UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Aqeel Labash

Using Deep Reinforcement Learning to Solve Perspective-Taking Task

Master's Thesis (30 ECTS)

Supervisor: Raul Vicente, PhD
Supervisor: Jaan Aru, PhD
Supervisor: Tambet Matiisen, MSc
Supervisor: Ardi Tampuu, MSc

Tartu 2017

Using Deep Reinforcement Learning to Solve Perspective-Taking Task

Abstract:
Perspective taking is the faculty that allows us to take the point of view of another agent. This capability is not unique to humans. This is an essential ability for agents to achieve efficient social interactions, including cooperation and competition. In this work, we present our progress toward reverse engineering how perspective taking task might be accomplished in our brains. We introduce an environment designed from scratch for the purpose of creating perspective-taking tasks, in which the environment is partially observable by its agents. We also show a set of different models that were able to pass multiple tests that would benefit from perspective taking capabilities. These models were trained using reinforcement learning algorithms assisted by artificial neural networks.

Keywords: Reinforcement learning, Theory of mind, Perspective taking, Deep learning, Neural networks

CERCS: P176, Artificial intelligence; P170, Computer science, numerical analysis, systems, control

Vaatenurga võtmist vajavate ülesannete lahendamine stiimulõppe abil

Lühikokkuvõte: Võime näha olukorda kellegi teise vaatenurgast on oluline oskus osalemaks mitme agendi koostöös või nendevahelises võistluses. See võime ei ole ainuomane inimestele. Antud töö uurib, mismoodi kellegi teise vaatenurga mudeldamine võiks toimida meie ajus. Selleks loodi virtuaalsetele agentidele keskkond, milles iga agent näeb ainult osa sellest. Näidatakse, kuidas erinevate stiimulõppe meetoditega on võimalik lahendada ülesandeid, mille puhul on kasu teise agendi vaatenurga mudeldamisest. Agendid kasutavad vastase vaatenurga modelleerimiseks tehisnärvivõrke.

Võtmesõnad: Stiimulõpe, vaimuteooria, vaatenurga võtmine, sügavõpe, närvivõrgud

CERCS: P176, Tehisintellekt; P170, Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1 Introduction

In real life many decisions we take depend on others, what they think, what they believe, and what we know they know. A thief will not try to steal under the owner's surveillance. Somewhat surprisingly not only humans have the ability to take into consideration what others know. In controlled experiments it has been shown that chimpanzees know what other chimpanzees see and know [HCAT00].

What about the state of the art artificial intelligence (AI) agents - could artificial agents using neural networks also infer what other agents perceive and know? Richard Gregory, a great psychologist and the author of "Eye and brain" has said "It may be necessary to invent imaginary brains-by constructing functional machines and writing computer program to perform perhaps much like biological systems. In short, we may have to simulate to explain; though simulations are never complete or perfect." [Gre15]. Having an AI agent able to simulate a biological behavior can help us to better understand perspective taking.

To have the ability for computational simulation of the biological experiments, a simulator was needed. For this purpose we designed an environment `APES` with a great extent of customizability to experimental needs. The environment allowed us to simulate experiments in [HCAT00] where a subordinate chimpanzee chooses whether to go for food or not depending on the observability of the food by the dominant chimpanzee.

Our approach was mainly about training as simple as possible artificial neural network with reinforcement learning to control an agent to learn this behavior. We tried a variety of approaches. In total more than 700 network models were trained which amounted to 1 year of computational time.

To assess the trained models ability for perspective taking tasks, more than 15000 simulation on specific defined tests were done. Few of the trained models, which have high performance as well, were able to reach similar results of what a perspective taking capable agent would reach.

In this thesis we will talk in the background about theory of mind and perspective taking. What are they, and how relevant they are to our life with examples. Afterward we will go over deep learning (DL) and reinforcement learning (RL). Followed by famous different approaches for combining DL and RL together.

The methods chapter will explain in details the environment features and capabilities. We will take a look at the network model behind the AI agent and explore its parameters, input and output and learning paths or curriculum learning. Methods also explain in details the test cases we used for testing the models for perspective taking behavior.

In the end we will see the results and discuss them concluding with the limitation of current work and our plans for the future work.

2 Background

2.1 Theory of mind

According to [PW78] theory of mind is the ability to infer the purpose, intention, knowledge, belief, thinking, doubt, guessing, pretending, liking, etc... from the behavior of another agent. It is called theory because mental states of other agents are not directly observable [PW78]. However, this faculty is essential to navigate in the social world because we depend on it to explain others' behavior [FF05]. Imagine that your family has the tradition of storing pins in a metal candy container. A family member would approach it for pins while guests would approach it for candies and they do not need to tell you why they approach it - you know what they want, you can infer that from their behavior. This is an example of false-belief, which is one of theory of mind branches [App11]. ToM allows us to know even more than that. If two persons were in one familiar room to both of them, neither of them will need to switch places to imagine what another can see and cannot see. From these examples we can understand how much we use ToM even without thinking about it. Indeed, it is thought that ToM is our natural way to infer others mental states by processing their actions in our own minds [App11]. Hence, developing a theory of mind is an essential faculty for an efficient interaction with other agents, including a successful cooperation with peers and competition with rivals.

2.1.1 Perspective Taking

Looking at things from a different perspective that differs from our own is called perspective taking [RBTBS15]. It could be defined as "the cognitive capacity to consider the world from another individual's viewpoint" [Dav83]. According to Galinsky and colleagues, it is one of the social competencies that motivates social understanding in many contexts [GMGW08]. When we describe the way for a foreigner, we describe it differently than if the person is familiar with the city [KF91]. Another example for perspective taking is when you tell your friend to swipe ice cream leftover from his cheek based on your left or his right side of the face.

Intriguingly, perspective taking ability is not unique to humans but can be also observed in other species like the chimpanzees [HCAT00]. Tomasello and others, did several perspective taking experiments on these great apes. Chimpanzee social status is organized hierarchically (dominant, subordinate) [GW97]. When there is food available that both can reach, the dominant always gets it. The experiments were about putting dominant and subordinate chimpanzees in open doors cages with different position for

food as shown in Figure 1 (left). The three conditions differ mainly in what the dominant and the subordinate can see. For example in the "subordinate door" condition one piece of food cannot be seen by the dominant chimpanzee. Can the subordinate take advantage of this fact? The results are shown in Figure 1 (right) and demonstrate that the subordinate obtained more food in this condition. Hence, the subordinate was able to take into account what the dominant chimpanzee could see. In other words the subordinate could take the perspective of the dominant chimpanzee. For other experiments and details see for example [HCAT00, DW16, Tom09]. These experiments show that chimpanzees can

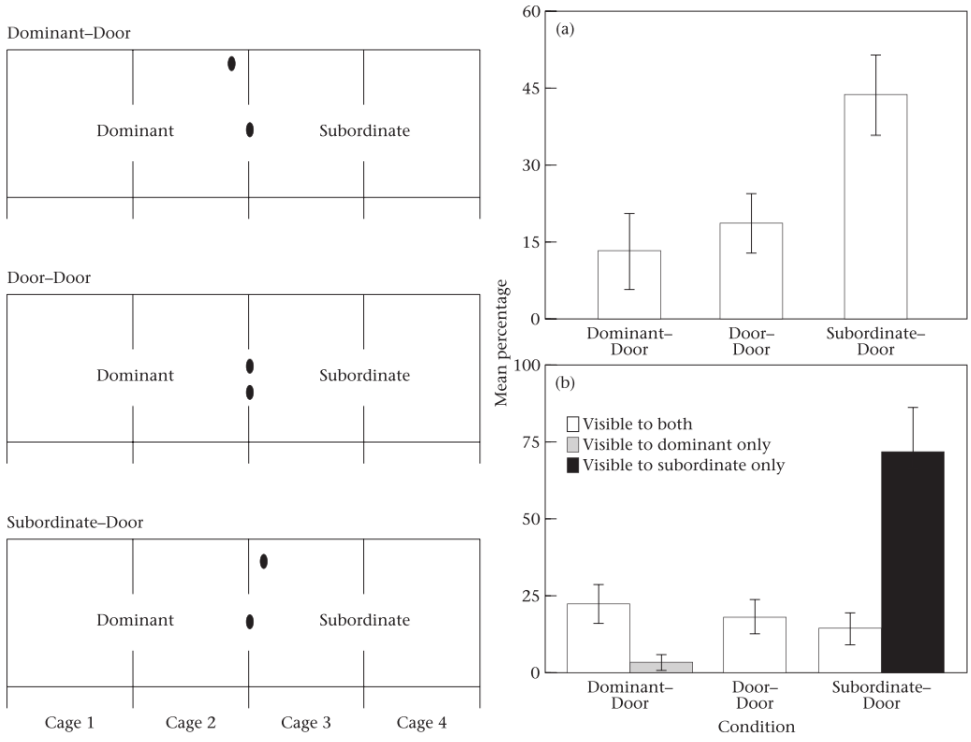


Figure 1. Left: three different positions for the food with different ability to observe it (food indicated by black ellipses, cage walls indicated by lines). Right top: Average percentage eaten food between different food positions. Right bottom: the average percentage based on food visibility condition. [HCAT00]

do perspective taking tasks, otherwise they would pick any food randomly regardless of its position. The aim of this thesis is to study whether an agent controlled by a neural network can learn to solve similar perspective taking task using reinforcement learning.

2.2 Reinforcement Learning

Reinforcement learning (RL) is a branch of artificial intelligence that allows an agent to learn by trial and error while interacting in an environment. In particular, the agent must learn to select the best action in each specific state to maximize its accumulated reward. A reward is a must for the agent to optimize when an action should be taken [SB98]. The agent can an autonomous robot [Lin93] [YG04] [RGHL09] or a character in video game [MKS+13, TMK+17] or even computer simulations based of biological systems [AS07].

The idea behind learning by interacting with an environment is inspired from how human and animal infants learn from the rich cause-effect or action-consequence structure of the world [SB98, Tho11, SDM97]. Therefore, RL is a biological plausible method of learning certain associations and behavior. Recently, RL has succeeded to make machines learn difficult tasks at which they achieved human level [MKS+15], and even outperform human at some cases as in the board game of Go [SHM+16]. In the last example with AlphaGo, RL managed to solve the problem although learning an explicit strategy is not possible due to computation and memory challenges.

2.2.1 Q-learning

Q-learning [Wat89] [WD92] is one of the most important algorithms in RL [SB98]. Its based on dynamic programming techniques to limit the computational needs [WD92].

In Q-learning a fundamental quantity is the action-value function $Q(s, a)$ which describes the outcome for taking action (a) from state (s) [SB98]. One can think of the action-value function as a 2d array were one dimension indicates all possible states while the other indicates all possible actions. Every value in that array represents the expected gain from taking action a while being at state s. Since we do not have this function it is approximated by the Q-learning algorithm, which is based on Bellman optimality equation and it has been proved to converge to the optimal action-value function $Q^*(s, a)$ with probability 1 if all actions are sampled in all states repeatedly [WD92].

The update rule describing the Q-learning algorithm is shown in Equation 1

$$Q(s_t, a_t) \leftarrow\!\rightarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] , \tag{1}$$

where s_t, a_t are the state and action at time t, α is a learning rate, r_{t+1} is the reward obtained at time $t + 1$, λ is the discount factor for future rewards, and $\max_a Q(s_{t+1}, a)$ is the highest possible outcome starting from the new state.

Although one can in theory solve any RL problem using Q-learning, in practice large problems scale quickly in the number of states which create huge memory and computation limitations. This renders a naive implementation of Q-learning unfeasible in many cases. For the environments explored in this thesis the number of states is around 24 million which typically exceeds the capabilities for a practical tabular Q-learning implementation.

2.3 Deep Learning

Deep learning typically refers to a set of artificial neural networks which are composed of several stacked layers of non-linear units [GBC16].

Artificial neural networks (ANN) could be understood as "highly simplified model of the biological neuronal networks" [YEG09] and thus, help us to gain some insights into the algorithms that biological neural networks could be using to solve certain tasks. In this thesis we can consider deep learning as the brain of our agent that processes the input, the state, and produces the output action while solving a perspective taking task.

In the past few years there has been a lot of work on deep learning and its methods achieved spectacular results such as adding sound to mute video [OIM+16], image colorization [ZIE16, LMS16, HZ, CYS15] or machine translation [BCB14, LCH16]. RL also benefited from deep learning techniques and they were used for AlphaGo, [SHM+16], the first agent to beat worldclass players in the board game of Go. Another successful example of using deep learning methods in RL is the Deep Q-learning algorithm proposed by DeepMind, which learn to play Atari games reaching human performance [MKS+15] or even surpass it in some games [MKS+13]. Next subsection briefly reviews the workings of Deep Q-learning.

2.3.1 Deep Q-Networks

Deep Q-Networks (DQN) is the result of combining deep learning with the Q-learning algorithm. The basic idea is to use a deep neural network to approximate the Q-value function. The algorithm proceeds by saving transitions (state s_t, action a_t, reward r_t, and the new state s_{t+1}) to be used later to train a network parametrized by a vector θ. The network is trained to decrease the distance between its prediction $Q(s_t, a_t; \theta)$ and the target value y_t given by [MKS+13]:

$$y_t = \begin{cases} r_t, & \text{if its terminal state} \\ r_t + \lambda \max_{a'} Q(s_{t+1}, a'; \theta), & \text{otherwise .} \end{cases} \tag{2}$$

In another more recent version the authors from [MKS+15] used two networks (a training network parametrized by θ and a target network parametrized by θ^- which is updated at every time step according to ($\theta^- = \theta \times \tau + \theta^- \times (1 - \tau)$). In this case the cost function to be minimized by the training network is the distance between its prediction and the following target value:

$$y_t = \begin{cases} r_t, & \text{if its terminal state} \\ r_t + \lambda \max_{a'} Q(s_{t+1}, a'; \theta^-), & \text{otherwise .} \end{cases} \tag{3}$$

2.3.2 Double Deep Q-learning

Double Deep Q-Network (DDQN) is an improved version of DQN to get rid of some biases occurring in the estimation of Q-values. The change proposed follows from adapting the double Q-learning algorithm to DQNs [HGS16]. The resulting algorithm chooses the action using the training network but estimates the value using the target network as shown in Equation 4:

$$y_t^{DDQN} \equiv r_t + \lambda Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta); \theta^-) \,. \tag{4}$$

In this thesis we used DQNs, DDQNs, and Dueling networks [WdFL15], which we explore in Section 3.3.1.

3 Methods

To pursuit our mission in training a model capable of solving tasks that require perspective taking, three things were needed. First, we needed to implement an environment specially designed for this purpose. The environment needed to be simple and focused on the task not to distract the research efforts on representations and feature analysis. Secondly, we needed to find a neural network model that could solve these tasks. Last, we needed to find out a training strategy, because the task is not easy to be learned in one shot. In this section we will visit those ideas one by one and describe the tools necessary to explore them.

3.1 Environment simulator

The environment (APES) was built as general as possible in the scope of perspective taking tasks and experiments conducted by Michael Tomasello and colleagues [HCAT00]. We did so with the view to use the environment in different types of tasks in our research.

APES is 2d grid world environment were every cell could be filled with food, obstacle, an agent, or could be left empty. The environment is programmed using python programming language and using Anaconda [ana16]. Figure 2 shows an example of the environment. In the following subsections we explain the environment use and

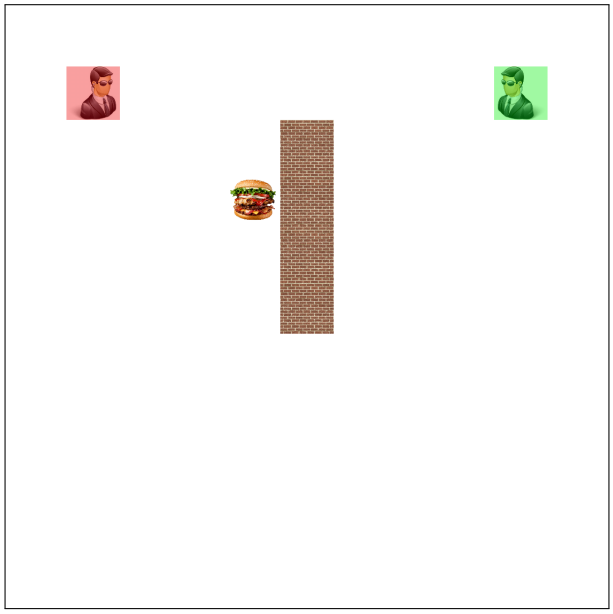


Figure 2. Example of the environment showing 2 agents, 1 food (reward) and 1 obstacle.

its capabilities in more detail. For clarity purposes we will focus on the abilities of

the environment rather than their algorithmic implementation. The algorithms used in aspects such autonomous movement of the dominant agent (A* algorithm), field of vision (recursive shadow casting), or world generation will be described in detail elsewhere.

3.1.1 General settings

APES environment has many aspects that could be configured to match research needs.

1. **World size:** the environment allows customizable world size as the established experiment require.
2. **Elements:** each environment cell can take one of three types of elements which are agent, obstacle, and food. These elements are explained in more detail in subsequent sections.
3. **Images:** each element in APES can have a different image.
4. **Cell size:**this feature allows one to present the experiments with different graphical qualities for better visualizations.
5. **Vision block:** all the elements in the environment have the ability to block agents' vision. Example is shown in Figure 3

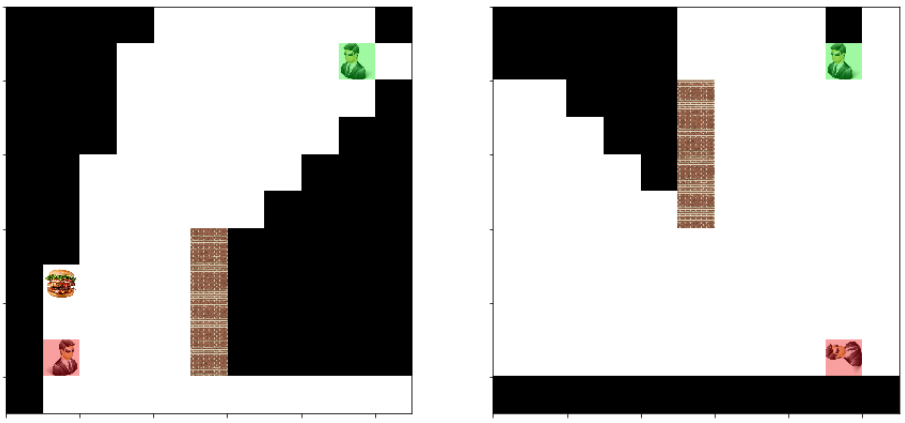


Figure 3. Two examples to show how environmental elements act as vision barriers for agents. The images represent red agent -subordinate agent- A_{sub} point of view.

6. **Elements' distribution matrix:** each element can have a distribution matrix which specify where an element can spawn at the beginning of each game or episode. This matrix will be converted to the probability matrix using the Dirichlet distribution. An example is shown in Figure 4.

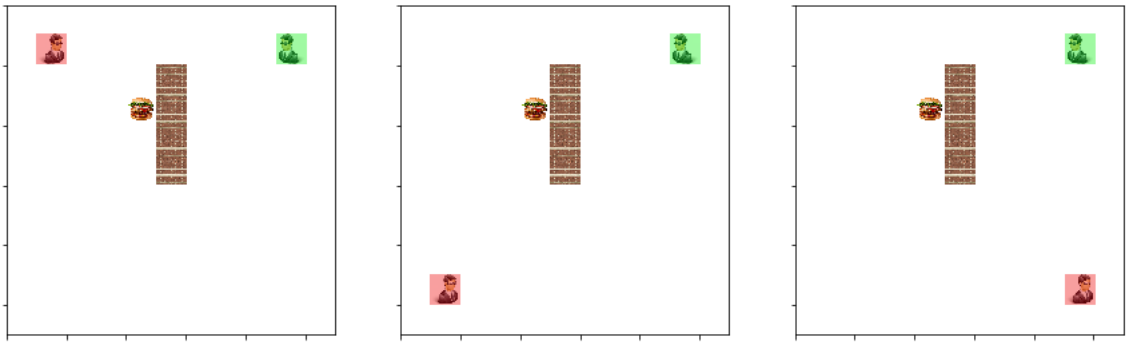


Figure 4. Example shows all three possible initial locations for the red subordinate agent A_{sub} when using a matrix with positive values at locations $(1, 1), (9, 1), (9, 9)$ and filled with zeros for other entries.

7. **Performance:** refers to the time between sending an action and its result in the environment. This update time is critical for training a neural network in the environment since it can increase the training time and become a bottleneck for the training of the network. In the environment the performance depends mainly on the number of agents and the computation power provided.

8. **Customizability:** usually many parts differ dramatically between different tasks. For that we implemented the environment with the ability to inject specific reward function (for rewarding strategy), and output function to specify the desired representation for the agent state.

9. **Actions:** there are 9 primitive actions for the agents. One action is a no-ops action or non-operational action, the rest are divided to two groups. First, looking actions, which change were the agent is currently looking and, second, moving actions, which move the agent one step in the specified direction. We can see the two groups in Figure 5. Any more complex set of actions could be generated. We used 5 complex actions, 4 to look and move in same direction and the last a no-ops action coded as `possible actions`.

3.1.2 Agents

The environment we developed can take more than 1 agent. The total number of agents depends on the needs of the task. Below we see the general features for the agents.

- **Power:** determines the dominance hierarchy between agents. The agent with more power is more dominant.

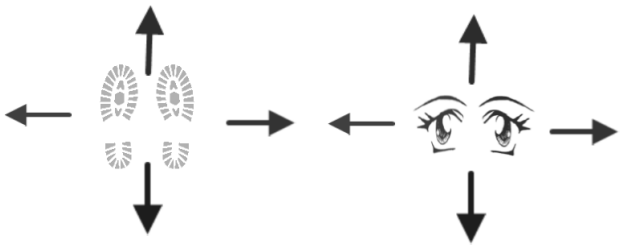


Figure 5. All possible actions represented in moving or looking in 4 directions.

- **Control Range:** determines the effective range of the agent. Any opponent agent in this range would be punished according the the reward scheme3.4.5.
- **Role order:** determine whose actions are executed first, this feature allows one to execute dominant moves before those of the subordinate.
- **Vision:** agent has different options for vision. To calculate the vision fields we used a "field of view using recursive shadowcasting" algorithm as described in [Ber].
  - **Altercenteric:** it is a type of vision representation where the agent sees his position and orientation change according to his movement action. One can imagine it as if the agent looks to the environment from a zenithal camera from the ceiling. Example of the field of vision for this type of representation is shown in Figure 6.

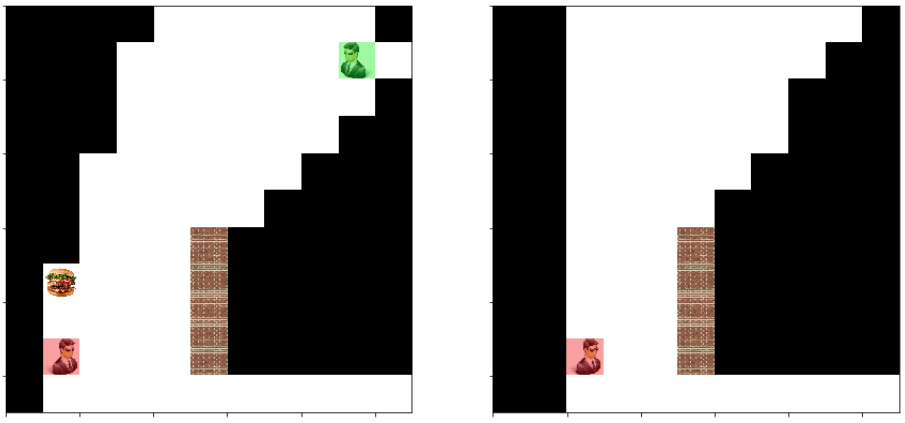


Figure 6. Example of an agent's move to east in altercentric mode.

– **Egocentric:** it is a type of vision representation closer to the natural way of looking at the world. This way of representation of the world puts the viewer always in the center as in real life. Example is shown on Figure 7.

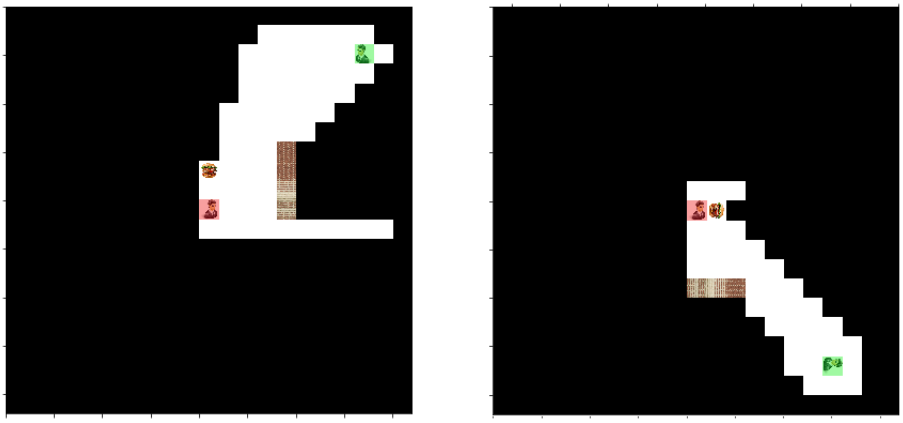


Figure 7. Example of an agent's field of vision change in egocentric mode when moving north and rotating 90° counter-clockwise.

– **Vision angle:** each agent can have a specific angle of vision, which allows for more realistic simulation. This is illustrated on Figure 8.

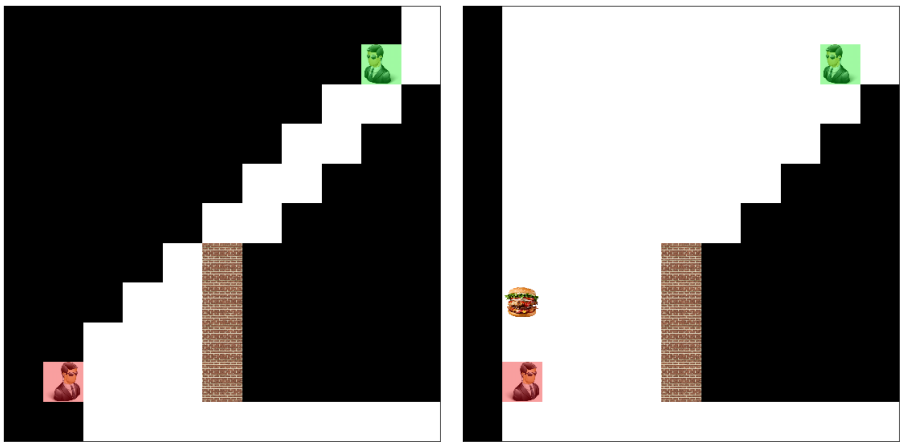


Figure 8. Example of an agent with different vision angles. Left panel: 90°; right panel: 180°.

– **Vision limit or range:** determines maximum distance the agent can see as shown in Figure 9.

Figure 9. Example of agent with different vision ranges. Left panel: range of 3; right panel: range of 6.

3.1.3 Auto pilots

APES environment has two auto-pilots to generate actions for the agents. Those agents could be used as baseline or as competitor or as the dominant based on the needs.

Random agent is an agent who choses actions randomly. Every time step the agent picks one random possible action (from the list described in the item 9 of General Settings), which will be executed in the following game step.

Deterministic agent is an agent driven by deterministic algorithm. The agent simply searches for the food randomly based on the unobserved area. Once the food is spotted (i.e. in the field of vision) the agent goes directly to it using A* algorithm [HNR68]. This agent uses incremental vision, which mean it does not forget what it saw but it just updates the new explored area.

3.1.4 Obstacles

In Tomasello experiments (see Figure 1) there were walls that limited chimpanzee's vision, which is the main purpose for the obstacles in our environment. Obstacles have the following features:

- Block agents' movement
- Block agents' vision
- Seeing through them could be allowed for special cases, but those cases are not included in the current research.

We also added a feature for the environment to accept shapes for obstacles by passing an array of zeros with ones where there is obstacle block.

3.1.5 Foods

Third and last element in the environment is food, which is just an encapsulation for the main reward. The environment allows for multiple foods based on the desired experiment.

3.2 Training Settings

In generating the environment we used the probability masks as shown in the Figure 10 to intialize the different elements in the environment. The colors represent different elements in the environment. We also need to notice that all the elements occupy a single cell except for the obstacles which occupy 4 vertical cells. For the movement of the dominant agent we typically use the deterministic algorithm explained in the section 3.1.3

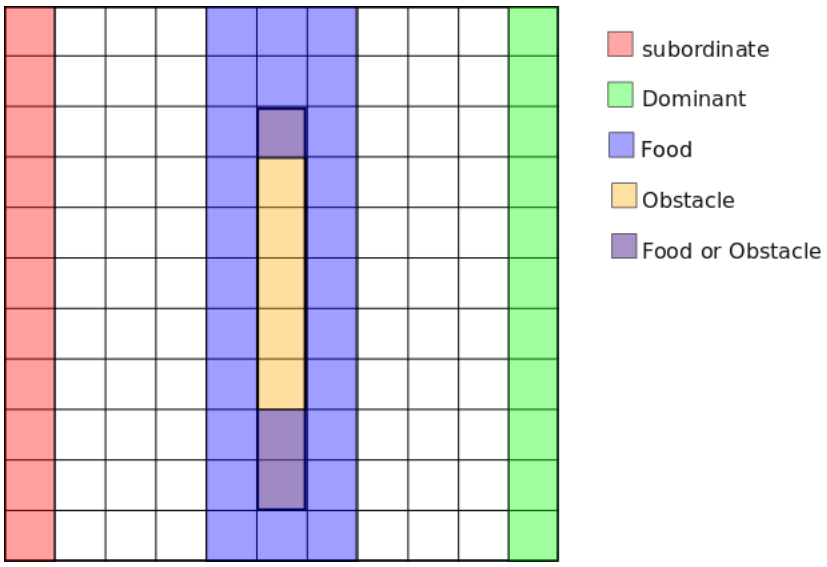


Figure 10. Possible places for each element in the environment.

In our experiments dominant agent A_{dom} and subordinate agent A_{sub} have the same vision angle 180°. Both agents have unlimited vision range. Agents and foods were treated as transparent in vision perspective. So any agent could see through other agents and food but not through obstacles.

3.3 Network Model

Before we state the architecture we used, we need to explain the basics of Dueling Networks (DN) [WdFL15]. Dueling networks will be the base for our models.

3.3.1 Dueling network

Dueling network is a neural network architecture which have been shown to perform better than DQNs in many Atari 2600 games [WdFL15]. The network architecture can be seen in Figure 11. The stream separation allow the network to intuitively learn how state is good without the need to learn about actions effect on each state [WdFL15]. This

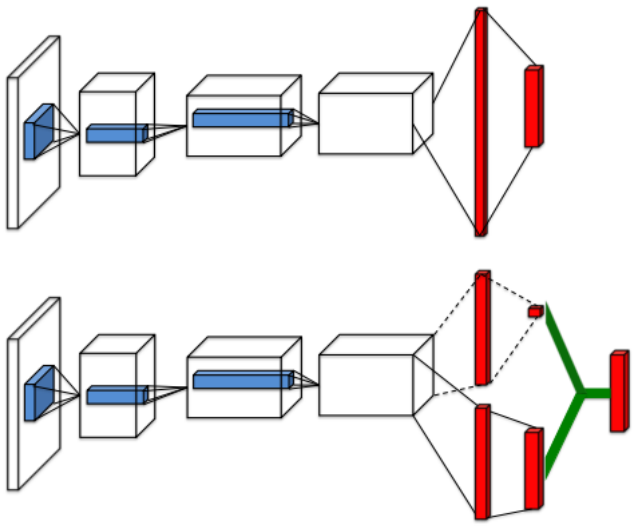


Figure 11. Q-network with single stream on top, compared to double stream (Dueling network) bottom.

architecture has two streams, one responsible for calculating the value function (V), the other for calculating a vector representing the advantage (A, as defined by Equation 5) from taking each possible action [WdFL15].

$$A(s,a) \equiv Q(s,a) - V(s) \tag{5}$$

The action-value (Q) is estimated from those two streams using either naive, average or max approaches described in equations (6), (7), (8), respectively. The two streams networks parameters are encoded in α, β symbols.

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha) \tag{6}$$

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|A|}\sum_{a'} A(s,a';\theta,\alpha) \right) \tag{7}$$

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \max_{a' \in |A|} A(s,a';\theta,\alpha) \right) \tag{8}$$

3.3.2 Architecture

The model we used is inspired by dueling network [WdFL15] but instead of splitting the last layer between advantage and value we used the full layer for value, and the advantage. Figure 12 shows the difference between both models. We need to mention that for our task we did not use convolutional neural networks since we did not use images as input but binary data. Feeding binary data allowed us to decrease the complexity and focus on the task of learning the perspective taking task.

For architecture programming we used `Keras` library [C+15].

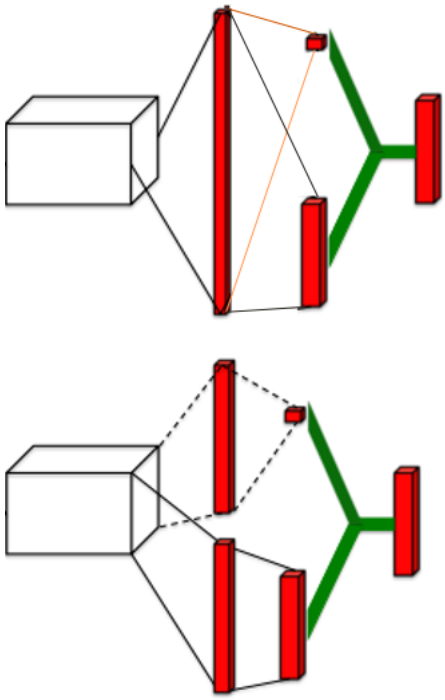


Figure 12. The difference between dueling network (bottom) and the model we used (top), is that our model use all the previous layer for V and A while the original only use half of it. Figure modified from [WdFL15].

3.3.3 Network Input/Output

We wanted to make the input as simple as possible, so we used binary features as input. The input consisted of a concatenation of different hot maps, and vectors representing different attributes. We can distinguish the following inputs:

- **Agent map:** where the network receives an input of the exact size of the world and there is a "1" at the position where the agent is located and a "0" otherwise. See Figure 13 (e).
- **Agent orientation:** one hot vector represents the direction the agent is looking at.
- **Food map:** this map contains "1" at the food position, if observed, and "0" everywhere else. Illustrated on Figure 13 (d).
- **Obstacles map:**contains "1" in the blocks occupied by obstacles and "0" everywhere else. See Figure 13 (g).
- **Observed map:** observed area for the agent will have "1" and "0" at the rest. Figure 13 (c).
- **Other agents map:** for each other agent the network receives a hot map with "1" at that agent place if observed. Figure 13 (f).
- **Other agents orientation :** for each other agent in the world the agent receives a hot map with "1" to indicate another agent orientation, if observed.
- **Actions history:** there were experiments in which we added the last n actions performed by the agent. Each action was represented by one hot vector.

Figure 13 shows an example of some maps that form part of the input received by the network.

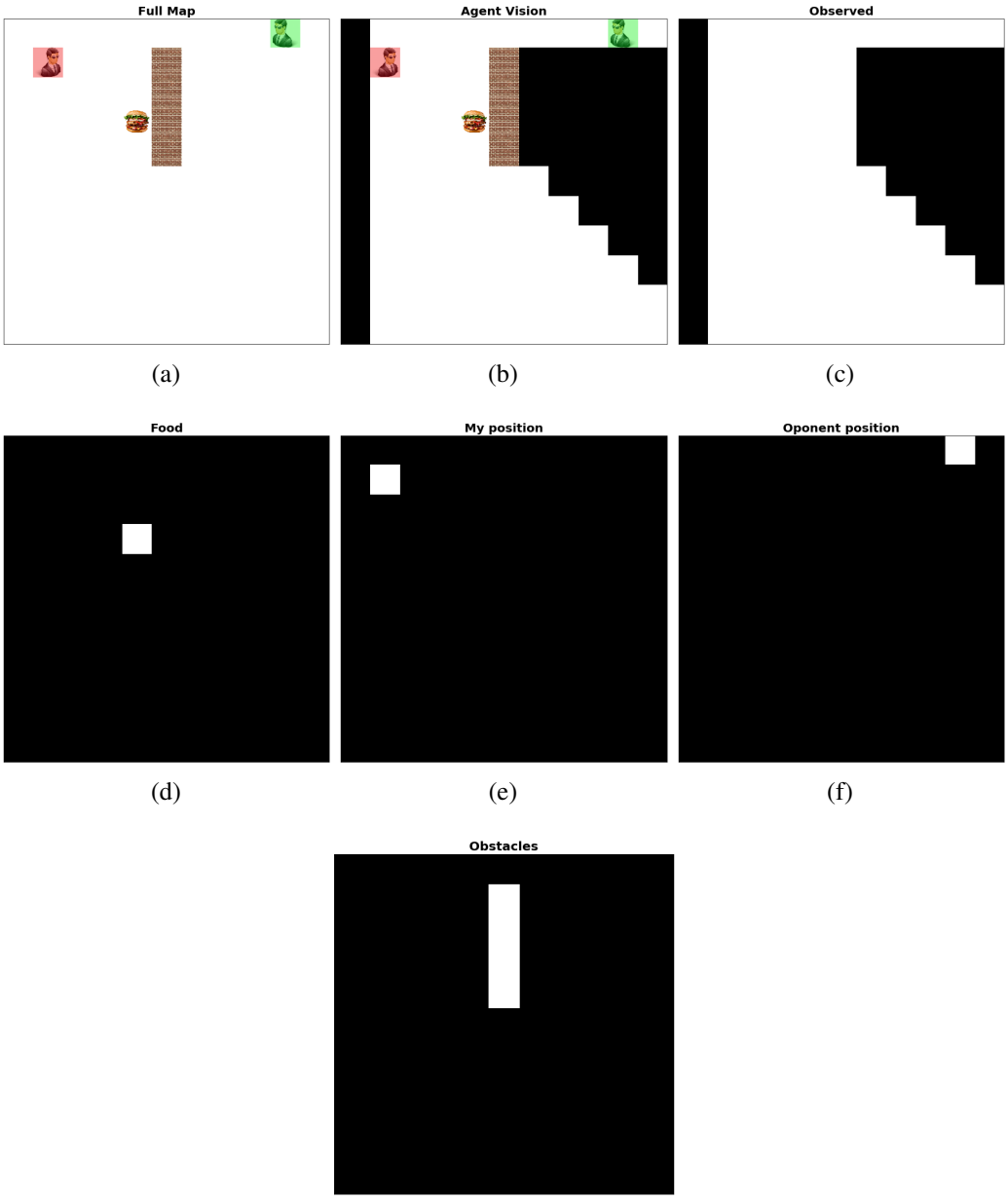


Figure 13. Example of maps forming the neural network input. (c) to (g) are represented in binary where white is True while (a) and (b) are not provided to the agent and given here just for clarity. (a) shows the full map of the environment at some state. (b) shows the agent vision at same state using 180° vision. (c) is the observed layer. (d) is the food position layer. (e) the agent position map. (f) represents the opponent position. (g) is the obstacle position layer.

The network output was a float vector of the same length as the possible number of actions and representing the action values. In all experiments described in Results section we restrict the set of actions to 4 movements and the no-ops action (described in item 9 in the section on General Settings).

3.3.4 Replay Memory

The replay memory or experience replay is a set of transitions $(s_t, r_t, a_t, s_{t+1}, terminal)$ which is used to train the network. Replay memory is a common practice when deep learning is used for RL tasks [WdFL15] [MKS+15] [MKS+13].

Each time step we trained the network with 32 randomly selected samples from the replay memory. We used a replay memory of size 100K. Once the replay memory is full oldest samples would be replaced.

3.3.5 Metrics for model selection

To compare the performance of different trained models we used the subordinate agent A_{sub} reward and steps. Steps represent the required number of time steps for the game to terminate. We used the average reward and average steps in two ways: firstly, we averaged the reward and steps over all the episodes in the experiment. Secondly, we used only the average reward and steps of the last episodes in the experiment were the exploration rate had reached the minimum allowed value or zero. While the first allowed us to estimate the overall performance and learning speed, the second showed us how good the model became when it mostly depended on its own.

3.4 Curriculum learning

A child does not start running immediately he needs to learn to flip, crawl, balance, stand, ...etc until he can run. Similarly, curriculum learning consist of training an agent on a sequence of tasks of increased complexity to improve the speed of learning in a more complex task [Nar16]. Putting A_{sub} to learn to directly accomplishing the task with dominant agent A_{dom} will not help A_{sub} to learn. This is because A_{dom} is using an efficient deterministic algorithm that will not give A_{sub} the chance to explore and learn. Subordinate needs to learn to find his way to food before learning to compete or avoid A_{dom}. This is the reason for the existence of the following tasks of learning. We have to note that A_{sub} has no clue to understand that the other is the dominant agent except from the reward it gets.

3.4.1 Task 1: Finding the food

In this stage the only task A_{sub} had was to find the food and eat it. This task might look easy but when we consider the vision and the obstacles it gets more complicated. Without a memory the agent has to act depending on the immediate vision of field. This stage optimized A_{sub} to look for the food and approach it. To make the task reasonable the obstacle will spawn randomly in specific region as shown in Figure 10. In this stage the dominant agent's input exists as zeros only. The reward for this stage is shown in Table 1. The network hyper-parameters were tuned in this stage and afterward those parameters kept the same.

Operation	**Reward**
Time step	-0.1
Eating food	1000

Table 1. The reward scheme for task 1 & 2

3.4.2 Task 2: Competing for the food

While in the previous task the subordinate A_{sub} aim is to find the food only, here A_{sub} needs to find the food before the dominant agent although there is no punishment from the A_{dom}. The same reward strategy applied in previous stage is applied here. For this stage we wanted the agent to optimize its road to the food with the existence of the dominant agent.

3.4.3 Task 3: Avoiding dominant agent

Finally, we trained the subordinate agent with a punishing dominant agent. The dominant gave -100 reward points for the subordinate whenever the subordinate was 1 or 2 steps away [1] from the dominant.

Operation	**Reward**
Time step	-0.1
Eating food	1000
punishment	-100

Table 2. The reward scheme for task 3

[1] both experiments were conducted separately.

3.4.4 Curriculum paths

For the three tasks there were two paths we followed:

- Task1 $\rightarrow$ Task3: here we trained the agent first to find the food and then moved to task 3.
- Task1 $\rightarrow$ Task 2 $\rightarrow$ Task 3: in this path A_{sub} is trained over all the three tasks.

3.4.5 Reward scheme

We used a reward scheme to grossly mimic some of the incentives found in the situations we are aiming to model. The agent has three types of reward,

- First, the agent receives a -0.1 per time step no matter what action he chose. This time step punishment adds to the discount reward to improve the speed of finding the food.
- Second, the agent receives a reward of value 1000 when he gets the food. This value was chosen to make a balanced reward system in the end, so getting a 1000 points of reward by step 1000 will give around 0.1 reward to the first step taken in that trajectory; this value is calculated for a discount rate of $\gamma = 0.99$.
- Third, the subordinate agent gets a punishment of -100 each time he enter the dominant agent control range.

3.5 Behavioral test cases

To check the subordinate behavior we created 12 test cases. We prevented dominant agent from moving in those test cases, although it was moving during training. The cases in which the dominant agent is static during test reveal more clearly on what the agent depends to take a decision to go for the food or not. Figures 14 to 17 illustrate the 12 cases considered which we briefly describe below.

- **Test case 1:** in this test the subordinate agent A_{sub} optimally supposed to go for the food since the food is unobserved by the dominant agent A_{dom}. In same time A_{sub} cannot see A_{dom}. Environment in Figure 14 (a,b,c)
- **Test case 2:** the target behavior from A_{sub} is to avoid the food as the dominant agent A_{dom} is visible and can see the food. Environment in Figure 14 (d,e,f)
- **Test case 3:** A_{sub} supposed to go for the food in case A_{dom} was not spotted in its field of vision before eating the food. Environment in Figure 14 (g,h,i)

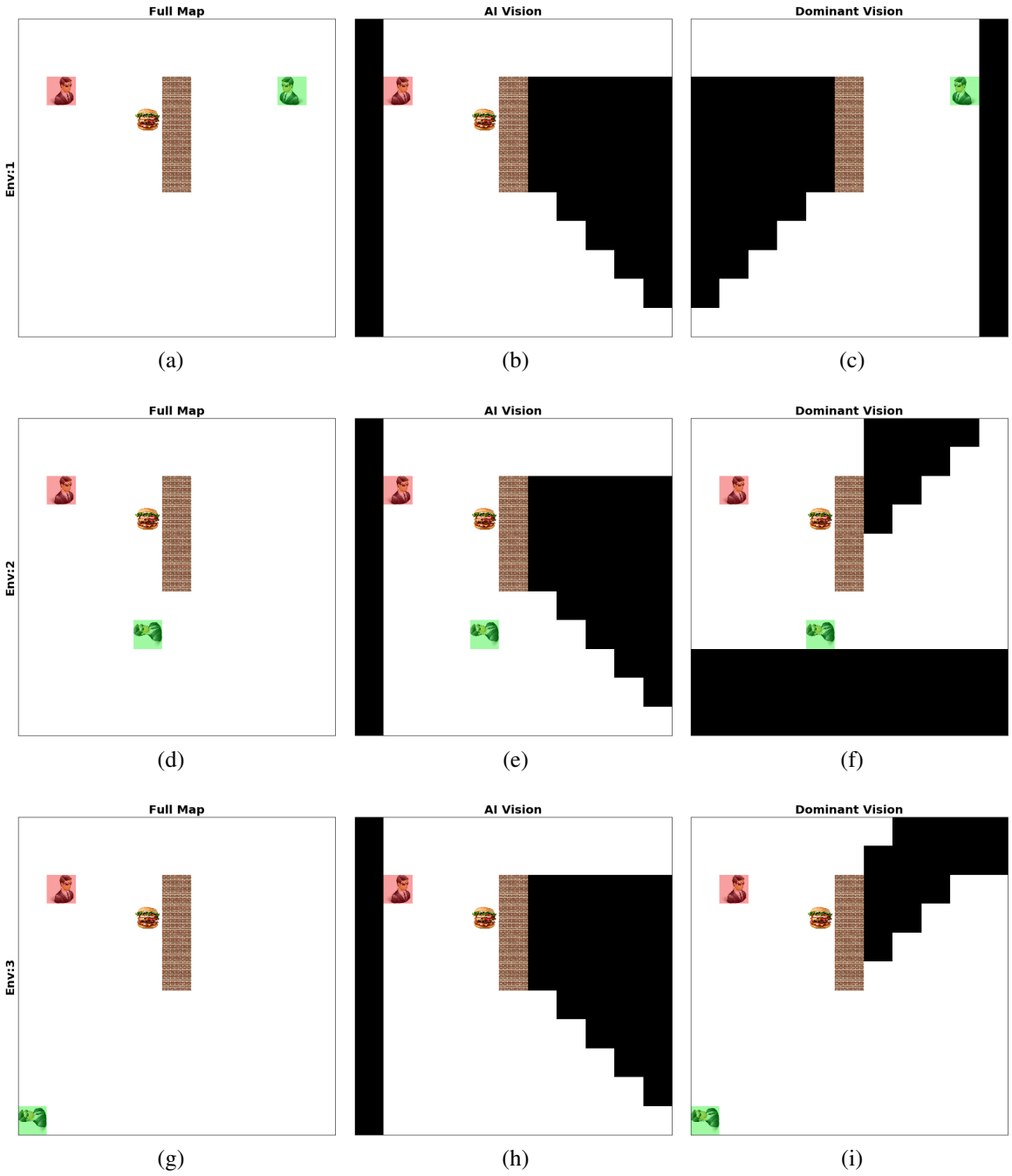


Figure 14. Test case 1 (a,b,c). Test case 2 (d,e,f). Test case 3 (g,h,i)

- **Test case 4:** A_{sub} and A_{dom} can see each other while only A_{sub} can see the food. A_{sub} supposed to go for the food since the dominant does not see the food.Environment in Figure 15 (a,b,c)
- **Test case 5:** only A_{dom} can see the food and both agents cannot see each other. A_{sub} is not supposed to go for the food as it cannot see the food and the food is under the dominant watch.Environment in Figure 15 (d,e,f)
- **Test case 6:** agents can see the food but not each other. A_{sub} should go to food as long as A_{dom} is not spotted before taking the last action.Environment in Figure 15 (g,h,i)
- **Test case 7:** both agents can see each other but not the food. The ideal behavior in this case is to explore.Environment in Figure 16 (a,b,c)
- **Test case 8:** agents can see each other but only A_{sub} can see the food. Similar case to Figure 15 (a,b,c) but A_{dom} here is closer to food. Environment in Figure 16 (d,e,f).
- **Test case 9:** A_{sub} can see both food and A_{dom} while the last cannot see neither A_{sub} nor the food. A_{sub} supposed to go for the food as the dominant is looking away.Environment in Figure 16 (g,h,i). 16 (a,b,c).
- **Test case 10:** A_{sub} see all other elements while A_{dom} does not.A_{sub} supposed to go for the food.Environment in Figure 17 (a,b,c)
- **Test case 11:** both agents can see all other elements in unique positions. This test is an exploration test since this case has never been seen while training. The subordinate is not supposed to go for the food. Environment in Figure 17 (d,e,f)
- **Test case 12:** unique test where the obstacle does not exist. A_{sub} is not supposed to get the food since A_{dom} in the field of vision, and A_{dom} can see both A_{sub} and the food. Environment in Figure 17 (g,h,i)

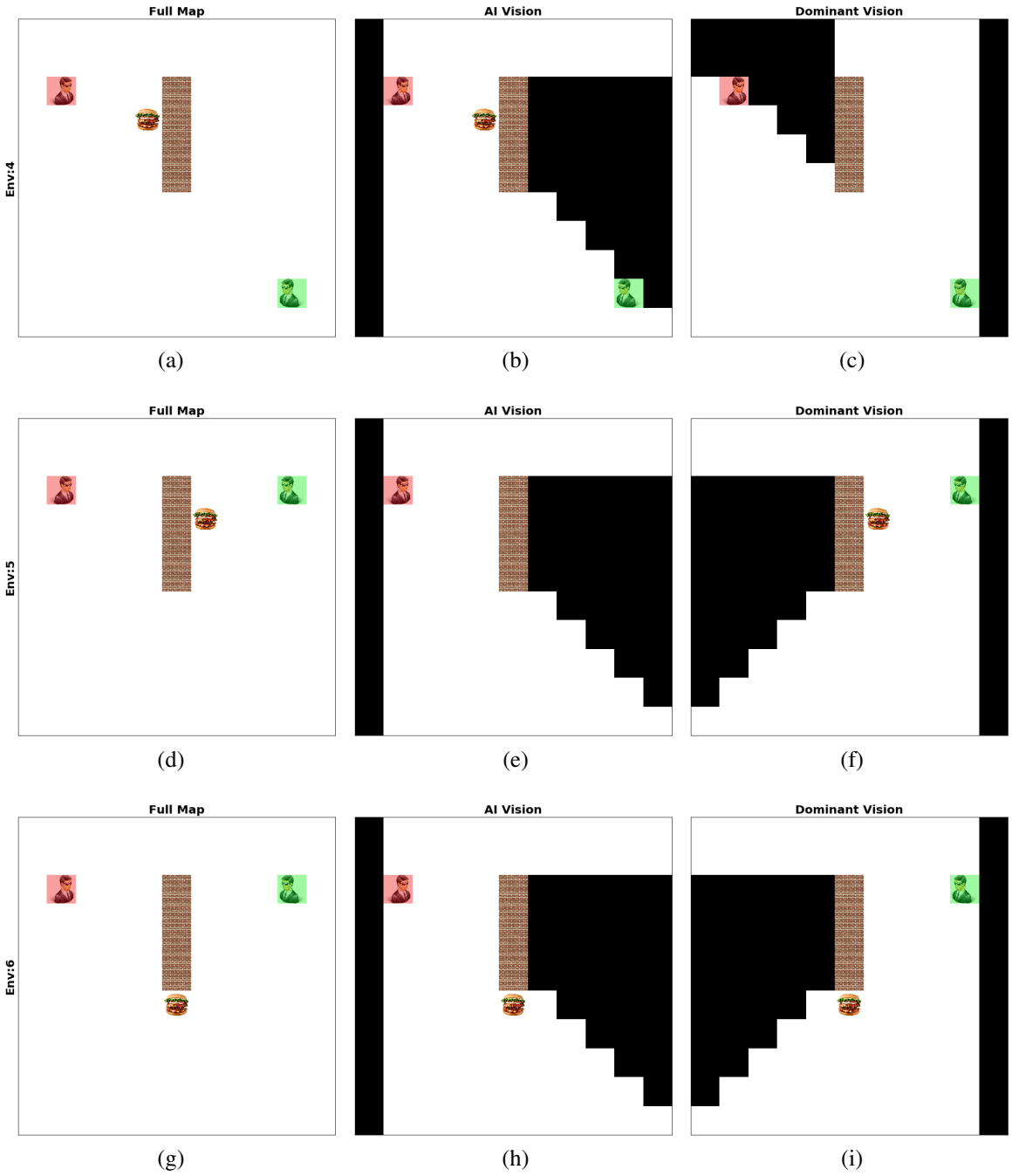


Figure 15. Test case 4 (a,b,c). Test case 5 (d,e,f). Test case 6 (g,h,i)

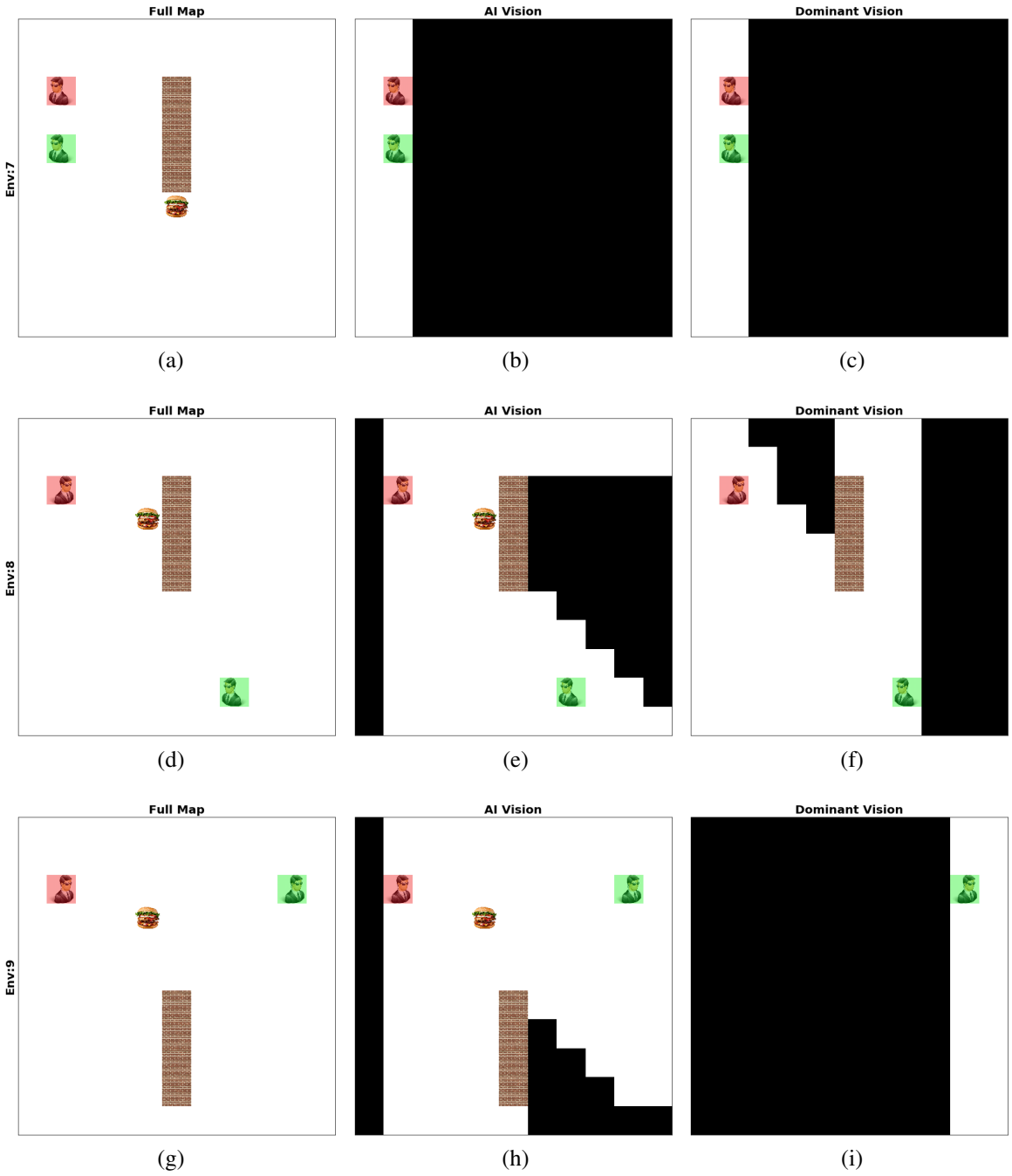


Figure 16. Test case 7 (a,b,c). Test case 8 (d,e,f). Test case 9 (g,h,i)

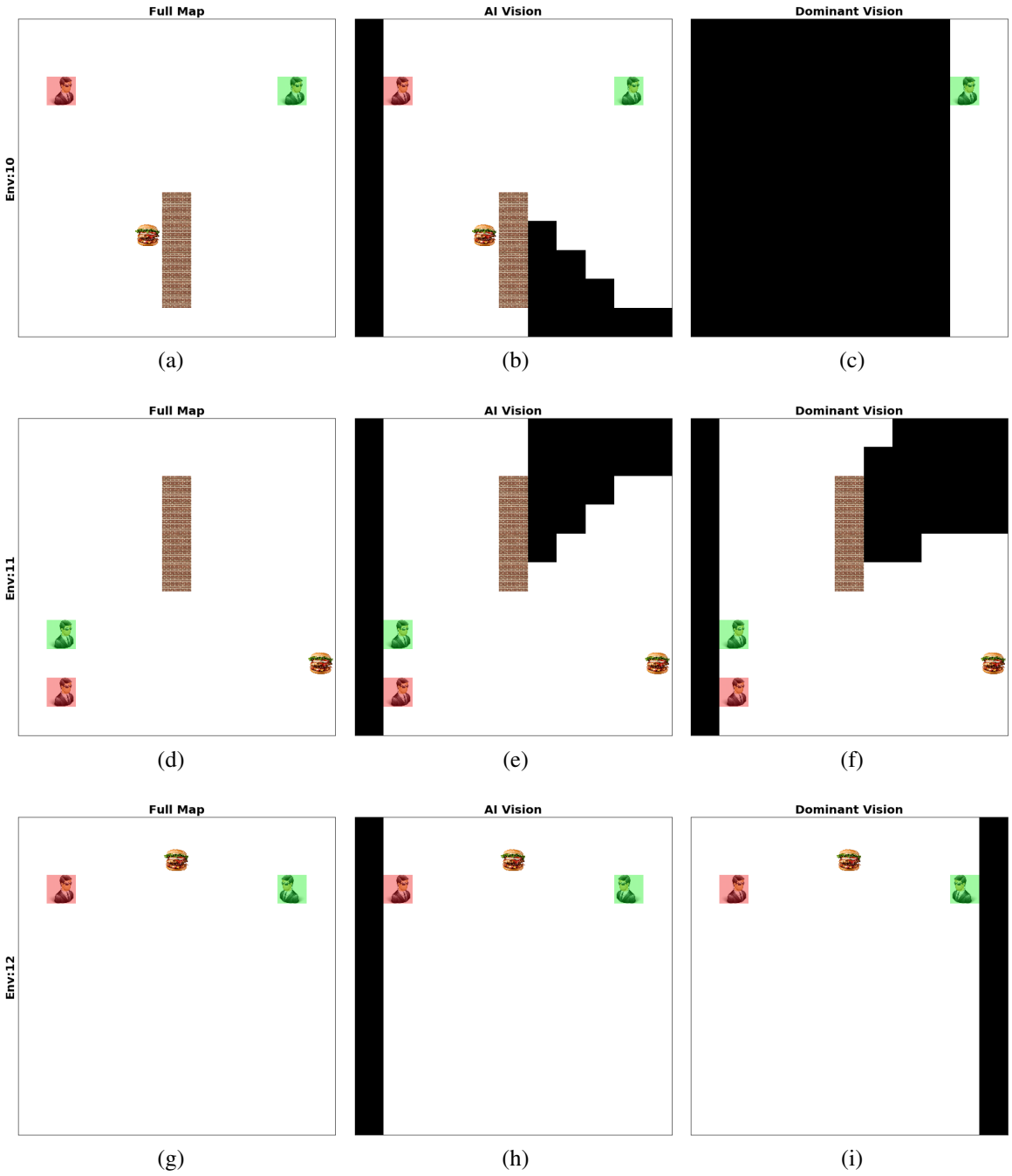


Figure 17. Test case 10 (a,b,c). Test case 11 (d,e,f). Test case 12 (g,h,i)

4 Results

Overall we conducted more than 750 experiments that consumed around 1 year of computation power. Those experiments built around 1500 models between training and target networks. With those models we launched around 15K simulations over the tests we defined in the Section 3.5 `Testing agent behavior`. Next we give an overview about the environment that allowed those simulations and.

4.1 Environment simulator

The simulator that we developed, allowed us to design environments satisfy all of our experiments current needs. An example of the environment shown in Figure 18. The

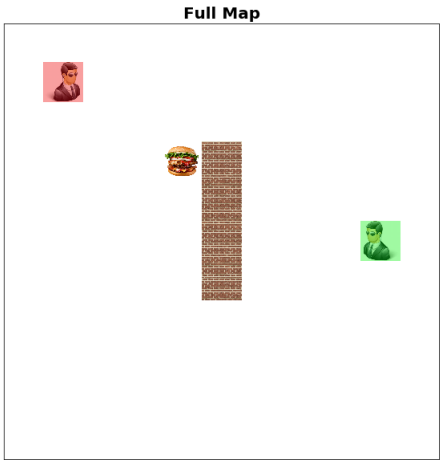


Figure 18. Example image of the environment

simulator have the following capabilities:

- Allow agents to have different power rank, vision angle, vision range, vision type (alter-centric or ego-centric) and control range.
- Adding walls can be done by shape or block.
- Possibility of passing a distribution matrix for each element.
- Ability for elements of the environments to have the property to block agents vision.
- Allow multiple foods in same experiment with each have different reward. Food can have control range as well to give reward when an agent inside that range.

- Ability to use user specific function for reward or to calculate the agent's output.
- Opportunity to have multiple obstacles like walls or water

4.2 Hyper parameter search and network tuning

We tuned the hyper-parameter of the controlling network while the agent in training in the task of food finding (Task 1 as described in Section 3.4.1). We chose Task 1 because it is relatively easier compared to the other two tasks. The following list explains the hyper-parameters we scanned.

- **Replay memory:** integer representing the maximum number of transitions stored to be sampled to train the network.
- **Number of hidden layers:** integer representing the number of hidden layers used.
- **Number of nodes:** integer marking the number of the nodes in every hidden layer.
- τ: float number determining how much we copied every time step from the training network parameters θ to the target network $theta^{-}$.
- **Advantage:** string that can be `naive, avg, max` each representing using one of the equations (6), (7), 8) for estimating the Q-value.
- **Activation function:** string to select the activation function `relu`[HSM+00] or `tanh` (hyperbolic tangent).
- **Batch size:** integer determining how many samples we get from replay memory to train the training network with every step.
- **Exploration:** float number between [0,1] identifying the probability of selecting a random action for A_{sub}.
- **Train repeat:** integer to limit how many times we train the network every step.

During the tuning we explored many values for these parameters. Typically we run grid search procedure for different combinations of pairs of parameters. The range of values and options scanned are summarized in Table 3.

In training, two policies of exploration were used. One was to use a constant small exploration rate. Another was to use decreasing exploration which either ended at 0 or 0.1. Importantly, in around half of our experiments we kept an exploration rate of 0.05 at testing or validation. This is a common practice to avoid the loopy behavior (repetition of same actions that form a loop in the end) that might converge in some states.

Choosing the best hyper-parameters was based on the training average reward as explained in Subsection 3.3.5 Metrics. We scanned the parameters using DQN update

Parameter	Tested values
Replay memory size(RM)	[100000, 200000, 300000, 400000, 500000]
#layers	[1, 2, 3, 5, 10]
#hidden nodes(HN)	[20, 32, 40, 64, 100, 128]
τ	[0.0001, 0.001, 0.1, 0.01]
Advantage	[`naive, avg, max`]
Activation	[`relu, tanh`]
Batch size(BZ)	[10, 16, 32, 64, 128, 256]
Exploration	[0.001, 0.01, 0.02, 0.1, 1.]
Train repeat(TR)	[1, 2, 4, 8, 16]

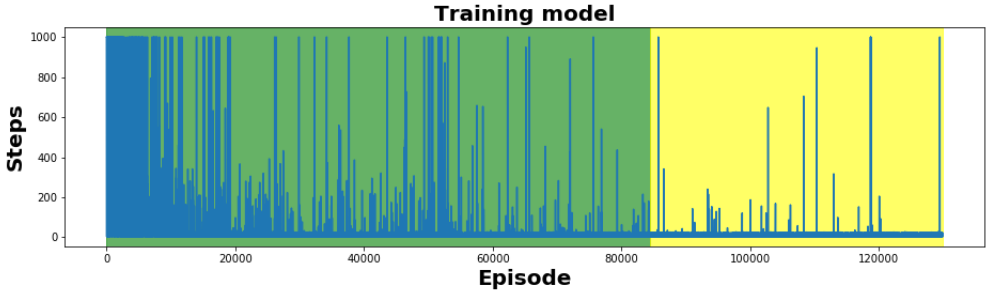
Table 3. The values we scanned for each hyper-parameter

rule and our version of Dueling Network. The best hyper-parameters were found on experiments 110, 201-207, and 349-355. Table 4 contains the shared hyper-parameters for those models.

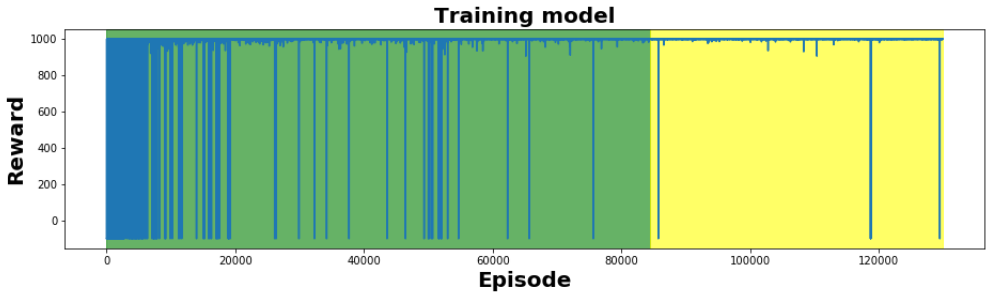
RM	#layers	HN	τ	Advantage	Activation	ϵ	BZ	TR
100K	1	100	0.001	`max`	`tanh`	0.01	32	1

Table 4. Best hyper-parameters

Those hyper-parameters performed the best over multiple random seeds for the network training and we used them for rest of the experiments. The best model at this stage was model 355 with average training reward of `996.15` knowing that the best possible reward is `1000`. In Figure 19 (top) we can see how the steps (number of steps needed before reaching the food) decreased over the training to stabilize in the last quarter. In same Figure 19 (bottom) we see the obtained reward in the episodes and how the reward get stable pretty early. The green part represent the training with exploration rate (possibility of picking random action not from the neural network) $\epsilon > 0$ while at the yellow part $\epsilon = 0$.



(a)



(b)

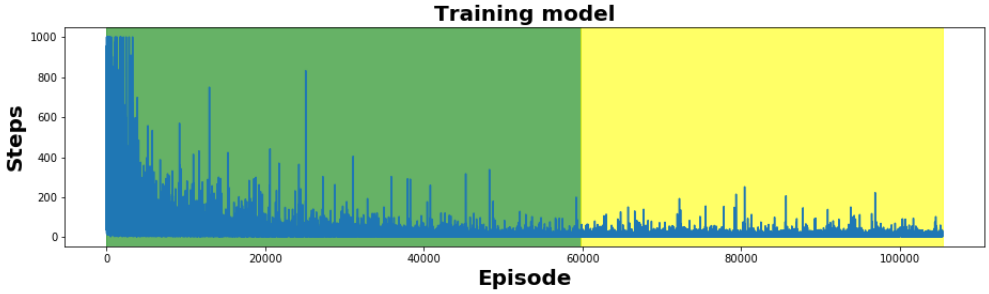
Figure 19. Experiment 355. (a) Steps per episode and how steps decrease by over episodes. (b) Reward per episode and how it increase to almost perfect. Green area represents episodes where $\epsilon > 0$ while yellow area represents $\epsilon = 0$

4.3 Curriculum learning toward perspective taking task

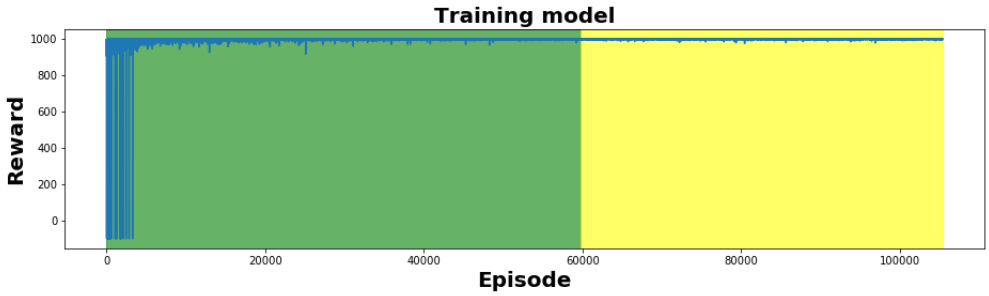
As noted previously for the agent to learn to compete against the other agent it first needs to learn simpler tasks. We show those steps in the next subsections.

4.3.1 Task 1: Finding the food

After stabilizing the hyper-parameters, different approaches were taken. We summarize those approaches in update rules of the network (DQN Equations 2 3, DDQN Equation 4) and adding last n actions to the input. The best average training reward achieved in this task taking account of the new approaches was 997.75 in experiment 549 and an average of 18.96 steps. This model used our version of dueling network shown in Figure 12 with update rule DQN Equation 3. The model also incorporate the last 5 actions taken by the agent in its input. We can notice in Figure 20 (top) that steps improved and the plot become much less noisy than previous model. In same figure (bottom) we can notice how the model converged pretty fast from reward gain and stability.



(a)



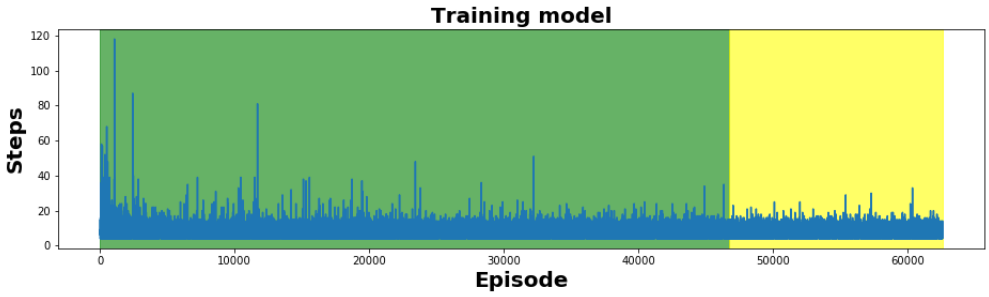
(b)

Figure 20. Experiment 549. (a) Steps per episode were can see how the needed amount of step kept decreasing to stabilize later. (b) Reward per episode where we can see that reward improved pretty fast and kept improving over to take almost steady line later. Green represent episodes where $\epsilon > 0$. Yellow represent $\epsilon = 0.1$

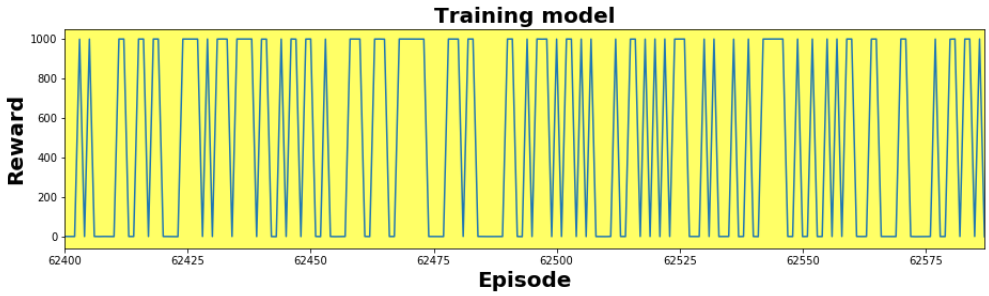
4.3.2 Task 2: Competing for the food

In this stage the agent already knows the way to food but needs to compete with another agent so it is not only important to find the food but to find it before the other agent. The dominant agent acts competitively in this task without punishing the subordinate. The dominant is equipped with a deterministic algorithm to explore the environment and once it sees the food it goes immediately to it in the shortest path. Experiment 604 was the best by achieving average training reward of 401. The reward decreased due the competition provided by A_{dom}. This model used DDQN update rule Equation 4 with our version of dueling in addition to the last 4 actions as additional input. Figure 21 clearly shows the general decrease in the required steps from the axis scale change. The irregular jumps in the number of steps is due to interference between A_{dom} and A_{sub} where subordinate block the dominant way (dominant will not try to avoid the subordinate if subordinate blocked the way). For the reward plot (Figure 21 bottom) we limited it to the last 200 episodes to avoid the zigzag plot. We can notice that reward was changing between dominant,subordinate based on how much the subordinate is

optimized and how close the food to the dominant.



(a)

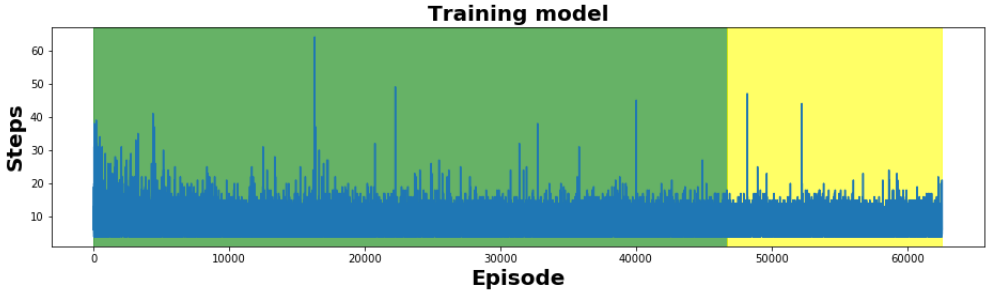


(b)

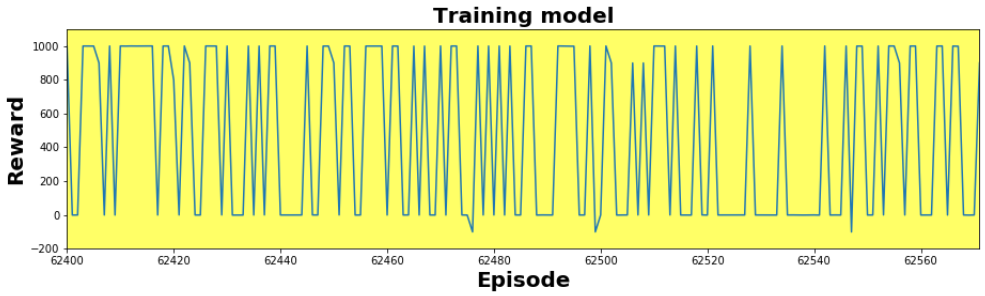
Figure 21. Experiment 604. (a) Steps per episode, not much improvement overtime for the steps. (b) Rewards per episode for last ~ 200 episodes which shows that subordinate either take almost full reward or around 0. Green represent episodes where $\epsilon > 0$. Yellow represents $\epsilon = 0.1$

4.3.3 Task 3: Avoiding the dominant agent

The final stage where the subordinate agent A_{sub} can get punished by the dominant A_{dom} if it was 1 or two blocks away from the dominant. The curriculum based on Task1 (finding food) $\rightarrow$ Task 2 (compete with dominant)$\rightarrow$ Task 3 (avoid dominant) did not success by means of average training. Experiment 609 achieved the best average training reward of 388.1. This model used DDQN Equation 4 with last 4 actions as additional input and evolved from task1$\rightarrow$ task3 curriculum path. Figure 22 shows the steps and last ~ 200 episodes reward. The steps plot (top) shows that the subordinate avoided blocking the dominant more compared to previous task. While for reward plot (bottom) we can notice some negative results or non-perfect reward due to interaction with dominant.



(a)



(b)

Figure 22. Experiment 609. (a) Steps per episode. (b) Reward per episode for last ~ 200 episodes, we can notice some negative rewards or non-perfect reward explained by interaction with dominant. Green represent episodes where $\epsilon > 0$. Yellow represents $\epsilon = 0.1$

4.4 Testing agent's behavior in perspective taking tasks

After training the models on the curriculum task we assessed the performance of the agents in the perspective taking tasks with the specific designed tests. The tests are shown in Table 5 with the desired behavior that an agent with perspective taking capability should show. We ran those 12 cases on 614 models from 307 experiments. Only 11 models passed those test cases. We picked the model that achieved the tasks with minimum steps since that means the behavior is more embedded in the network than others. The chosen model was the target model (which copy 0.001 or τ from training model as explained in DQN subsection 2.3.1) of experiment 599.

- **Test case 1:** the subordinate A_{sub} went for the food in 6 steps. The followed path was optimal and A_{sub} did not see the dominant. For this test case the agent achieved the expected behavior.

- **Test case 2:** A_{sub} avoided the food although the food is in the same position as `Test case 1`. The only difference is the dominant position and direction. Here

Test case	Thumbnail	Desired behavior toward food
1		go
2		not go
3		go
4		go
5		not go
6		exploration
7		not go
8		go
9		go
10		go
11		not go
12		not go

Table 5. The values we scanned for each hyper-parameter

the subordinate stayed in position where the food and dominant stayed in its field of view. The concluded behavior is also what one would expect from an agent with perspective taking ability.

- **Test case 3:** A_{sub} went for the food, although the dominant is there and able to see everything, but the subordinate never witnessed A_{dom} until the last move of eating the food. This behavior is also expected to happen.
- **Test case 4:** in this case the subordinate reached for the food. Although the

dominant was looking toward the subordinate, the food was not visible for the dominant. This behavior matches what an agent with perspective taking would do.

- **Test case 5:** A_{sub} did not manage to see the food in this experiment. We think this occurred because whenever the food was on the other side the dominant was always faster to get it. A loopy behavior is noticed as well. We believe that a normal agent with perspective taking should look for food and avoid it only if it is under dominant surveillance. Although we would argue that if we do not see the food and cannot see the dominant, then it is probably better not to try even. Specially if we know that we have never got the food on the other side, or it never happened to get the food from the other side.

- **Test case 6:** A_{sub} avoided the food although the dominant was not spotted in its field of view. It might be the case that food position is dangerous from previous experience while training. The same loopy behavior from previous test case is also spotted here.

- **Test case 7:** A_{sub} did not go for the food and was in the same loopy behavior as test cases (5,6).

- **Test case 8:** this test case is identical to test case 4 with the change in the distance between the dominant and the food. The subordinate went for the food in this experiment which is a behavior to expect from an agent capable of perspective taking ability.

- **Test case 9:** subordinate went for the food in this test case. Since the food was not spotted by the dominant, it is plausible to go for the food.

- **Test case 10:** in this case we changed the food place and the obstacle where they are not seen by dominant. The subordinate retrieved the food successfully which is a behavior coincide with perspective knowledge.

- **Test case 11:** in this case the agent did not go for the food which in principle fits with an agent with perspective taking. However, since this case is never seen by the network, we can not be sure about the reson behind the behavior.

- **Test case 12:** the agent went toward the food to wander in a loopy behavior under it. The behavior of not eating the food suits an agent with perspective taking capabilities. Although wandering around the food would be suspicious behavior.

5 Discussion

Perspective taking is the ability to take the point of view of another. This faculty is essential to navigate in many social interactions and it has been found in other animals including chimpanzees. Thus, it is of great interest to equip machine and agents with similar abilities. Our aim was to train a model using artificial neural networks and reinforcement learning,to able to perform perspective taking tasks. To do so we built the APES environment which allow us to define tests to match to some extent Tomasello experiments [HCAT00] on chimpanzees. In the following we will review and discuss the results obtained.

5.1 Summary

Using our APES environment we trained many models by different update rules, environment states representations, and different curriculum learning paths. The best networks performed well in finding the food task approaching nearly optimal trajectory toward the food. The best models ended with positive reward or finding the food in $\sim 40\%$ of episodes in Task 2 (compete with dominant) and Task 3 (avoiding the dominant).

From the trained models only 11 managed to achieve results that would be also achieved by an agent with perspective taking capabilities on the defined 12 test cases. When the favored behavior was toward approaching the food like in Test cases (1, 3, 4, 8, 9, 10), the subordinate managed to approach it in almost optimal time steps.

When the subordinate should avoid the food, he does avoid the food while performing a repetitive behavior. We would argue that the loopy behavior in Test cases (5,6,7) might occur because of the food position is closer to the dominant. In such cases, during training the dominant agent would have obtained the food item. This means that even if the subordinate managed to reach to food area it would get punished by the dominant or in best cases not rewarded. This tension between obtaining the food and avoiding the nearby dominant might result in the repetitive movement of the subordinate in a region away from the dominant agent.

We consider that the learned behavior is a first promising step toward equipping agents with some visual perspective taking abilities. However, much more stringent tests are needed to evaluate the performance of the agents in this task. It is also important to open the network controlling the agent to see which representation is used to learn this behavior, but that is probably a topic for another thesis.

5.2 Related work

Although we and [BJEST17] share the same aim to reverse-engineer from computation to human mind, we took different roads. The work in "Rational quantitative attribution of beliefs, desires and percepts in human mentalizing"[BJEST17] used Bayesian inference

while we used artificial neural networks. The environments and the specific task were different as well.

Another research [TMK+17] studied the interaction between multiple agents using deep reinforcement learning. In particular, they focused on the relation between two agents playing pong depending on competitive and cooperative incentives. However, in all cases the environment was fully observable for both agents.

Recently Microsoft launched an artificial intelligence competition [mal]. The competition aims for an AI agent that can collaborate with humans. For the collaboration to succeed AI should learn to understand human intent, an ability that typically requires perspective taking capabilities.

5.3 Limitations

5.3.1 Environment

Although the environment is extremely generalizable, it still has some limitations:

- Two elements cannot co-exist on the same cell. For example, the agent cannot hold the food and move it.
- Performance is not optimal. For example handling agent field of view is single threaded which limit the processing speed.

5.3.2 Perspective taking task

The neural network controller and the task contained also important limitations for our goal of learning visual perspective taking:

- The agent does not have an explicit memory to remember the previous states.
- The variability in the initial locations of the agents and other items is limited to some ranges. While the variability during training helps to generalize over unseen situations, too much variability would have probably implied much longer training sequences.

5.4 Future work

5.4.1 Environment

For future we plan to:

- Improve the speed of the pipeline by using multi-threading and optimizing the field of view algorithm.

- Add more features including novel items with different properties and to allow elements co-exist on same cell.

5.4.2 Perspective taking task

We envision a large amount of future directions and work to continue this project.

- Add memory for the agent to remember previous states. More generally, implement and experiment with other neural architectures that exploit different types of memory.
- Model more complex aspects of Theory of mind, including the inference of what other agents know (e.g. information another agent holds about an item that is currently out its view).
- Model Tomasello experiments [HCAT00] with all agents being controlled by learning neural networks.
- Test the learning more complicated behaviors, such as the ability of multiple subordinate agents to cooperate to take over a dominant agent.
- Experiments with tasks involving more than one piece of food or reward.
- Experiment with different reward schemes (e.g. punishing the subordinate for simply eating food in the field of vision of the dominant).
- Study the representations learned by successful networks.

6 Conclusion

In the previous chapters we took a look to some reinforcement learning experiments conducted in the new APES environment. This environment was designed from scratch to simulate biological environments in Tomasello experiments [HCAT00]. Those experiments showed how subordinate chimpanzees ate the food unobserved by the dominant chimpanzee. That means subordinate inferred what dominant can and cannot see. To simulate subordinate behavior in computational agents we trained different reinforcement learning algorithms supported by artificial neural networks on APES environment.

The results showed many models with tendency toward the desired behavior, which is to avoid the food spotted by dominant. We are currently working on embedding a memory within the agent and other improvements on the environment. We are looking toward doing more tests and experiments in different contexts, such as how would two subordinates cooperate effectively to get over the dominant, or which representations the networks learned while solving these tasks.

7 Acknowledgment

Thank to almighty Allah for given me the grace to start and continue till this moment of writing, in this study.

Am thankful to my family, which made me who I am now.

Thanks to Raul Vicente, for hosting me in Neuroscience lab while working on my thesis for the past year. His patience, guidance, suggestions, comments are what made this thesis written.

Thanks to Jaan Aru, for insightful advises, and dedication. I learned a lot from you, Thanks for being there.

Thanks to Tambet and Ardi, for there tips and tricks and the nice journey and laughs I had

Thanks to Daniel, Axel and Ilya for the informative discussions.

The author of this thesis was funded by Dora plus scholarship.

The computation necessities were provided by University of Tartu High Performance Computing and Estonian Scientific Computing Infrastructure (ETAIS).

References

[ana16] Anaconda software distribution. computer software (vers. 2-2.4.0), 2016.

[App11] I. Apperly. *Mindreaders: The Cognitive Basis of "theory of Mind"*. Psychology Press, 2011.

[AS07] Francesco Amigoni and Viola Schiaffonati. Multiagent-based simulation in biology. In *Model-Based Reasoning in Science, Technology, and Medicine*, pages 179–191. Springer, 2007.

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[Ber] Björn Bergström. Fov using recursive shadowcasting - roguebasin. `http://www.roguebasin.com/index.php?title=FOV_using_recursive_shadowcasting`.

[BJEST17] Chris L Baker, Julian Jara-Ettinger, Rebecca Saxe, and Joshua B Tenenbaum. Rational quantitative attribution of beliefs, desires and percepts in human mentalizing. *Nature Human Behaviour*, 1:0064, 2017.

[C+15] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[CYS15] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 415–423, 2015.

[Dav83] Mark H Davis. Measuring individual differences in empathy: Evidence for a multidimensional approach. *Journal of personality and social psychology*, 44(1):113–126, 1983.

[DW16] Frans De Waal. Are we smart enough to know how smart animals are?, 2016.

[FF05] Chris Frith and Uta Frith. Theory of mind. *Current Biology*, 15(17):R644 – R645, 2005.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GMGW08] Adam D Galinsky, William W Maddux, Debra Gilin, and Judith B White. Why it pays to get inside the head of your opponent: The differential effects of perspective taking and empathy in negotiations. *Psychological science*, 19(4):378–384, 2008.

[Gre15] Richard L Gregory. *Eye and brain: The psychology of seeing*. Princeton university press, 2015.

[GW97] Tony L Goldberg and Richard W Wrangham. Genetic correlates of social behaviour in wild chimpanzees: evidence from mitochondrial dna. *Animal Behaviour*, 54(3):559–570, 1997.

[HCAT00] Brian Hare, Josep Call, Bryan Agnetta, and Michael Tomasello. Chimpanzees know what conspecifics do and do not see. *Animal Behaviour*, 59(4):771–785, 2000.

[HGS16] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2094–2100. AAAI Press, 2016.

[HNR68] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[HSM+00] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, 2000.

[HZ] Jeff Hwang and You Zhou. Image colorization with deep convolutional neural networks.

[KF91] Robert M Krauss and Susan R Fussell. Perspective-taking in communication: Representations of others' knowledge in reference. *Social cognition*, 9(1):2–24, 1991.

[LCH16] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*, 2016.

[Lin93] Long-Ji Lin. *Reinforcement learning for robots using neural networks*. PhD thesis, Fujitsu Laboratories Ltd, 1993.

[LMS16] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision*, pages 577–593. Springer, 2016.

[mal] The malmo collaborative ai challenge. `https://www.microsoft.com/en-us/research/academic-program/collaborative-ai-challenge/`. Accessed: 2017-05-16.

[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.

[Nar16] Sanmit Narvekar. Curriculum learning in reinforcement learning:(doctoral consortium). In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 1528–1529. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[OIM+16] Andrew Owens, Phillip Isola, Josh McDermott, Antonio Torralba, Edward H Adelson, and William T Freeman. Visually indicated sounds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2405–2413, 2016.

[PW78] David Premack and Guy Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(04):515–526, 1978.

[RBTBS15] Rachel A Ryskin, Aaron S Benjamin, Jonathan Tullis, and Sarah Brown-Schmidt. Perspective-taking in comprehension, production, and memory: An individual differences approach. *Journal of Experimental Psychology: General*, 144(5):898, 2015.

[RGHL09] Martin Riedmiller, Thomas Gabel, Roland Hafner, and Sascha Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73, 2009.

[SB98] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. A Bradford book. Bradford Book, 1998.

[SDM97] Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

[SHM+16] 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.

[Tho11] Edward Lee Thorndike. *Animal intelligence: Experimental studies*. Macmillan, 1911.

[TMK+17] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

[Tom09] Michael Tomasello. *The cultural origins of human cognition*. Harvard University Press, 2009.

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

[WD92] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[WdFL15] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015.

[YEG09] B. YEGNANARAYANA. *ARTIFICIAL NEURAL NETWORKS*. PHI Learning, 2009.

[YG04] Erfu Yang and Dongbing Gu. Multiagent reinforcement learning for multi-robot systems: A survey. Technical report, tech. rep, 2004.

[ZIE16] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Aqeel Labash**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis

   Using deep reinforcement learning to solve the perspective-taking task

   supervised by Raul Vicente, Jaan Aru, Tambet Matiisen, Ardi Tampuu

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 18.05.2017