.
Conclusion	The introduction of a "plus" environment did add significant training time until convergence (48300 episodes compared with 1880 in Figure 9-7). This is not surprising since the environment has introduced many restraints on actions, and hence rewards are more sparse. The slightly lower MER is most likely attributed to the extra distance each agent would need to travel to their goal. In section 9.2.1.3, each agent could move in more or less a straight line to their goal, however now agents are required to move into the centre of the "plus" and out again, adding distance and hence more step cost.

9.2.1.6. Navigation, collision avoidance and negotiation (3 agents)

Results	After 60000 episodes, the final MER was -524.7362. An 'eye-test' revealed that agents were more or less able to negotiate and somewhat able to reach their goals. However it was not clear whether agents could do this on a consistent basis. Rarely would collide head-on with each other, but it would occur. Whilst for the most part, they exhibited the correct cooperative behaviour.
Conclusion	Similar to 9.2.1.4, these results that agents are very dependent on the reward function parameters that are set. Emergent behaviours are very sensitive to the smallest changes in these parameter values.

9.2.1.7. Navigation, collision avoidance and negotiation (4 agents)

Results	After 60000 episodes, the final MER was -1401.5645. An 'eye-test' revealed that agents were somewhat able to negotiate and reach their goals. However this behaviour was certainly not consistent. Often, agents would become stuck and unsure of how to proceed.
Conclusion	Compared with section 9.2.1.6, the MER was almost 3 times worse, which was not linearly proportional to adding another agent (expected to be an addition of approx. -175). Adding additional agents to an already congested environment with sparse rewards seemed to overwhelm the algorithm.

9.2.1.8. Navigation, collision avoidance and negotiation (5 agents)

Results	After 54400 episodes, the final MER was -1884.85689. An 'eye-test' revealed that agents could not reach their goals or negotiate at all. Agents seemed lost and unsure of what their objectives were. Very poor performance.
Conclusion	For similar reasons to section 9.2.1.7, the algorithm simply could not deal with too much congestion and sparse rewards.

9.2.2. Parametric Sweep

The results of simultaneously sweeping through reward function parameters are discussed in this section. The inputs were the reward function parameters:

- Crash punishment (0-9)
- Cooperative reward (0-9)

The outputs were the benchmarked metrics:

- Average number of collisions per episode per agent.
- Average number of goals reached per episode per agent (0-1).

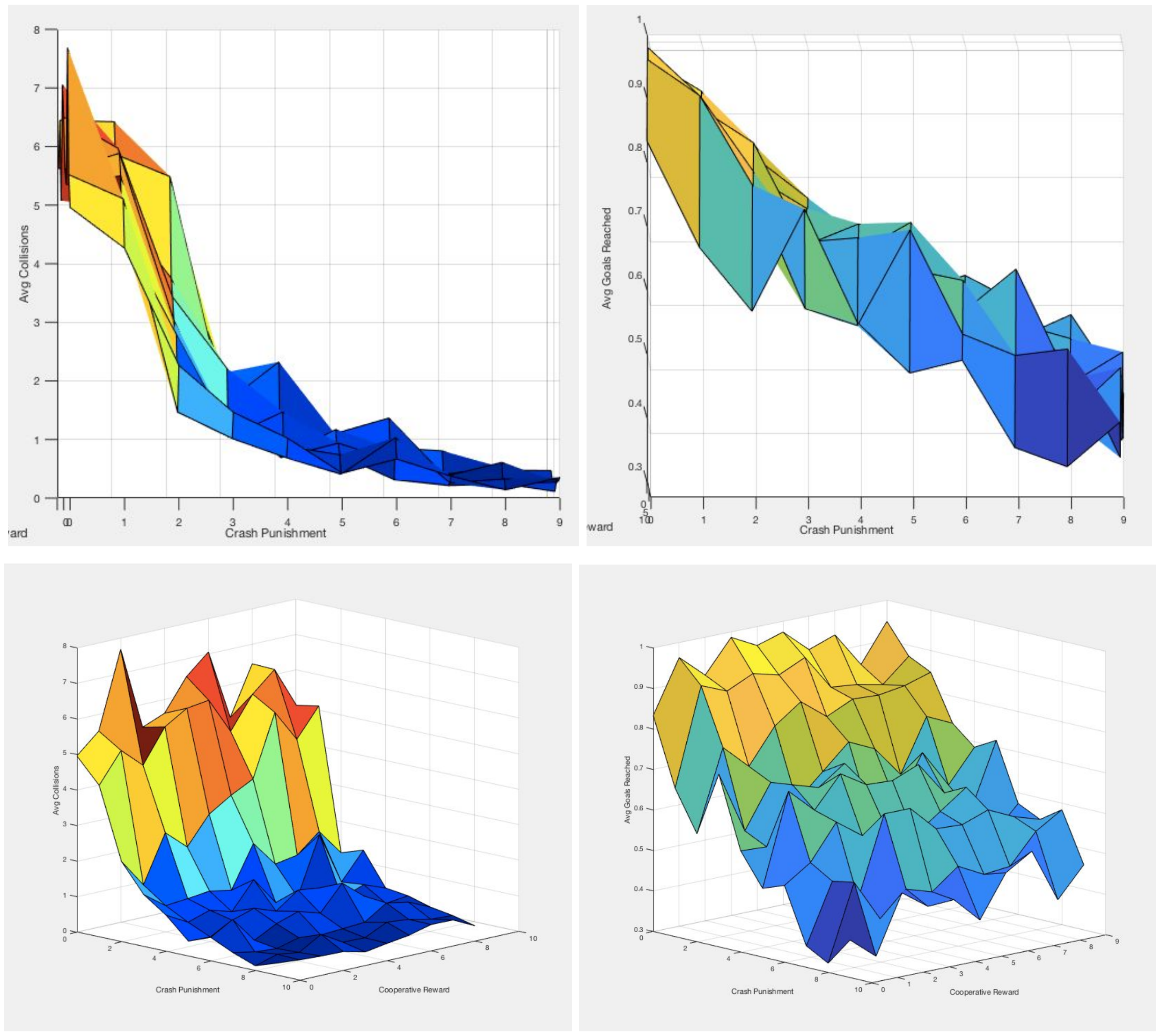


Figure 9-10

The results in Figure 9-9 indicate a high inverse correlation between “crash punishment” and average number of collisions. The relationship, as observed in the top left figure, does not appear to be linear, but rather more of a decaying exponential. The results also indicate that there is a high inverse correlation between “crash punishment” and average goals reached. According to the top right figure, the relationship appears more or less linear. The “cooperative reward” parameter showed no real correlation to any of the outputs. Thus it can be concluded that the only input that caused variation in the output of this experiment was the “crash punishment”.

Essentially this means that the more sensitive agents were to collisions with each other, the more defensive their behaviour. It is intuitive that defensive behaviour leads to less collisions, which is desirable, but occurs at the cost of not reaching goals, which is undesirable. From this analysis it is preferable to choose a “crash punishment” value of approximately 5. At this value, there is a reasonable low level of collisions, whilst goals are somewhat reached. Any value of “cooperative reward” is suitable because of the little effect it has on the output. However in this experiment, no specific combination of parameters were found that achieve the ideal behaviours.

9.2.3. Human-AI Interaction

9.2.3.1. “Rugby” game

Although qualitative, the results of the rugby game were interesting. It was found that the defender agents (red), whose policies were trained via MARL, attempted to minimise the distance between their positions and the attacker agent (blue). This is the desired behaviour. The defender agents were behaved in the same manner regardless of whether the attacker agent was MARL-trained or human-controlled. It was found that no combination of keyboard inputs by a human could throw the defender agents off by prompting unexpected behaviour.

9.2.3.2. “FYP” structure

The FYP structure was found to more or less be resilient to the random and unpredictable movements of the human “destructor” adversary. Results were qualitatively measured, with an eye-test, and displayed that each agent was able to follow its own objective function in order to move to its own desired location. Some agents, however, did seem to dwindle a little.

10. Conclusions and Future Work

10.1. Self-play

The MARL algorithm seemed to work successfully when scenarios were simple (1 or 2 agents, with or without obstacles). However introducing 3+ agents seemed to create issues in achieving consistent desirable behaviour. In these more complex scenarios, desirable behaviour was observed, however on a sporadic basis.

MARL adds another level of complexity where agents' actions affect the state of other agents, especially when agents physically block each other from their goal locations. The neural network is required to learn an optimal policy for all agents, however this task becomes difficult when one agent causes a chain of blocking. For example, an agent may block another agent who, by attempting to resolve the conflict, ends up unintentionally creating more blocking with other agents. Perhaps a change of scenario is necessary to show similar behaviours but in a less physically constrained environment (ie. less obstacles).

Surprisingly, on the other hand the tabular Q-learning scenarios showed very consistent desirable behaviour. However tabular methods are limited by the number of agents and subsequently states in the environment. If number of agents > 4, tabular environments would have very long and impracticable load times.

10.2. Parametric Sweep

RL is good at optimising the reward function. However when attempting to produce a specific behaviour, a lot of work must be done in order to set precise reward parameters. Additionally, the reward function itself must be set up in such a way to create the correct incentives. These are not a trivial tasks.

The failure of the experiment to display the ideal behaviours could be an artefact of the experiment setup. Perhaps the reward function is too sparse for agents to really understand how to collaborate in a consistently desirable manner. Another limitation might reside in the parametric sweep itself. Perhaps a wider range of "cooperative reward" should be explored, which may increase the "average goals reached" metric. Future work would involve enhancing this experiment to produce more consistent desirable output behaviours.

10.3. Human-AI Interaction

The rugby game displayed a level of certainty in transferring RL-trained agents from the simulation realm to a realm where unpredictable human behaviour exists. Whilst the human interacted with the RL agents within the context of a computer game, the next logical step would involve training RL agents offline in simulation followed by testing in the real physical world with human interactivity.

The “FYP” structure again shows that RL trained agents are resilient to unpredictable human behaviour. However some agents were shown to dwindle. I suspect the reason is the long training time needed in order to reach a optimal policy with 42 agents. Scaled back to 3 agents in a rudimentary experiment, the agents were able to reach and stay at their goal locations with more consistency and less dwindling.

An interesting area for future work is the concept of mixed training. That is, a mixture of Imitation Learning and Reinforcement Learning. In the case of AV learning, the concept can be analogous to teaching a new driver to drive. For example, a child grows up as a passenger in a car, only watching their parent’s driving behaviour. This can be linked to the imitation component of the learning. When, finally, the child learns to drive they already start with a base of knowledge of the desirable driving patterns. It now is a matter of learning-by-doing in order to refine their skills. This is the reinforcement component of the learning. Many engineering designs and architectures have been borrowed from nature, such as how a plane wing design generates lift similar to birds. The concept of mixed training may be another such example of borrowing natural principles in engineering design.

11. References

[1] Van den Berg, J., Lin, M. and Manocha, D., 2008, May. Reciprocal velocity obstacles for real-time multi-agent navigation. In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on (pp. 1928-1935). IEEE.

[2] Foderaro, G., Ferrari, S. and Wettergren, T.A., 2014. Distributed optimal control for multi-agent trajectory optimization. Automatica, 50(1), pp.149-154.

[3] Lagoudakis, M.G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A.J., Koenig, S., Tovey, C.A., Meyerson, A. and Jain, S., 2005, June. Auction-Based Multi-Robot Routing. In Robotics: Science and Systems (Vol. 5, pp. 343-350).

[4] Open AI, “FrozenLake-v0.” [Online], Available: https://gym.openai.com/envs/FrozenLake-v0/. [Accessed Aug. 25, 2018].

[5] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. Nature, 518(7540), p.529.

[6] Sutton, R.S., McAllester, D.A., Singh, S.P. and Mansour, Y., 2000. Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems (pp. 1057-1063).

[7] Shalev-Shwartz, S., Shammah, S. and Shashua, A., 2016. Safe, multi-agent, reinforcement learning for autonomous driving. arXiv preprint arXiv:1610.03295.

[8] Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P. and Mordatch, I., 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In Advances in Neural Information Processing Systems (pp. 6379-6390).

[9] Omidshafiei, S., Pazis, J., Amato, C., How, J.P. and Vian, J., 2017. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. arXiv preprint arXiv:1703.06182.

[10] Tesauro, G., 1995. Temporal difference learning and TD-Gammon. Communications of the ACM, 38(3), pp.58-68.

[11] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587):484–489, 2016.

[12] OpenAI. OpenAI Dota 2 1v1 bot, 2017. URL https://openai.com/the-international/ .

[13] Bansal, T., Pachocki, J., Sidor, S., Sutskever, I. and Mordatch, I., 2017. Emergent complexity via multi-agent competition. arXiv preprint arXiv:1710.03748.

[14] Bowling, M. and Veloso, M., 2003, August. Simultaneous adversarial multi-robot learning. In IJCAI (Vol. 3, pp. 699-704).

[15] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T. and Lillicrap, T., 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815.

[16] Thurau, C., Bauckhage, C. and Sagerer, G., 2004, November. Imitation learning at all levels of game-AI. In Proceedings of the international conference on computer games, artificial intelligence, design and education (Vol. 5).

[17] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J. and Zhang, X., 2016. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316.

[18] Hamahata, K., Taniguchi, T., Sakakibara, K., Nishikawa, I., Tabuchi, K. and Sawaragi, T., 2008, November. Effective integration of imitation learning and reinforcement learning by generating internal reward. In Intelligent Systems Design and Applications, 2008. ISDA'08. Eighth International Conference on (Vol. 3, pp. 121-126). IEEE.

[19] A. Juliani, “Simple Reinforcement Learning with Tensorflow Part 0: Q-Learning with Tables and Neural Networks,” Medium, August 25, 2016. [Online], Available: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-0-q-learning-with-tables-and-neural-networks-d195264329d0. [Accessed Oct. 5, 2018].

12. Appendix

View the “Self-play” test logs at:
https://docs.google.com/spreadsheets/d/1A_PyMzVX_amdu_8cTvVNfNAc_x4i3zsD3nhQw8AGP2M/edit?usp=sharing

View the GitHub repositories at:
https://github.com/dylman123/Multi-Agent-RL (for tabular Q-Learning related work)
https://github.com/dylman123/multiagent-particle-envs (for DQN related work)