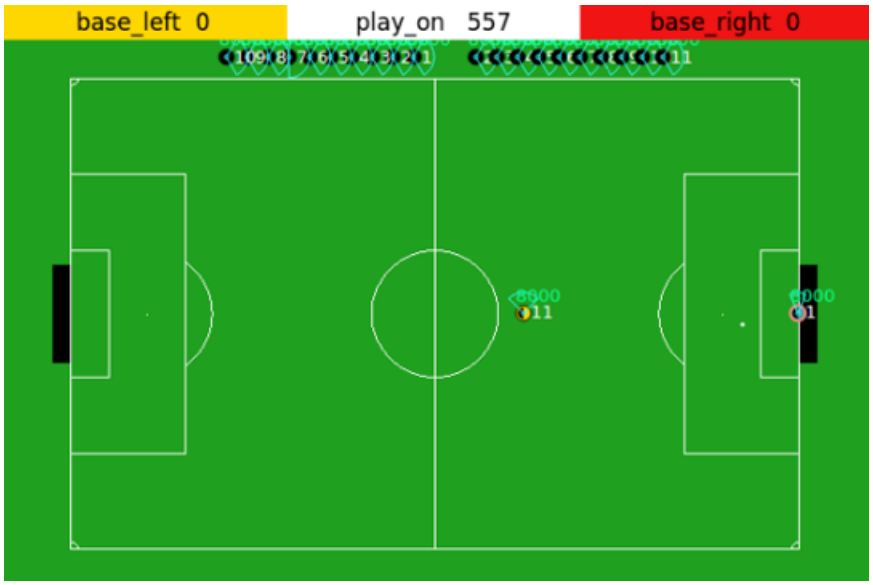
Making a robot stop a penalty

Using Q Learning and Transfer Learning

by Ruben van Heusden



Supervisor
dhr. dr. A. Visser

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

June 29th, 2018

Making a robot stop a penalty
Using Q Learning and Transfer Learning

Ruben van Heusden
11022000

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
dhr. dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

June 29th, 2018

Acknowledgements

I would like to sincerely thank my supervisor Arnoud Visser, who has been a great support in supplying constructive feedback on the project and giving me advice about the approach best taken.

Abstract

This thesis addresses the problem of decision making for an autonomous goalkeeping robot in a penalty shootout. A solution to this problem would be an advancement in the ability of autonomous decision making and learning in robots, and would contribute to the goal of having robot soccer players outperform human soccer players by 2050. The problem is complex as the amount of information the goalkeeper has is limited and the goalkeeper only has a limited time frame within it has to make a decision. Previous approaches often involved a human coding complex behavioral rules for the robots, which is very inflexible and the process of constructing behavioral rules and discovering the right parameters can be very time consuming. By using reinforcement learning, the goalkeeper in this thesis was able to stop a penalty without hand coded rules, learning from its own experience. The approach presented in this thesis uses Q learning combined with a deep neural network applied to discrete state and actions spaces and transfer learning to learn a behavioral policy. With intermediate rewards the algorithm is able to successfully stop a penalty in approx. 96% of the trials, with an average training length of 813 trials. Without intermediate rewards and using transfer learning, the agent was able to stop a penalty with an accuracy of approx. 64.5% after an average of 7476 trials.

Contents

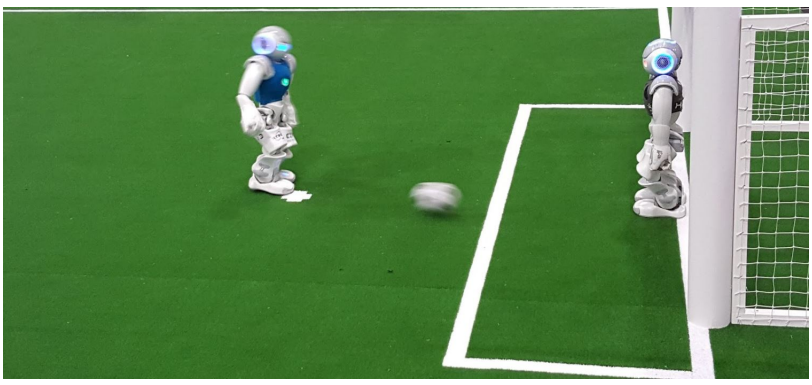
1 Introduction

In recent years, major advancements in the field of reinforcement learning applied to robotics have been made (Kober et al. (2013); Duguleana and Mogan (2016); Martínez-Tenor et al. (2017)). These advancements have sparked new research initiatives in the field of robotics and teaching complex behavior to robots. One such initiative is the RoboCup. In 1997 the first RoboCup was held, a competitive platform in which teams of robots are programmed to autonomously compete against each other in a game of soccer (Noda et al. (1998)). The aim of the RoboCup is to be able to create robots that are able to outperform human players by the year 2050[1].

The RoboCup is an important part of AI research, as a complex task such as playing soccer imposes many different challenges for robots, such as using computer vision to detect the ball and the goals, team tactics and localization techniques to maintain track of the robot's position on the field(Ferrein and Steinbauer (2016)). Learning to overcome those challenges and finding solutions to such problems will not only help robots improve at playing soccer, but acquired knowledge can also be readily applied to many other applications of Artificial Intelligence.

As stated in the RoboCup rulebook[2], the robot that is taking the penalty is only allowed to contact the ball once and has to score a goal within 30 seconds. If the kicker does not score within the time limit, the ball is stopped or the ball does not reach the goal, this will count in favor of the goalkeeper.

Figure 1: A penalty situation in the RoboCup SPL



(a) Picture taken from *https* : *//naoteamhtwk.blogspot.com*/2017/07/

For the construction of the goalkeeper behavior, multiple approaches exist. One option that is still quite common is to code complex rules by hand that dictate how the robot should behave in different situations based on hard-coded parameters. For example, the robot should approach the ball if it is within a certain distance or dive towards the ball if the ball has a certain speed and angle relative to the robot. The problem with these approaches however, is

[1] `http://www.robocup.org/objective`

[2] `http://spl.robocup.org/wp-content/uploads/2018/04/SPL-Rules_small.pdf`

that they are very inflexible and the making of these rules and the tweaking of the parameters is time-consuming. For this reason, a preferred method would be to let the robot learn the optimal behavior from its own experience in the environment using a learning algorithm. A class of learning algorithms that has proven to be quite successful in learning complex tasks is *reinforcement learning* (Sutton and Barto (1998)).

The principle of reinforcement learning is an agent interacting with an environment, e.g. taking actions and experiencing the results of those actions in the form of rewards or penalties. Traditionally, reinforcement learning has had difficulties in the field of robotics, where the state spaces are often very large or continuous.

However, recent research has shown that if reinforcement learning is combined with deep neural networks, impressive results can be obtained even in the cases of very large or continuous state spaces, for example by (Hausknecht&Stone, 2015) who used reinforcement learning combined with a neural network to train a robot penalty kicker to reliable score a goal against an hand-coded expert goalkeeper (Hausknecht and Stone (2015)).

The research question of this thesis is : *can reinforcement learning be used as a method of training a goalkeeper to stop a penalty in the RoboCup?* This research question is subdivided into two main parts, namely : *is it possible for a goalkeeper to stop a penalty with reinforcement learning with intermediate rewards?* and *is it possible for a goalkeeper to stop a penalty with reinforcement learning without intermediate rewards using transfer learning?*.

In Section 2, the theoretical background that is needed to understand the method that is used in this thesis sn introduced. In Section 3, the method that is used to answer the research question is formulated, along with specification of the used software. In Section 4 the results of this thesis are presented and analyzed. In Section 5 the conclusion of this thesis will be presented, and in Section 6 the related work, applicability of the developed method on the Nao Robot and some problems with the method used will be discussed. In Section 7 some suggestions for future work are made.

2 Theoretical Background

2.1 Reinforcement Learning

In reinforcement learning, the task that is to be solved by an agent is often represented by a Markov Decision Process (MDP)(Sutton and Barto (1998)). In this experiment, this is also an appropriate representation as it satisfies the *Markov Property*, meaning that a future state is only dependent on the current state and the current actions and not of the preceding sequence of events. An MDP consists of states $s \rightarrow S$, actions $a \rightarrow A$ and rewards $r \rightarrow R$.

In this setting, the optimal policy for an agent to follow is considered to be the policy that maximizes the sum of the agent's future (discounted) rewards, where γ is a parameter that weights the value of the reward based on how far in the future it occurs, up to a planning horizon T.

$$R(t) = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{1}$$

With this equation, the Q value of a state-action pair i.e. the 'quality' can be calculated, based on the current state and the action selected by the agent based on the current policy π.

$$Q^{\pi}(s,a) = E_{\pi}\{R(t)|s_t = s, a_t = a\} \tag{2}$$

The goal is to find a policy that always selects the action that maximizes the future return as formulated in equation 1.

2.1.1 On-Policy Vs. Off-Policy

Multiple methods for finding the optimal policy exist and can be broadly classified into two categories, namely *on-policy* and *off-policy* algorithms. An on-policy learning algorithm selects future actions based on the policy that the agent is currently following. An example of such an on-policy algorithm is SARSA (Sutton and Barto (1998)). The SARSA learning algorithm uses the following formula to calculate a state-action value for a given state-action pair.

$$Q(s,a) \leftarrow Q(s,a) + \alpha * [r + \gamma Q(s',a') - Q(s,a)] \tag{3}$$

where γ is a parameter for the weight of future rewards, and r is the height of the received reward. As can be seen in Equation 3 the update rule depends on $Q(s',a')$ which is the value of the state-action pair (s',a') , selected by using the current policy of the agent. An advantage of using on-policy learning is that, if the agent has already found the optimal policy it will always choose actions according to this policy, not choosing suboptimal actions. In contrast, off-policy learning algorithms determine the value of a state-action pair independent of the current policy. An example of such an algorithm is Q Learning. The update rule for Q Learning is the following :

$$Q(s,a) \leftarrow Q(s,a) + \alpha * [r + \max_{a'} Q(s',a') - Q(s,a)] \tag{4}$$

As can be seen in the above formula, the update of the Q value for (s,a) depends on $\max_{a'} Q(s',a')$. This means that Q learning selects a state $Q(s',a')$

based on the maximum of all the actions a' in state s'. This means that it is selected independent of the current policy the agent is following. The advantage of Q learning is that actions are chosen independent of the policy that is followed, meaning that exploration is built in to the algorithm, and alternative actions will automatically be explored.

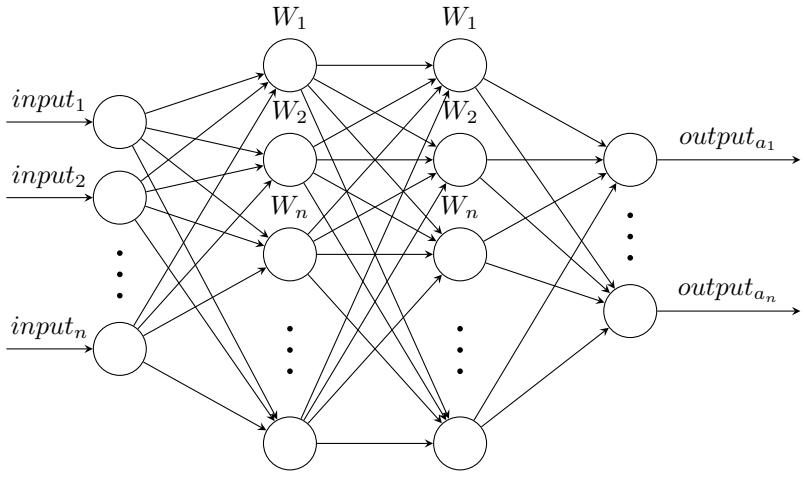
Traditionally, Q learning is used with a look up table to store all the state-action pairs (s, a), resulting in a look up table of size $num\_states * num\_actions$. This is problematic for very large state spaces, as the look-up table would quickly become very large and take a large amount of memory to store, which is problematic in for example the Nao Robot, which has very limited memory resources. A solution to this problem is to use a *function approximator*. A function approximator is a structure that takes as input attempts to approximate the state-action value of a state based on certain characteristics e.g. features of the state, instead of storing a state-action value for every possible state-action pair in the environment.

2.2 Reinforcement Learning with Neural Networks

In this thesis, a neural network is used as a function approximator for the Q values of possible actions. Function approximators have the advantage that in general, the algorithm is more memory efficient than a classic tabular based approach. Moreover, the use of a function approximator makes it possible for an agent to act in an environment with a continuous state space and still being able to learn an optimal policy.

As explained above, a neural network is used as a function approximator for the Q values of possible actions. The current state of the environment s is given as an input to the network in vector form. The input passes through several hidden layers, before an action value for every possible action is outputted.

Figure 2: Function Approximation with a neural network



The network is then updated according to a loss function e.g. the difference between predicted the Q value and the target value. In this algorithm, the target value is the Temporal Difference (TD) value :

$$Q(s, a) = r + \gamma Q(s', a') \tag{5}$$

where r is used to represent the reward the agent might receive, and γ is a discount factor weighing future states. In the neural network setting, the network itself is used to output the value for $Q(s', a')$. This leads to the following equation for the loss function of the neural network :

$$L_i(\theta_i) = \mathbb{E}((s, a, r, s')) \sim U(D)[(r + \gamma maxQ(s', a'; \theta_i^-)) - Q(s, a; \theta_i))^2] \tag{6}$$

(a) as stated in (Mnih et al., 2015)

where θ stands for the parameters of the Q network.
The use of the neural network function approximation has some drawbacks. One of the disadvantages of using a neural network is that the Q value predictions are approximations and therefore the predicted Q values do not represent the Q values exactly and might be incorrect. This has theoretical implications, as it is no longer guaranteed to converge to the optimal policy (Mnih et al. (2015)). This is problematic and can pose a problem if not dealt with correctly.

2.3 Deep Q Learning

The power of the use of neural networks as function approximators was first demonstrated by (Mnih et al. (2015)). In their paper, the propose an architecture for Deep reinforcement learning named *Deep Q Networks*(DQN). DQN offers a solution to the problems of stability in the use of neural networks as function approximators. By using two concepts they call *experience replay* and *target networks*, the stability of the networks is greatly improved.

2.3.1 Experience Replay

The first improvement made to the neural network is *experience replay*. The idea behind experience replay is that, instead of updating the model at every step, the agent keeps a record of the states it has visited in the past together with the corresponding actions it took and the rewards it received as a tuple (s, a, r, s'). Now when training the model, a batch of random experiences is drawn from this 'memory' and used to update the network. This is advantageous, as in the original version that updates at every step, the states used for the updates are all path of a trajectory the agent is following and are therefore more correlated, which in term leads to inaccurate predictions of Q values.

2.3.2 Target Network

The second improvement to the neural network is a the use of a *target network*. The use of a target network is another addition to the algorithm that is used to remove the correlation between the parameters in the network and improve the

stability of the network.

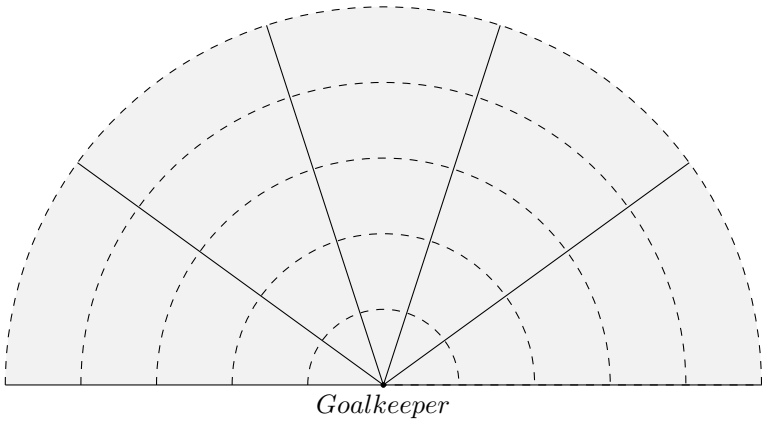
In the Q network, the target values for the network that are used in the loss function and to update the network are produced by the same network that is used for training. This approach creates a correlation between the Q values the network outputs and the target Q values. To solve this issue, a target network is used. This is an exact copy of the Q network used for training, but it is only updated towards the trained network with a very small amount τ every n steps to remove the correlation between the predicted Q value and the target value. This target Q network is then used to produce the target for the loss function of the neural network.

3 Method

3.1 State Space

For our purposes the state space consists of the discretesized position of the ball that is binned according to the position and angle of the ball with respect to the agent.

Figure 4: Bins based on angle and distance relative to the goalkeeper



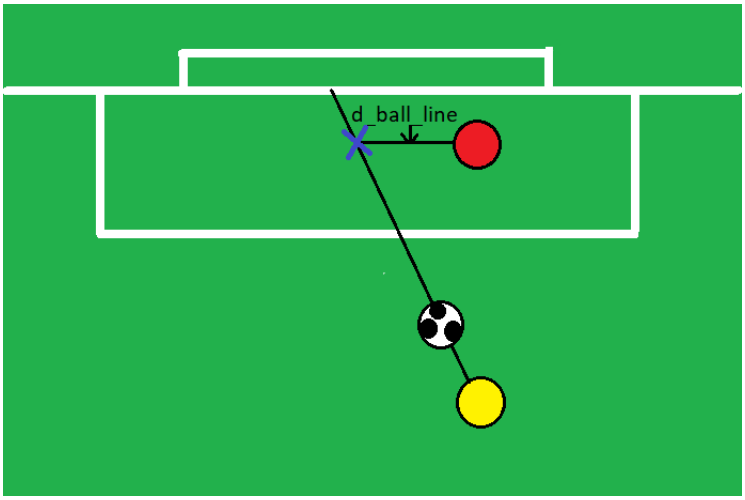
The more states used, the more detailed and accurate the agents understanding of the environment. However, increasing the number of states also increases the complexity of the environment and this may result in longer training times and increased memory usage. In the experiments, a state space consisting of 5 possible bins for distances and 5 possible bins for angles were used, where the maximum distance was the penalty spot and the angles varied from -90 to 90 degrees.

3.2 Rewards

The penalty shootout situation naturally consists of very sparse rewards. Intuitively, the keeper would receive a positive reward for stopping the ball, and

a negative reward if the penalty taker is able to score a goal. Because there are many steps between taking and action and receiving a reward however, it would be very difficult for the agent to ever make connections between the actions it takes and the rewards it receives. Therefore, intermediate rewards are introduced to guide the agent in the right direction. In this case, some caution with the type of reward must be taken as for the specific type of features the reward is based on. It might be intuitive to award the agent for its proximity to the ball, but this might yield unexpected results, as the ball is almost always heading towards the agent, and so any action taken by the agent will result in a reduced distance to the ball. To overcome this problem, a reward signal that is not dependent on the movement of the ball is chosen. The reward signal is based on the distance between the goalkeeper and a line that represents the trajectory of the ball, based on it's current velocity. By using the difference in distance between the current state and the previous state, the agent will receive a positive reward for moving towards the interception point with the line of the ball trajectory, and a negative reward if it moves away from this point.

Figure 5: The Reward Signal



For the agent that does not receive intermediate rewards, the reward signal is represented by the formula below :

$$r_{trial} = -500 * goal_{trial} + 500 * capture_{trial}$$

where $goal_{trial}$ and $capture_{trial}$ or both boolean values, e.g. 1 if true 0 otherwise.

In mathematical form, the equation would look like the following:

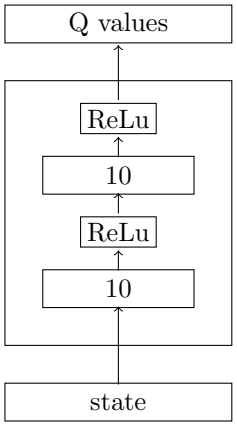
$$r_t = r_{capture} + 100 * (dist_{t-1}(goalie, ball\_line) - dist_t(goalie, ball\_line)) \quad (7)$$

In this equation, $r_{capture}$ is 500 if the goalkeeper catches the ball, and -500 if the kicker is able to score a goal.

3.3 Network Architecture

The architecture of the neural network is similar to that used in (Hausknecht & Stone., 2015). two hidden layers of 10 nodes each have been used. all these for layers are fully connected and use a Rectified Linear Unit (ReLu) as activation function, except for the output layer, which has a linear activation function. The target network has the same structure as the main network.

Figure 6: Neural Network Architecture



3.4 Training

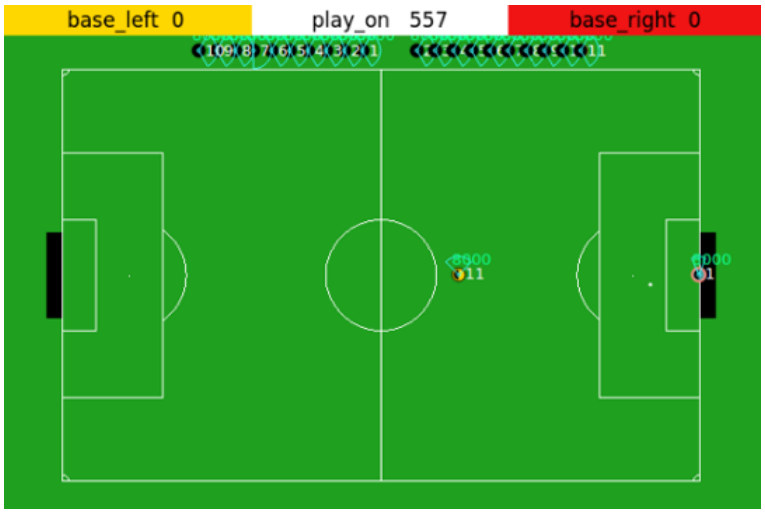
In the training of the neural network, two distinct phases can be identified.(Watkinson and Camp (2018)) In the first phase, the agent performs random steps for a certain amount of steps, in order to fill the experience replay buffer enough so that the network has substantial data to train with. In the second stage, the network trains every 50 steps. In each update step, a batch of 16 experiences is taken randomly from the 'memory' and used to update the network. Now, to calculate the loss function for the network, the target network is used. The target network starts out with the same parameters as the main network, but gets updated with a factor of $\tau = 0.001$ towards the main network every training step.

3.5 Simulation Environment

For the conduction of the experiments, the Half Field Offense (HFO) 2D Soccer simulator has been used[3]. In this 2D simulator, the behavior of robots can be programmed using various commands. Two feature sets containing information about the environment are available, for this thesis, the low level feature set is used as this provides low-level information, which can be translated to custom high level features later.

The simulator supports multiple levels of actions, from low level moves such walking and tackling to more complex strategic moves as reducing the angle between the ball and the goal. Because in this particular simulation the use of high level actions would make the task more or less trivial and it allows for much less freedom in the movement of the robot, the low level action set is chosen.

Figure 7: Screenshot of the HFO Simulator Environment



3.5.1 Actions

For this thesis, the choice has been made to use 6 possible actions. These actions are 'move slightly left/right', 'move moderately left/right' 'move left/right at full speed'. This choice was made as the penalty kicker can only shoot the ball once and so the keeper only has to be in the line of the ball and not necessarily approach the ball. Furthermore, the discretization of the actions reduces the size of the action space. Using a continuous action space would require a much more complicated algorithm, which is not necessarily needed for this particular task.

[3] `https://github.com/LARG/HFO`

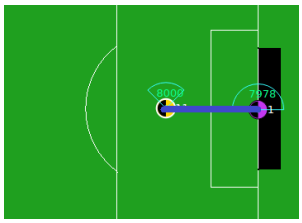
3.6 Transfer Learning

A promising technique that can be used in an attempt to reduce the amount of trials for the goalkeeper to learn the desired behavior is called *transfer learning.*(Taylor and Stone (2009)). In transfer learning, the agent starts with a relatively simple task and steadily progresses to more challenging tasks, using the information learned in the previous tasks to guide the learning in the current phase. In this thesis, 4 distinct phases are used in the training process.

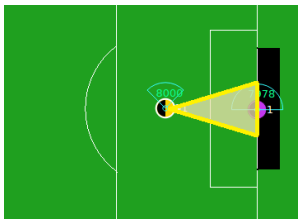
3.6.1 Phases

The transfer learning experiment consists of 4 tasks increasing in difficulty. In each task, the penalty kicker starts at the penalty spot in front of the goal and every shot is taken with a power setting of 40.0, as a maximum power setting makes it impossible for the goalkeeper to catch the ball if it is shot to one of the corners due to an inability for the goalkeeper to perform a diving save in the simulator.

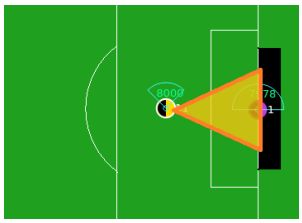
Figure 8: The four phases of training in increasing difficulty



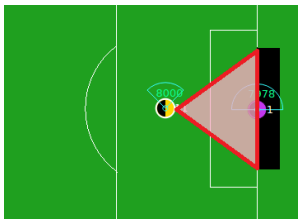
(a) Phase 1
angle of 0°



(b) Phase 2
angle between -10°and 10°



(c) Phase 3
angle between -20°and 20°



(d) Phase 4
angle between -30°and 30°

If the agent was able to stop a penalty 25 times in a row, the status of the neural network at that time is saved and the agent progresses to the next phase until it reaches Phase 4, after which it's accuracy is measured.

3.7 Evaluation

In the experiments, there is no set amount of training trials for the goalkeeper. Instead, the goalkeeper is trained until it is able to successfully stop 25 penalties in a row before it progresses to the next phase. This is done because there is a considerable variation in the amount it takes for the robot to master a particular task. This has several reasons, one of them being the random initialization of the

parameters in the neural network, the other being the stochastic environment in the simulator, meaning in two training sessions, the ball shot by the penalty taker can be very different.

4 Results

For all the experiments conducted below, the same settings for the learning parameters have been used, except for where noted.

parameter	value
α	0.1
τ	0.001
γ	0.99

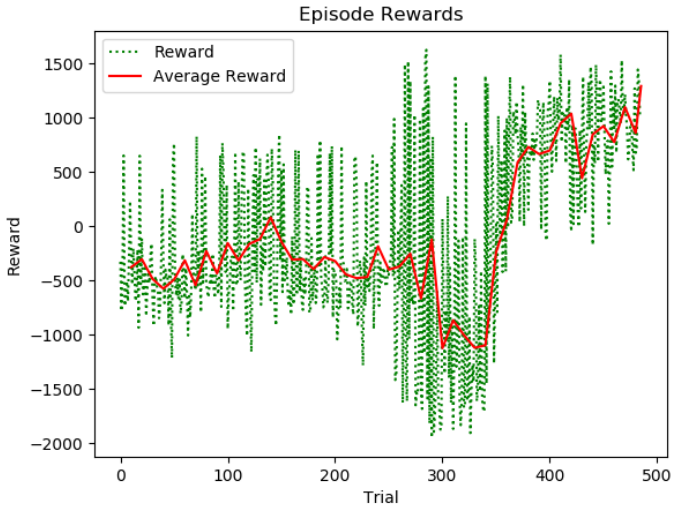
4.1 Learning with Intermediate Rewards

In comparison with the agent that received no reward during training, this agent receives intermediate rewards according to Equation 7. As this is expected to guide the learning in the right direction significantly, this agent is not trained using transfer learning, but is trained on the expert goalkeeper immediately.

On average, the goalkeeper with intermediate rewards learned to successfully stop 25 penalties in a row in 813 trials. After this training, the goalkeeper achieved an accuracy of 96.7%.

The figure below represents a typical learning curve for an agent that uses intermediate rewards. for the first 300 trials, the received rewards are distributed somewhat uniformly. This is because the agent performs random actions during this period until it has filled it's experience buffer with enough experiences to begin with training. During training it can be seen that the agent starts out with little positive rewards but the amount of rewards it receives per trial increasing as the training progresses, until it has stopped 25 shots in a row and moves on to the next phase.

Figure 9: Rewards per trial of agent with intermediate rewards



4.2 Learning without Intermediate Rewards

For the agent that was given no intermediate reward, it proved impossible to learn the task of stopping a penalty against a penalty kicker if it started out on full capacity. For this reason, transfer learning was introduced to attempt to let the agent learn the complex task by dividing it into tasks of increasing difficulty. This in order to make the agent first learn the basic concept behind the stopping of the ball, before introducing it to more complex situations e.g. wider angles of the ball that require the agent to move more. (see section 3.6 for more detail about the phases)

Below are the results for the agent trained with the intermediate reward, showing the average amount of trials it took for the agent to progress to the next phase.

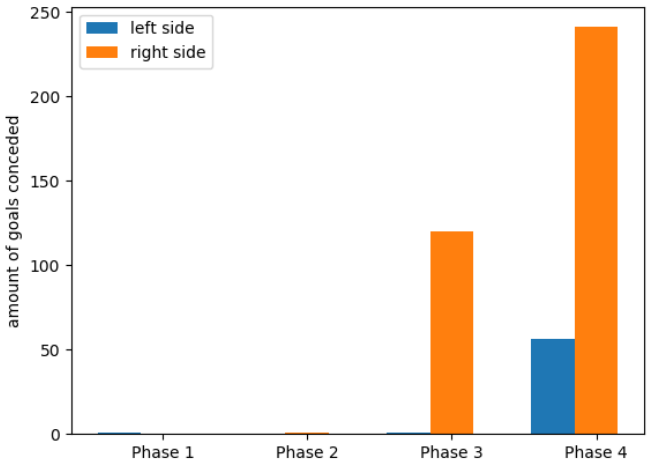
Experiment ID	Average Trials
Phase 1	844
Phase 2	1930
Phase 3	1769
Phase 4	2943

It was apparent during the training of the agent without intermediate rewards, that the learning process was very unstable, meaning that the amount of trials for the agent to succeed in a task varied widely, in contrast to the agent with intermediate rewards, where the learning process was much more stable. However, the agent without intermediate rewards was able to complete all 4 phases successfully. The average accuracy of the agent trained without intermediate rewards that had progressed through all 4 phases was 64.5%

4.2.1 Adjustments To Transfer Learning

Because the results of the transfer learning experiment did not outperform the agent trained with the intermediate reward, adjustments to the learning process were made in an attempt to improve the agent's results. By looking at the behavior of the agent through the visualizer, it was apparent that the agent had began to favor one side of the goal, most likely due to an uneven distribution of the amount of ball shot in each direction. Tests confirmed that the agent was performing very consistently with balls shot to one side, while performing very poorly with balls shot in the opposite angle. The reason for this behavior is not entirely clear, but it might be due to the structure of the neural network, as the goalkeeper performs significantly better if the agent is trained on ball shots to one side of the goal only than if trained to bot sides such as in the training phases.

Figure 10: sides to which a goal was conceded for an agent without intermediate rewards

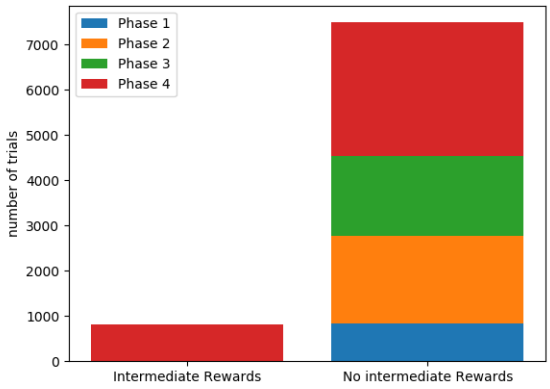


(a) total of 1000 trials

One option for improving the performance of the agent is to train the agent on balls shot to it's 'weak' side. As the goalkeeper always seemed to favor a specific side of the goal e.g. either left or right, the training was adjusted in order to compensate for this tendency. The goalkeeper was trained normally up to Phase 2, after which it was determined through visual inspection of the goalkeepers behavior in the simulator which side was favored. Before progressing through Phase 3, the goalkeeper was trained on catching penalties on it's weak side. Unfortunately this did not improve the performance of the agent and could even decrease the performance of the agent because the training would override the agent's previous strategy and overfit and only catching balls from one side of the goal if the learning rate was too high.

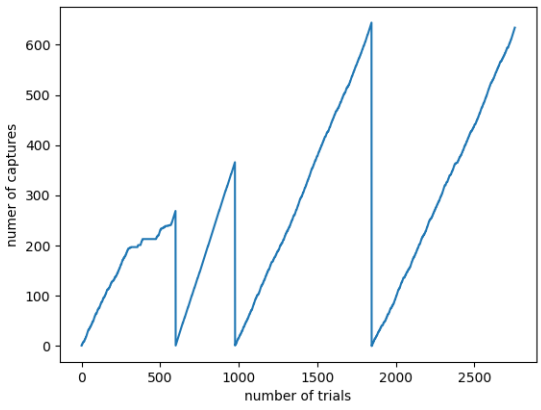
4.3 Comparison between Intermediate Rewards and No intermediate Rewards

Figure 11: Average amount of trials for both agent to complete the task



The above figure shows the average amount of trials needed for both agents to reach expert-level e.g. Phase 4. As can be seen from the figure, the agent that receives intermediate rewards is able to learn to master the expert task in much fewer trials than the agent that was not trained using intermediate rewards.

Figure 12: Amount of captures for all 4 Phases



The above figure shows the amount of captured balls for all 4 phases of the tansfer learning training process. The four phases can cleary be seen in the figure, and the scope of the line shows that the agent learns to stop the ball during the phase 4, the steeper the rate of descent, the better.

It is apparent from the above figures, that the agent that was given intermediate rewards outperformed the agent that did not receive intermediate rewards

on was trained using transfer learning. However, the agent that used transfer learning was able to learn the concepts of goalkeeping, with the only problem being did it did not seem to learn the appropriate behavior for both sides of the goal.
Video's of the behavior of both of the goalkeepers can be found on YouTube.[4][5]

5 Conclusion

In this thesis a method for applying Deep Q learning combined with transfer learning to a goalkeeper in a penalty situation in the RoboCup has been introduced. The results show that it is possible for a goalkeeper to learn to stop a penalty, both with and without intermediate rewards, with different accuracies. The goalkeeper that was trained with the intermediate rewards showed to learn the task faster than the agent trained without intermediate rewards, scoring a much higher accuracy, and displaying more stable learning. However, the agent that was trained without using intermediate reward was able to reach the final phase of training, meaning it did possess reasonable goalkeeping skills. Transfer learning proved to help in the training process for an agent without intermediate rewards, as an agent without intermediate rewards and no transfer learning was not able to reach Phase 4 of the experiments.

6 Discussion

6.1 Related Work

Extensive research on the application of learning algorithms for robotics and robot soccer has already been done in recent years. In a 2005 paper, Sutton, Stone and Kuhlmann developed behavior policies for keepers in the Half Hield Offense(HFO) keepaway game(Stone et al. (2005)). In the HFO Keepaway scenario, there are two teams, the *keepers* and the *takers*. The objective of the takers is to obtain the ball from the keepers while the keepers try to keep the ball in their possession for as long as possible. The approach taken in the paper involved using the SARSA learning algorithm with linear tile encoding as a function approximator to learn the agent the optimal policy using actions such as holding the ball or passing the ball to a teammate. This approach has similarities with this thesis in that at also attempts to make a soccer robot learn control policies using reinforcement learning. In this paper however, the keepers use high level control actions, whereas this thesis focuses on learning behavior from more low-level actions. A similar study has been done by (Hester et al. (2010)) in learning a robot how to score a penalty and this research focused on even lower level actions then our approach, namely on the specific joint angles of the robot. In the area of robot soccer penalty shoot-outs, multiple approaches have been used, most of them concerning the penalty kicker. In a thesis by (Lagrand (2017)), reinforcement learning combined with a neural network was used to make a penalty kicker learn to score against hand-coded goalie. The approach is similar in our approach in both the action space as well as the action space,

[4] `https://www.youtube.com/watch?v=ZcAtLxHpHR4`
[5] `https://www.youtube.com/watch?v=MlHNmP-_R3g`

which is also divided into bins for angles and distances relative to the robot. However, In this thesis, the focus is on training a goalkeeper, using reinforcement learning and different additions such as transfer learning to increase the performance of a goalkeeper not using intermediate rewards. The use of transfer learning in the RoboCup was first used by (Watkinson & Camp,. 2018). In this paper, transfer learning was used to train a penalty kicker to score a penalty against an expert goalkeeper (Watkinson and Camp (2018)). In their paper, they also used multiple phases of learning, but used in Actor Critic Network as function approximator combined with a target network and experience replay. In the paper, the agent was able to learn to score reliably against an empty goal and against a goalkeeper up to 50% after which intermediate rewards were used to make the agents able to defeat expert-level hand-coded defense agent. Their approach did not use the discretized state space like used in this thesis, but rather a continuous vector of various features of the environment, such as the distance and angle of the robot to various landmarks and the soccer field.

6.2 Applicability on the Nao Robot

The difficulty with the applicability of the current approach to the Nao Robot lies within the used abstractions of the state space, and that the features used in the robot simulator do not bear a one-on-one to the features available to the Nao in real life. Mainly, the discretization of the the position of the ball with respect to the robot is based on the absolute position of the ball on the field. This information is not available in this form and thus the representation of the bins shall have to be done differently. This could be done be retrieving the equivalent features of angle and distance from the camera of the Nao robot, but this is not straightforward and will require additional computational power in addition to the learning algorithm.

6.3 Transfer Learning

Although the use of transfer learning did significantly improve the ability for the goalkeeper without intermediate rewards to stop a penalty, the resulting strategy was visibly sub-optimal. The agent seemed to perform very well if balls were shot to one side of the goal, and very poorly if balls were shot to the other side, although the agent did make an attempt to catch all the balls to it's weak side. On explanation for this behavior could be that the goalkeeper has experienced more shots to one side of the goal and had learned to stop those shots better. However, compensating this by training it more on the agents' weak side after regular training did not improve the performance. One reason might be that the policy has reached a local optimum, which is very hard to overcome. Experiments with the size of the neural network can be done in an attempt overcome this behavior.

6.4 Size of The Neural Network

In thesis a neural network consisting of 2 hidden layers both having 10 nodes has been used. Although this approach yielded good results, the size of the network means that not all Q values cannot be fully represented by the neural network, something that could be problematic in learning the task. This can

be solved by using a larger network, possibly with a different architecture, for example a diamond shaped network, to be able to faithfully represent all the Q values.

7 Future Work

In this thesis the goalkeeper uses a set of discrete actions in order to stop a penalty. More research can be done into the use of continuous action parameters,similar to the learning algorithm developed in (Hausknecht and Stone (2015)), enabling more precise control for the robot. In this thesis, Q learning was used as the primary learning algorithm. More research can be done into the usage of other learning algorithms, for example SARSA. The used method in this thesis has proved to work as a method for a goalkeeper to stop a penalty. Research can be done into the generalization of this method so that the method can be used as a general policy for a goalkeeper during the entire game. In this thesis, the pre-train policy for an agent consisted of random movement in the Phase 1, and the network from the previous phases in the phases after that. More research can be done into using a different pre-train strategy, perhaps something similar to that used in (Smart and Kaelbling (2000)). Here, learning system of the agent observes for example a human performing the task, and in this way learns interesting components of the environment. Something that could enhance the performance of the goalkeeper is to use a different binning of the states than used here. For example, the closer the ball is relative to the goalkeeper, the more bins are used, thus giving the robot a better understanding of where the ball is if it is very close to the robot. This could be advantageous, because the closer the ball is to the goalkeeper, the more important it is for the goalkeeper to know the exact position of the ball, in order to make more effective decisions.

References

Duguleana, M. and Mogan, G. (2016). Neural networks based reinforcement learning for mobile robots obstacle avoidance. *Expert Systems with Applications*, 62:104–115.

Ferrein, A. and Steinbauer, G. (2016). 20 years of robocup. *KI-Künstliche Intelligenz*, 30(3-4):225–232.

Hausknecht, M. and Stone, P. (2015). Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*.

Hester, T., Quinlan, M., and Stone, P. (2010). Generalized model learning for reinforcement learning on a humanoid robot. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2369–2374. IEEE.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274.

Lagrand, C. G. (2017). Learning a robot to score a penalty minimal reward reinforcement learning. `https://esc.fnwi.uva.nl/thesis/`. Bachelor Thesis, University of Amsterdam.

Martínez-Tenor, A., Fernández-Madrigal, J. A., Cruz-Martín, A., and González-Jiménez, J. (2017). Towards a common implementation of reinforcement learning for multiple robotic tasks. *Expert Systems with Applications*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Noda, I., Suzuki, S., Matsubara, H., Asada, M., and Kitano, H. (1998). Robocup-97: The first robot world cup soccer games and conferences. *AI magazine*, 19(3):49.

Smart, W. D. and Kaelbling, L. P. (2000). Practical reinforcement learning in continuous spaces. In *ICML*, pages 903–910.

Stone, P., Sutton, R. S., and Kuhlmann, G. (2005). Reinforcement learning for robocup soccer keepaway. *Adaptive Behavior*, 13(3):165–188.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685.

Watkinson, W. B. and Camp, T. (2018). Training a robocup striker agent via transferred reinforcement learning. *RoboCup symposium 2018, Montreal*.