Efficient Training of Foosball Agents Using Multi-Agent Competition

Adriatik Gashi, Elke Hergenröther, Gunter Grieser

Darmstadt University of Applied Sciences
Schöfferstraße 3, 64295 Darmstadt
Germany

Abstract. In this work, an efficient training concept for training striker and goalkeeper Foosball agents using Deep Reinforcement Learning is designed and implemented. Based on a literature review, individual components such as the observation and action space are defined and suitable configurations for efficient learning are determined by experiments. Furthermore, different aspects for the modeling of multi-agent training are discussed on the basis of further literature research and a concept is derived from this for the use in the Foosball domain. Using only two consumer laptops, it could be shown that the use of multiple agents is also suitable in resource constraint domains. Despite an imbalance in the difficulty of the tasks of striker and goalkeeper agents, both agents were able to learn individual strategies while taking the respective opponent into account.

Keywords: reinforcement learning, multi-agent training, Foosball agents

1 Introduction

In recent years, deep reinforcement learning (DRL) has been successfully used in various areas like board- or real-time strategy games [30, 6]. These successes were achieved using multiple agents training with each other. By modeling competitions, agents could be trained with each other to develop strategies against competitors. At the same time, the aforementioned works have a high computational cost in common. *AlphaGo* used only for the match against the world champion in Go a total of 1920 CPUs and 280 GPUs [30]. For *OpenAI Five*, the agents were trained with a training duration of ten months and a training volume of approximately two million processed images every two seconds [6]. This raises the question of the extent to which training agents with each other

This is a preprint of the following chapter: A. Gashi, E. Hergenröther, G. Grieser, Efficient Training of Foosball Agents Using Multi-Agent Competition, published in Intelligent Computing, edited by Arai, K., 2023, Springer reproduced with permission of Springer Nature Switzerland AG 2023. The final authenticated version is available online at: https://doi.org/10.1007/978-3-031-37717-4_30

is possible with limited computational resources. For this purpose, the semi-automated Bosch Rexroth Foosball table from Figure 1 is investigated as a problem domain within the scope of this work. A simulation of the Foosball

Fig. 1. Semi-automated Bosch Rexroth Foosball table [9].

table is used to train DRL agents to operate the striker- and goalkeeper rods using multi-agent competition. Two consumer laptops are used for the training of the agents. Therefore, a training concept is developed that considers the limited computational resources available. To accomplish this, we first formalize the environment as a single-agent Markov Decission Process (MDP) from the perspective of the striker-agent in section 4, selecting a suitable learning algorithm and evaluating different configurations by experiment. Afterwards, we extend the MDP to a Markov Game to contain the goalkeeper-agent in section 5, adjusting the training process accordingly.
Our results show that striker- and goalkeeper agents were able to develop offensive as well as defensive strategies in the multi-agent setting. While the task design showed an imbalance in the difficulty for the rods, both agents were able to develop behaviour to successfully shoot goals and control the ball with high precision. This suggests that using multi-agent competition to develop complex behaviour with relatively small computational resources is possible.

2 Related work

First, related work with respect to the Foosball domain will be presented, followed by works that used simulation environments to train agents.
In [34] an automated Foosball table was developed, whose control was implemented by means of a decision tree. In addition, a simulation implemented by the authors allows testing of different strategies by using two agents. [37] and [13] are other Foosball examples, where a Foosball rod was taught different action sequences by imitating human actions, and a method for tracking the ball using a camera was developed, respectively. In [9], DRL agents were used to teach a

striker rod to score goals. The focus of their work was the "Sim-to-Real" domain, where the Foosball example was chosen to represent a complex manufacturing-like process, which was optimized using DRL. Using a simulation of the foosball table, the authors were able to train agents that successfully learned demanding Foosball control strategies using sparse reward signals. However, in contrast to this work, only one agent is trained and deployed in the respective environment. The authors used the same Foosball table as considered in this work. Furthermore, in [27] a Foosball goalkeeper was taught to defend the goal using DRL. The authors used Deep Q-Learning [18] to train a goalkeeper agent on a variety of direct shots at the goal. However, the authors used a discrete action space that represents only lateral movements of the rod, disregarding rod rotations altogether.
In [20] a robot hand could be taught to solve a Rubix cube. For this purpose, the authors used a simulation in which the agent was trained and then used this model to control a real robot hand. Thus it could be shown that by using a simulation, agents can be trained that achieve comparable results in reality. This was also shown in [21] by using visual object recognition to position objects by a robot hand as in a pictorial representation. In [30, 32, 31] agents learned grandmaster-level behaviours for board games like Chess, Shogi and Go using simulations. In [6], a complex training system was used to beat world champions in an online real-time strategy game. However, these works necessitated the use of large training systems with numerous computing units, and the training process for the agents took several weeks to several months.

3 Foosball Domain

The starting point of this work is the Foosball domain. Here, this work refers to the semi-automated Foosball table developed by Bosch Rexroth shown in Figure 1. In order to be able to train agents for this Foosball table, a simulation with the CAD model of the Foosball table was recreated in Unity. With this simulation, a goalkeeper agent was already successfully taught to defend shots [27]. We extend the work from [27] to train striker and goalkeeper agents in this simulation. However, only two consumer laptops with the specifications mentioned in Table 1 are available for this purpose. This has to be taken into account when designing the training procedure, and therefore the efficiency of the training is of great importance when determining the individual training components. At the same time, the simulation is only an abstraction of the Foosball table, which is why the simulation-to-reality gap must also be taken into account.

4 Single Agent DRL

In order to use DRL agents to control the striker and goalkeeper rods, the task to be learned is formalized as an MDP, a suitable way to formally define an DRL problem [35, p. 10, 12]. Additionally, a learning algorithm must be determined. For this purpose, this section first defines potential dimensions of observation

Table 1. Hardware specification of the available computational resources.

System model	ThinkpadP1Gen2	Lenovo81NX
Processor	Intel Core i7-9750H @ 2.60GHz	Intel Core i7-9750H @ 2.60GHz
RAM	32.0GB	16.0GB
GPU	NVIDIA Quadro T1000	NVIDIA GeForce GTX 1650 with Max-Q Design

and action space, as well as a reward function from the perspective of the striker rod. Then, a DRL algorithm for training the agents is selected, followed by a methodological determination of the observation and action space to be used for multi-agent training.

4.1 Definition of MDP

Observation Space Observations depict state representations of an MDP. For this purpose, the frequency and the representation of an observation must be defined. Considering the observation frequency, it should be chosen as high as possible on the one hand, in order to be able to draw precise conclusions regarding the selected actions, but on the other hand realistic regarding the real problem domain, so that the simulation-to-reality gap remains small. The observation frequency chosen was 60 observations per second, as this is a frequently used frequency in established DRL benchmark environments [5] and is within realistic limits with respect to the human observation frequency of the eye [8, 23].

Feature vectors were chosen to represent observations, as this significantly reduces dimensionality compared to using images and eliminates the complexity of image processing. Of particular importance is which observation channels are made available to the agent for the selection of actions, as MDPs assume the Markov-Property to be satisfied. Therefore information about both the rod and the ball must be provided. Tables 2 and 3 present observation channels relevant to rod and ball, respectively, and the value ranges used to encode this information. Since no boundary values are known for the ball velocity, the range of values here were limited to $[-10, 10]$ in a zero-centered manner, in order to avoid larger expressions that could negatively influence training stability.

However, this minimal configuration may not represent the optimal combination of observation channels. One piece of information that is currently hidden from the agent is the number and position of players on a rod. By adding this information, the agent would not have to implicitly learn the number of players and their respective positions, but could accurately track them via the observation channels. Since the players are fixed to the rod and the distances between them thus remain the same even when they move, all that is needed here is a scalar value of the position of the players.

Furthermore, several works have investigated the influence of binary contact information on locomotion problems with the result that it can enable smaller

Table 2. Observation channels regarding the Foosball rod with respective encoding

Observation channel	encoding
Lateral rod position	$[-1, 1]$
Angular rod position	$[-1, 1]$
Lateral rod velocity	$[-1, 1]$
Angular rod velocity	$[-1, 1]$

Table 3. Observation channels regarding the Foosball ball with respective encoding

Observation channel	encoding
x-coordinate ball	$[-1, 1]$
z-coordinate ball	$[-1, 1]$
x-velocity ball	$[-10, 10]$
z-velocity ball	$[-10, 10]$

performance gains [26]. From this, two potential observation channels can be derived: binary contact information and binary range information, where the binary contact information indicates a contact between player and ball and the binary range information indicates when the ball is in range of the striker rod.
In the context of this work, the focus is on an efficient training of agents, which has to be considered when determining the observation channels to be used. For this purpose, the method described in [15] was used to determine an optimal observation space. The experimental setup and the results of this method are explained in more detail in section 4.3, since an action space and a reward function must first be defined for this purpose.

Action Space The action space of a Foosball rod can be divided into two dimensions: the angular and lateral movement of the rod. Here, the encoding of both dimensions has to be determined, which can be continuous or discrete in nature. In [14], it was shown that by discretizing action spaces, training efficiency could be significantly increased. However, [33] showed that while discretization can be advantageous in complex tasks, in simpler tasks a continuous action space can be essential for achieving a high return.
To determine the encoding for the action space, an experiment was held to compare discrete and continuous value ranges. This experiment will be explained in section 4.4, because for this a reward function has to be determined first.

Reward Structure The goal of the Foosball striker is to score goals. At the same time, it must be avoided that the ball gets behind the striker. Thus, it is first noted that a positive reward is given for a goal and a negative reward is given if the ball gets behind the striker. If we assign a value of 1 for a positive reward and -1 for a negative reward, we can assume that scoring goals is just as important to the agent as defending the ball. However, since there are wide walls next to the goal, the probability of the ball bouncing back when using a random policy is much higher than randomly shooting goals. Since this can quickly create a negative incentive to refrain from shooting altogether, a negative reward of -0.1 is chosen instead of the negative reward of -1. With this ratio of $\frac{1}{10}$ it can be assumed that the incentive to shoot is still maintained despite initial bounces. At the same time, it can be assumed that optimizing a policy leads to the prevention of wall bounces.

However, to make the training more efficient, the method potential-based reward shaping (PBRS) is used [19]. This allows a policy-invariant condensation of the reward function. For this the ball position on the field is used to define a potential function ψ^{striker}:

$$\psi^{\text{striker}}(S_t) = \frac{\psi_x^{\text{striker}}(X_{S_t}) + \psi_z^{\text{striker}}(Z_{S_t})}{2} \tag{1}$$

where ψ_x^{striker} and ψ_z^{striker} describe the potential of the ball with respect to its x- and z-coordinate respectively. Given the encoding $[-1, 1]$ of the x-coordinate of the ball, where the striker rod represents the value 1 and the goal the value -1, ψ_x^{striker} can be defined as the following linear function resulting in the value range of $[0, 1]$:

$$\psi_x^{\text{striker}}(X_{S_t}) = -0.5X_{S_t} + 0.5 \tag{2}$$

where X_{S_t} denotes the x-coordinate of the ball at state S_t and time step t. However regarding the z-coordinate, the width of the goal corresponds to the range $[-0.26, 0.26]$, which should be equivalent to the highest potential value of 1. The potential of the ball's z-coordinate decreases with values below or above the goal width. In order to represent that, the following non-linear potential function ψ_z^{striker} is used, which clips the potential values at 1 and results in the value range $[0, 1]$:

$$\psi_z^{\text{striker}}(Z_{S_t}) = \min(-1.36|Z_{S_t}| + 1.36, 1) \tag{3}$$

where Z_{S_t} denotes the z-coordinate of the ball at state S_t and time step t.
This results in the following shaping-reward function $F(S_t, S_{t+1})$ based on [19]:

$$F(S_t, S_{t+1}) = \gamma\psi^{\text{striker}}(S_{t+1}) - \psi^{\text{striker}}(S_t) \tag{4}$$

where $\gamma <= 1$ is the discount factor of the environment and S_t represents the state at time step t. Given the value ranges of ψ_x^{striker} and ψ_z^{striker} being $[0, 1]$, the value range of the shaping-reward function becomes $[-1, 1]$.
Combining this shaping-reward function F and the rewards for the terminal states of the MDP concludes to the following reward function from the striker perspective:

$$\mathcal{R}(S_t, A_t, S_{t+1}) = \begin{cases} 1 \text{ if } S_{t+1} = \text{goal event} \\ -0.1 \text{ if } S_{t+1} = \text{ball behind striker event} \\ F(S_t, S_{t+1}) \text{ otherwise} \end{cases} \tag{5}$$

4.2 DRL-Algorithm

Proximal Policy Optimization (PPO) [29] is used as the DRL algorithm, since it is suitable for discrete and continuous action spaces and was successfully used in the Foosball domain [9] as well as complicated multi-agent environments [6, 36]. However, the implementation to be used must also be considered. As shown in [10], individual source code optimizations can have a greater impact on the

result than a different choice of learning algorithms. Due to this the implementation Stable-Baselines3 (SB3) is used [25]. It has the source code optimizations mentioned in [10], a high test coverage and was already used in further work [14].

Hyperparameter PPO has several hyperparameters that can sensitively influence learning behavior. In this context, hyperparameter determination is often a task that takes a lot of time [12]. Since computational resources are limited and multiple experiments are to be performed in this work, a time-consuming hyperparameter search is not performed. Nevertheless, in order to be able to select hyperparameters that enable learning, the work of [3] is consulted, which represents a systematic investigation of individual hyperparameters for on-policy learning. Furthermore, the optimized hyperparameters for SB3 in known DRL benchmark environments mentioned in [24] are used, since these have been specifically determined for the implementation using Optuna[1]. Last, the work of [9] and the hyperparameters used there are also considered, since the authors also use the Foosball domain and teach a striker rod to shoot. Table 4 shows the determined hyperparameters for the PPO implementation of SB3, where [24, 3] stated that the neural network width and learning rate depend on the complexity of the environment and can cause significant drops in performance if set poorly. Additionally, we use running mean normalization, since this is also an important source code optimization as described in [10].

Table 4. Defined Hyperparameter for PPO based on [3, 9, 10, 24]

Hyperparameter	Value	Source	Hyperparameter	Value	Source
clip range value function	0.2	[10]	activation function	*tanh*	[10]
max grad norm	0.5	[10]	learning rate	3e-5	[24, 3]
entropy coefficient	0.01	[24, 9, 3]	rollout length	2048	[3]
value function coefficient	0.5	[10]	minibatch size	64	[24, 3]
advantage normalization	*true*	[24]	amount epochs	10	[3]
orthogonal initialization	*true*	[10]	discount factor	0.99	[3]
network type	fully connected	[24, 3]	GAE lambda [28]	0.95	[24, 9]
network layers	2	[24, 3]	clip range	0.2	[10]
neurons per layer	64	[24, 3]			

4.3 Evaluation of Observation Space

The optimization algorithm described in [15] is used to determine the observation channels to be used. The initial observation space is determined as a combination of the rod and ball observation channels as shown in Tables 2 and 3. The reward function mentioned in section 4.1 is used. Figure 2 represents the initial situation of an episode. The ball is randomly placed in front of the players of the

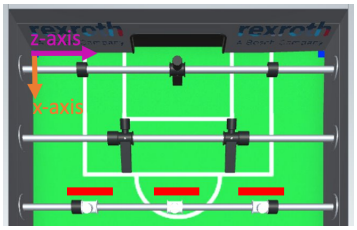

Fig. 2. Initial state distribution for striker rod. The red lines mark the areas where the ball can randomly spawn at the beginning of an episode.

striker rod at the start of an episode, as represented by the red lines. The striker rod is placed in the middle lateral position at the beginning of each episode with the players in the vertical position. This is to keep the initial state distribution small to allow efficient learning on the available computational resources [26]. If the ball stops in an area that the striker rod cannot reach, the episode is terminated. Here, the infinite bootstrap trick elaborated in [22, 26] is used to provide a distinction between terminal states in the MDP and termination after timeout.

A continuous two-dimensional action space with the value range $[-1, 1]$ is used. The experiment was repeated with five different random seeds, as the network initialization can influence the performance of an agent greatly [11]. In [9], a striker agent was successfully taught to shoot after only 48 hours, with a significant reward achieved after only 24 hours. Since the neural network used in this work, with two hidden layers and 64 neurons each $[64, 64]$, is significantly smaller than the one used by the authors $[512, 512]$, a training duration of eleven hours is used, which corresponds to 1050 rollouts or approximately 2.16 million time steps in the simulation.

Comparing the achieved performance with the initial observation space and the initial observation space with player positions in Figure 3, it can be seen that by using the player positions, a slight improvement of about 5.3% could be achieved, measured using the maximum achieved values of both.

Now, to determine whether this combined observation space has harmful observation channels, the dropout permutation test described in [15] is applied. First, the five trained agents were evaluated over 100 episodes and the achieved reward per episode was recorded. This value is used as a baseline for calculating an importance score for each observation channel. To calculate this, the individual observations were saved during training and histograms were created for the respective observation channels. For the dropout permutation test, data from the histogram was used for individual observation channels instead of data from the current episode in order to provide the agent with false but realistic data. This was done for each observation channel and all five trained agents for 100 episodes and the achieved reward per episode was recorded. By relating these perturbed reward values to the baseline reward value, the importance scores shown in Figure 4 are obtained, where a negative importance score represents a

useful observation channel and a positive score represents a harmful one.

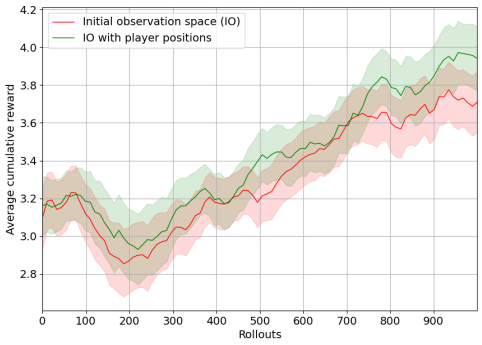

Fig. 3. Average cumulative reward per rollout describing the effect of player observation channels. The shaded area represents the respective bootstrapped 95% confidence interval.

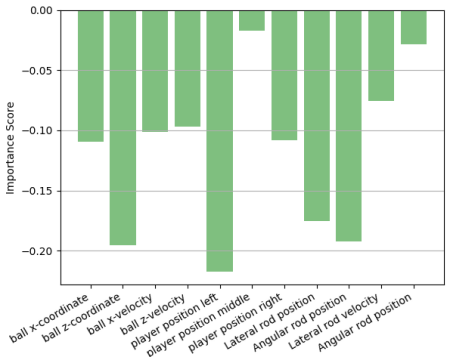

Fig. 4. Importance score of the individual observation channels compared to baseline, based on [15].

The results show that all observation channels achieve a negative importance score. Nevertheless, to test whether reduced dimensionality has a positive effect on learning efficiency, the observation channel with the lowest importance score is removed and the experiment is repeated with the same seeds. Figure 5 shows

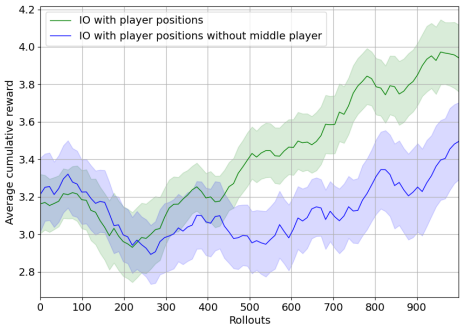

Fig. 5. Average cumulative reward per rollout describing the effect of removing the least important observation channel regarding its importance score. The shaded area represents the respective bootstrapped 95% confidence interval.

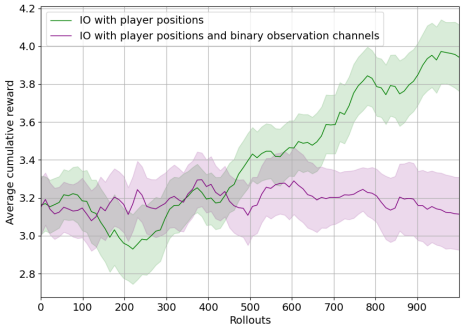

Fig. 6. Average cumulative reward per rollout describing the effect of binary observation channels. The shaded area represents the respective bootstrapped 95% confidence interval.

that removing the observation channel with the worst importance score has a

negative impact on the achieved cumulative reward. Consequently, no observation channel is removed, as all other observation channels have better importance scores. If we now add the binary observation channels from section 4.1 and compare the obtained rewards, we see in Figure 6 that they have a strong negative effect on learning.
Due to this, the initial observation space will be extended with the player positions of the rod. At the same time, however, it should be noted that the learning curve does not yet show saturation. This can be an indication that a higher training duration can lead to a better performance.

4.4 Evaluation of Action Space

The same experimental setup as in section 4.3 was used, as well as the final observation space from section 4.3. However, since no saturation of the learning curves could be shown, the training duration for this experiment is doubled to 22 hours or 2100 rollouts. This experiment is also repeated with five different random seeds. Here, a two-dimensional action space is compared with a continuous and discrete coding. For the continuous value space $[-1, 1]$ is chosen for both dimensions. A discrete encoding can be mapped both one-dimensionally or discretely and multi-dimensionally or multi-discretely. However, in [14] it could be shown that multi-discrete action spaces allow more efficient learning, which is why this variant was used. As discretization factor in [33], the value 11 could convince in different environments, which is why this value is used here as well. Thus, these action spaces are compared with each other.

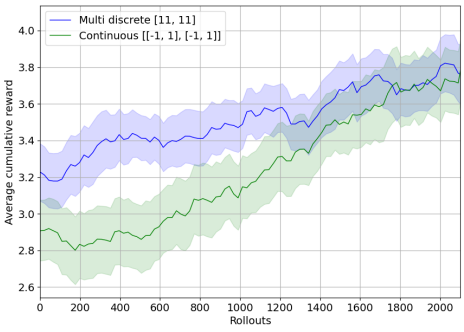

Fig. 7. Average cumulative reward per rollout with initial hyperparameter configuration. The shaded area represents the respective bootstrapped 95% confidence interval.

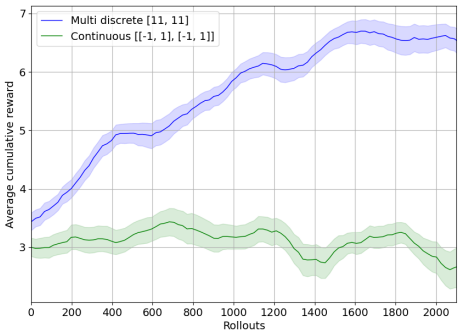

Fig. 8. Average cumulative reward per rollout with increased learning rate and neural network width. The shaded area represents the respective bootstrapped 95% confidence interval.

Figure 7 shows that the multi-discrete action space achieves a higher reward up to approx. rollout 1600. After rollout 1600, however, both curves converge strongly, which is why a final classification is difficult. At the same time, it was

observed that no significant increase in reward could take place despite double the training duration. For this reason, the experiment is repeated with adjusted hyperparameters. First, the learning rate is increased to 3e−4, since the slope of the reward curves is relatively low. Second, the network width is doubled to $[128, 128]$, since according to [3] this hyperparameter can have a large impact on the performance of the agents. The result of the adjusted parameters is shown in Figure 8. It can be clearly seen that the adjusted hyperparameters led to a strong improvement with the multi-discrete action space, but made learning with a continuous action space much more difficult. The worsening of the results for continuous action space with increased learning rate and larger network may indicate that the use of a continuous action space is harder to learn and therefore benefited from a smaller learning rate. This would also be in line with the results from [3], where a smaller learning rate may be essential especially in more complex environments. For the multi-agent training, this work continues with the multi-discrete action space.

5 Multi-Agent DRL

Foosball represents a domain in which two players play against each other. It follows that a good striker agent must be able to play well in a team and well against an active opponent, among other things. Thus, while a striker can be trained alone to learn the dynamics of the ball, this is no substitute for training against an active opponent who can pursue his own strategies. This is why training a striker agent requires an active opponent. In the case of this work, striker and goalkeeper agents are trained together for this purpose. This requires an extension of the MDP defined in section 4.1. In order to train the agents with each other, a modeling of the opponent's policies has to be done and a method to determine the training partners has to be defined. Finally, the designed training concept is executed in an experiment and the results are evaluated.

5.1 Extension of MDP

Since each agent has its own action space, the action space for both agents each remains a two-dimensional multi-discrete action space with dimensions $[11, 11]$. Nevertheless, the agents' environment is extended by adding an opponent rod, which requires the observation space of striker and goalkeeper agents to be extended by their respective information regarding the opponent. At the same time, the defined reward function has to be analyzed and modified, since it is only focused on the striker agent and now has to be extended to include the goalkeeper as well.

Observation Space The opposing player is not part of the observation space and thus cannot be explicitly considered when selecting an action. To counteract this and to make the training easier, information about the opponent is added to the observation space. Therefore, the observation space contains the lateral-

and angular position and -velocity of each agent, as well as the ball position and velocity.

Reward Structure To determine the reward function of each agent, their objectives are considered. The objective of a goalkeeper is to prevent goals, while the objective of a striker is to score goals. Because of these strictly opposite objectives, it is suitable to define it as a zero-sum Markov game. A zero-sum Markov game is a Markov game in which the reward of one agent is the negative reward of another agent in the environment. While this is trivial for the goal- and ball-behind-striker-events, the potential function of both agents has to be redefined in order to fulfill the zero-sum requirement. As defined earlier, for the striker both the x- and z-coordinate of the ball are important, as the goal is only in the middle of the field. However, this is not the case from the goalkeeper perspective. The goalkeeper has to shoot the ball past the striker rod, which can happen at an arbitrary z-coordinate. Therefore, the potential function of the goalkeeper only regards the x-coordinate of the ball as follows:

$$\psi^{\text{goalkeeper}}(S_t) = 0.5X_{S_t} + 0.5 \tag{6}$$

where X_{S_t} describes the x-coordinate of the ball at state S_t and time step t. Finally, for the potential functions to fulfill the zero-sum requirement, the following potential functions ψ' are used respectively:

$$\psi'^{\text{ goalkeeper}}(S_t) = (\psi^{\text{goalkeeper}}(S_t) - \psi^{\text{striker}}(S_t))/2 \tag{7}$$

$$\psi'^{\text{ striker}}(S_t) = (\psi^{\text{striker}}(S_t) - \psi^{\text{goalkeeper}}(S_t))/2 \tag{8}$$

5.2 Adversarial Training

In [4], the authors tested how training with only one training partner compared to training with multiple training partners. Their results showed that training with only the most current training partner resulted in an imbalance in training. Using different training partners of different strengths led to more stable training and learning of more robust policies by both agents. Therefore, an ensemble of agents is used for training in this work. This raises the possibility of using a centralized critic, which will be discussed. Furthermore, it has to be determined whether only one type of agent should be used for striker and goalkeeper rod and how the opponent selection should be performed. Finally, the episode structure has to be redefined so that striker and goalkeeper agents can train with each other.

Centralized vs. Decentralized Critic Since PPO is an actor-critic algorithm, it uses an actor network and a critic network. This circumstance was used in [36] to provide the critic network with more information than the agent has available in the subsequent evaluation environment. Through this, the true value function

of the environment should be better approximated by the critic network, allowing better policies to be learned by the actor network during training. This approach is referred to as Centralized Training for Decentralized Execution (CTDE) [36]. The impact of centralized critics was studied in [17]. In their empirical evaluation, the authors concluded that the choice between centralized and decentralized critics is equivalent to a bias-variance tradeoff, since decentralized critics have a higher bias due to the use of less data and learn less correct value function than centralized critics. At the same time, it was observed that decentralized critics consistently performed more robustly than centralized critics in the environments they used, as more stable adaptations of the networks in these environments seem to be more important than more correct estimation of the value function. Due to the diversity of the environments used by the authors, as well as the consistently positive results achieved by decentralized critics in those, decentralized actor-critic architectures for goalkeeper and striker agents are used in this experiment.

Modelling of Adversarial Policies It is necessary to determine whether one agent should be trained for striker and goalkeeper rod or one agent for each rod type. This type of self-play is often used in symmetrical environments where it does not matter on which side an agent is positioned in each environment. In this work, however, this is not the case, as both rods have different movement constraints and a different number of players per rod. At the same time, the goalkeeper rod has a wall from which balls can bounce without problems and raised corners that influence the ball dynamics. Thus, due to the asymmetry of the game, the use of different agents for each striker and goalkeeper is suitable. This method was also used in [4] for asymmetric games like *kick-and-defend* or *you-shall-not-pass*.

Opponent Selection Different works have taken different approaches to training partner selection. In [38], the authors evaluate these different approaches and develop a framework for training partner selection. They show through experiments in different environments that their approach achieves better results than using the most recent, the historically best, or a random historical agent. This algorithm is chosen as the method for opponent selection in this work. Here, an ensemble of agents of size n is trained over N iterations. At the beginning of an iteration, the strongest opponent is determined for each agent. For this purpose, an evaluation over m episodes takes place by competing each agent with each other. After the evaluation, the strongest training partner is determined for each agent and deployed into the environment as a stationary opponent. The agent to be trained is trained against this stationary opponent for a duration of l steps. This takes place for all n agents per iteration N.
Thus, four parameters need to be determined for the use of this algorithm: The number of iterations N, the size of the agent ensemble n, the training duration per iteration l, and the number of episodes for the evaluation of the agent strength m. Since only small computational resources are available, the number of iterations N is kept small while the duration of a training l is increased. This

is to increase the stationarity of the environment, which should contribute to more stable learning.
Furthermore, in section 4.4, convergence of the striker agent could be observed after about 1200 rollouts or 12 hours. Based on this, the training duration $l = 1200$ rollouts is set. As in the experiments before, the number of episodes for evaluating the agent strength m is chosen to be 100. Like in [38], parallel runs are used to reduce the training duration by training striker and goalkeeper agents each in parallel in two environments.
The ensemble size evaluated by the authors was 2,4 and 6, with a larger ensemble consistently resulting in stronger agents. In the context of this work, an ensemble of $n = 4$ agents is chosen, which corresponds to four striker and four goalkeeper agents. This is to promote stability in training compared to using two agents at a time.
Considering this ensemble size and the training duration per iteration, one iteration takes about 48 hours on the available hardware. For this reason, a small number of iterations of $N = 3$ is specified, which corresponds to a total training time of about 144 hours for a total of 4 striker and 4 goalkeeper agents and a total training time per agent of 3600 rollouts or 36 hours.

Episode Structure When determining the episode definition, it is important to ensure that the difficulty levels of both agents are as similar as possible to avoid unstable training with early dominance of one agent. At the same time, care should be taken to ensure that the initial state distribution is kept narrow, as in the previous experiments, in order to increase training efficiency. For this,

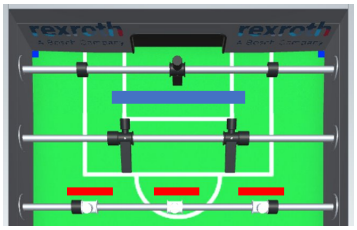

Fig. 9. Initial state distribution for striker and goalkeeper rod. Red and blue represent where the ball spawns in front of the striker- and goalkeeper rod, respectively. The initial state distribution for the goalkeeper rod is larger compared to the striker rod.

the ball is placed alternately in front of the striker and goalkeeper rods. For the striker, the ball is placed randomly on the red lines in Figure 9, as in the previous trials, since this allowed a good policy to be learned, but it was not too easy to converge directly. For the goalkeeper, there are several aspects to consider. First, the goal to be defended is much smaller compared to the striker. Furthermore, the goalkeeper has only one figure instead of three, and has goal walls that are supportive in this scenario. Therefore, it can be assumed that the goalkeeper

has a much easier task to learn than the striker. To compensate for this, a wider initial state distribution is determined for the goalkeeper. This is to prevent the goalkeeper agent from quickly getting into a dominant position over the striker agent. This can also be seen in Figure 9. Here, the ball appears randomly on the blue line, which is significantly wider than the striker's line. Except for these changes, the same episode definition is used as in the experiments before, i.e. the use of a time limit and ending an episode when the ball is not reachable by any rod and comes to a stop.

6 Discussion

The learning behavior is compared on the basis of the achieved reward curves of the striker and goalkeeper agents. From this, conclusions about the training process will be drawn. Afterwards, observable strategies of the agents during the training will be described and discussed.

6.1 Quantitative Analysis

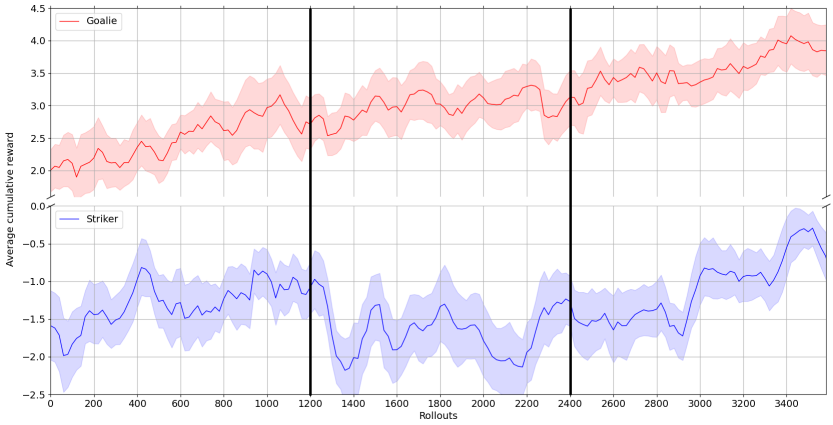

Fig. 10. Average cumulative reward per rollout for games 1, 2 and 3 of goalkeeper agents (red) and striker agents (blue). The shaded area represents the respective bootstrapped 95% confidence interval. The second game starts at rollout 1200 and the third at rollout 2400.

Figure 10 represents the run of all goalkeeper- and striker agents, each of which had three games. The first game ends at rollout 1200. One can immediately see a clear discrepancy between the goalkeeper and striker reward curves. Since the Markov game has been defined as a zero-sum game, the positive reward of an agent is equal to the negative reward of the opposing agent. The high discrepancy between the individual reward curves is an indication that the

behaviors to be learned are of different difficulty for the striker and goalkeeper rods. Since the goalkeeper consistently achieves a higher reward than the striker from the beginning, it can be assumed that despite the goalkeeper's more difficult initial state distribution, the problem is nevertheless easier for the goalkeeper compared to the striker. However, a positive learning behavior can be observed for both types of agents. The graph shows that despite the consistently high reward for the goalkeeper, it continued to increase over the course of the first game. The striker's reward curve also shows that the agents were able to achieve a significant increase in reward over the course of the training. It should be noted here that the respective training agents are trained with a stationary version of the opponent, which allows both reward curves to increase despite being modeled as a zero-sum game.
When looking at the second game, which starts at rollout 1200 and ends at rollout 2400, one can see a slight drop in the reward for the goalkeeper agent and a significant drop for the striker. The goalkeeper agents were able to increase the reward after the initial drop, while the rewards of the striker agents fluctuate much more in comparison. This may indicate that the strategy learned by the striker agent in the first game is not effective against the trained goalkeeper. However, the increase in the striker's reward in the second game may indicate that the learned behavior from game 1 could be adapted to the trained goalkeepers. The steady reward increase of the goalkeeper can be interpreted that even after changing the opponent, the learned strategies did not need much change.
Looking at the third game from rollout 2400 until 3600 one can see a strong increase of rewards for goalkeeper- and striker agents. Additionally, at the beginning of the game only the striker agents experience a slight initial reward drop, but recover after about 200 rollouts. The absence of this strong reward drop might be an indication that the learned strategies are more resilient against changing opponents. The goalkeeper agents were able to continuously increase the reward, following the trend from the games before. Comparing this to the striker agents, one can see that the striker reward curves have a higher variance and seem generally more fluctuating, which can indicate that the goalkeeper strategies seem more robust compared to the striker strategies.

6.2 Qualitative Analysis

In this section, observed strategies of striker and goalkeeper agents are shown. These were recorded after the training and can be viewed at [16]. The strategies observed can be classified into offensive and defensive categories. In this regard, the goalkeeper agents will be discussed first.
Concerning the defensive strategies, one could see that the goalkeeper agent considered the ball- as well as striker agent position in order to position itself in between both, to avoid a direct shoot of the striker agent, as illustrated in Figure 11 (1). Regarding the offensive strategies 11 (2), the goalkeeper learned to shoot the ball quickly after the start of an episode in between the striker rod's players. One can observe a significant shooting speed, which can be, depending on the spawn position of the ball, very difficult to catch for the striker agent.

Additionally, a special shooting technique could be observed, which shows the exploitation of the approximated simulation. When the ball was positioned in front of the goalkeeper figure, the goalkeeper rod would position itself above the ball, followed by a quick front-to-back rotation. This front-to-back rotation made it possible to achieve particularly fast shots.

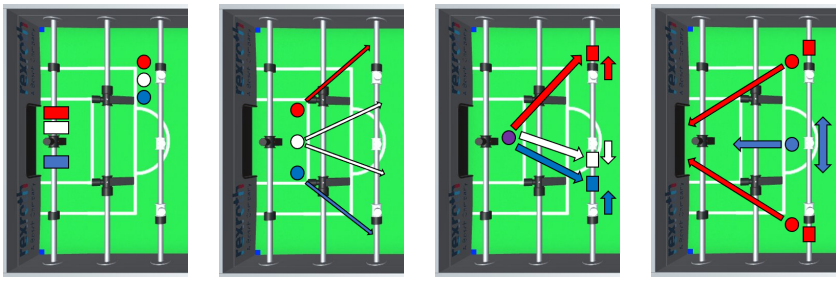

Fig. 11. Observed offensive and defensive strategies regarding goalkeeper- and striker agent. From left to right: (1) defensive strategy goalkeeper (2) offensive strategy goalkeeper (3) defensive strategy striker (4) offensive strategy striker

Regarding the defensive strategies of the striker rod 11 (3), one could see that the rod tries to catch the ball with the most close figure, but is often too slow to catch the fast shots of the goalkeeper agent. Interestingly, one could see that the striker agent learned precise ball control skills, which are able to stop a fast ball with a striker figure, damping its speed significantly. This can also be observed in cases when the striker rod is not able to catch the ball with a figure, but the ball travels very close to it. Then it can be seen, that the striker rod stops its slight rotations completely.
Finally, looking at the offensive strategies of the striker 11 (4), one could see that if the ball spawned at the left or right figure in a convenient position, the striker agent immediately shoots at the goal. However, if the ball spawns in front of the middle player, the striker agent does not shoot immediately. Instead, one could observe a up-and-down lateral movement of the striker rod, followed by a shoot several seconds later. One explanation for this could be, that the goalkeeper is right in the way when spawning before the middle player, which makes waiting a good choice until the goalkeeper agent moves out of the way.

7 Conclusion and Future Work

Considering the results, one can see that multi-agent training can be successful with limited computational resources, thereby allowing an ensemble of striker and goalkeeper agents to be trained with each other. Despite the observed imbalance in difficulty levels for striker and goalkeeper agents, both types of agents exhibited positive learning behavior, displaying the ability to learn advanced behaviors that consider the opponent's actions. In addition, one could observe an

increasing stability in the reward curves for both agent types, indicating learned behaviour that is more robust against changing opponents. Our work extends [27, 9], where we were able to train agents with a more complex action space [27] in a shorter time [9] respectively, while also training in a more complex Markov game environment with changing opponents. It can be concluded that the designed training concept is suitable for the Foosball domain. However, as shown in section 4.4, hyperparameters can have a large impact on the training behavior of the agents. Here, the impact of optimized hyperparameters with respect to multi-agent training would be an interesting aspect to explore. Additionally, it should be investigated to what extent the imbalance in difficulty levels can be compensated by longer training or changed environmental conditions, such as a broader initial state distribution.
Further work can also look at using imitation learning [7] or pretraining [2], training agents to an initial strength, followed by training in multi-agent environments. This could potentially increase training speed as environment dynamics do not need to be learned from scratch in a complex multi-agent Markov game.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. CoRR **abs/1907.10902** (2019). URL http://arxiv.org/abs/1907.10902
2. Anderson, C.W., Lee, M., Elliott, D.L.: Faster reinforcement learning after pretraining deep networks to predict state dynamics. In: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–7 (2015). DOI 10.1109/IJCNN.2015.7280824
3. Andrychowicz, M., Raichuk, A., Stanczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., Bachem, O.: What matters in on-policy reinforcement learning? A large-scale empirical study. CoRR **abs/2006.05990** (2020). URL https://arxiv.org/abs/2006.05990
4. Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., Mordatch, I.: Emergent complexity via multi-agent competition. CoRR **abs/1710.03748** (2017). URL http://arxiv.org/abs/1710.03748
5. Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research **47**, 253–279 (2013). DOI 10.1613/jair.3912. URL https://doi.org/10.1613\%2Fjair.3912
6. Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., de Oliveira Pinto, H.P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., Zhang, S.: Dota 2 with large scale deep reinforcement learning. CoRR **abs/1912.06680** (2019). URL http://arxiv.org/abs/1912.06680
7. Cai, X.Q., Ding, Y.X., Chen, Z.X., Jiang, Y., Sugiyama, M., Zhou, Z.H.: Seeing differently, acting similarly: Imitation learning with heterogeneous observations (2021). DOI 10.48550/ARXIV.2106.09256. URL https://arxiv.org/abs/2106.09256
8. Davis, J., Hsieh, Y.H., Lee, H.C.: Humans perceive flicker artifacts at 500 hz. Scientific Reports **5**(1) (2015). DOI 10.1038/srep07861. URL https://doi.org/10.1038/srep07861

9. De Blasi, S., Klöser, S., Müller, A., Reuben, R., Sturm, F., Zerrer, T.: Kicker: an industrial drive and control foosball system automated with deep reinforcement learning. Journal of Intelligent & Robotic Systems **102**(1), 1–18 (2021)
10. Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., Madry, A.: Implementation matters in deep policy gradients: A case study on PPO and TRPO. CoRR **abs/2005.12729** (2020). URL https://arxiv.org/abs/2005.12729
11. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. CoRR **abs/1709.06560** (2017). URL http://arxiv.org/abs/1709.06560
12. Hossain, M.R., Timmer, D.: Machine learning model optimization with hyper parameter tuning approach. Global Journal of Computer Science and Technology (2021)
13. Janssen, R., de Best, J., van de Molengraft, R.: Real-time ball tracking in a semi-automated foosball table. In: J. Baltes, M.G. Lagoudakis, T. Naruse, S.S. Ghidary (eds.) RoboCup 2009: Robot Soccer World Cup XIII, pp. 128–139. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
14. Kanervisto, A., Scheller, C., Hautamäki, V.: Action space shaping in deep reinforcement learning. CoRR **abs/2004.00980** (2020). URL https://arxiv.org/abs/2004.00980
15. Kim, J.T., Ha, S.: Observation space matters: Benchmark and optimization algorithm. CoRR **abs/2011.00756** (2020). URL https://arxiv.org/abs/2011.00756
16. Kitaird: kitaird/sai-foosball-agents: Initial release (2022). DOI 10.5281/ZENODO.7199993. URL https://zenodo.org/record/7199993
17. Lyu, X., Xiao, Y., Daley, B., Amato, C.: Contrasting centralized and decentralized critics in multi-agent reinforcement learning. CoRR **abs/2102.04402** (2021). URL https://arxiv.org/abs/2102.04402
18. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015). DOI 10.1038/nature14236. URL https://doi.org/10.1038/nature14236
19. Ng, A.Y., Harada, D., Russell, S.: Policy invariance under reward transformations: Theory and application to reward shaping. In: In Proceedings of the Sixteenth International Conference on Machine Learning, pp. 278–287. Morgan Kaufmann (1999)
20. OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., Zhang, L.: Solving rubik's cube with a robot hand (2019). DOI 10.48550/ARXIV.1910.07113. URL https://arxiv.org/abs/1910.07113
21. OpenAI, Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., Zaremba, W.: Learning dexterous in-hand manipulation (2018). DOI 10.48550/ARXIV.1808.00177. URL https://arxiv.org/abs/1808.00177
22. Pardo, F., Tavakoli, A., Levdik, V., Kormushev, P.: Time limits in reinforcement learning. CoRR **abs/1712.00378** (2017). URL http://arxiv.org/abs/1712.00378
23. Potter, M.C., Wyble, B., Hagmann, C.E., McCourt, E.S.: Detecting meaning in RSVP at 13 ms per picture. Attention, Perception, &amp Psychophysics **76**(2),

270–279 (2013). DOI 10.3758/s13414-013-0605-z. URL https://doi.org/10.3758/s13414-013-0605-z

24. Raffin, A.: Rl baselines3 zoo. https://github.com/DLR-RM/rl-baselines3-zoo (2020)
25. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research **22**(268), 1–8 (2021). URL http://jmlr.org/papers/v22/20-1364.html
26. Reda, D., Tao, T., van de Panne, M.: Learning to locomote: Understanding how environment design matters for deep reinforcement learning. CoRR **abs/2010.04304** (2020). URL https://arxiv.org/abs/2010.04304
27. Rohrer, T., Samuel, L., Gashi, A., Grieser, G., Hergenröther, E.: Foosball table goalkeeper automation using reinforcement learning. In: T. Seidl, M. Fromm, S. Obermeier (eds.) Proceedings of the LWDA 2021 Workshops: FGWM, KDML, FGWI-BIA, and FGIR, Online, September 1-3, 2021, *CEUR Workshop Proceedings*, vol. 2993, pp. 173–182. CEUR-WS.org (2021). URL http://ceur-ws.org/Vol-2993/paper-17.pdf
28. Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation (2015). DOI 10.48550/ARXIV.1506.02438. URL https://arxiv.org/abs/1506.02438
29. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR **abs/1707.06347** (2017). URL http://arxiv.org/abs/1707.06347
30. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016). DOI 10.1038/nature16961. URL https://doi.org/10.1038/nature16961
31. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T.P., Simonyan, K., Hassabis, D.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm. CoRR **abs/1712.01815** (2017). URL http://arxiv.org/abs/1712.01815
32. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., Hassabis, D.: Mastering the game of go without human knowledge. Nature **550**(7676), 354–359 (2017). DOI 10.1038/nature24270. URL https://doi.org/10.1038/nature24270
33. Tang, Y., Agrawal, S.: Discretizing continuous action space for on-policy optimization. CoRR **abs/1901.10500** (2019). URL http://arxiv.org/abs/1901.10500
34. Weigel, T., Nebel, B.: KiRo – an autonomous table soccer player. In: RoboCup 2002: Robot Soccer World Cup VI, pp. 384–392. Springer Berlin Heidelberg (2003). DOI 10.1007/978-3-540-45135-8_34. URL https://doi.org/10.1007/978-3-540-45135-8_34
35. Wiering, M., van Otterlo, M. (eds.): Reinforcement Learning. Springer Berlin Heidelberg (2012). DOI 10.1007/978-3-642-27645-3. URL https://doi.org/10.1007/978-3-642-27645-3
36. Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A.M., Wu, Y.: The surprising effectiveness of MAPPO in cooperative, multi-agent games. CoRR **abs/2103.01955** (2021). URL https://arxiv.org/abs/2103.01955

37. Zhang, D., Nebel, B.: Learning a table soccer robot a new action sequence by observing and imitating. In: J. Schaeffer, M. Mateas (eds.) Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference, June 6-8, 2007, Stanford, California, USA, p. 61. The AAAI Press (2007)
38. Zhong, Y., Zhou, Y., Peng, J.: Efficient competitive self-play policy optimization (2020). DOI 10.48550/ARXIV.2009.06086. URL https://arxiv.org/abs/2009.06086