Machine Learning Approaches for Attacking Behaviors in Robot Soccer

Justin Rodney

Alfredo Weitzenfeld

University of South Florida

ABSTRACT

In this work, Machine Learning approaches were applied to attacking behaviors in RoboCup Small-Size League autonomous robot soccer. Neural networks were used in order to get a binary prediction of an attacking action's success, while deep reinforcement learning was leveraged to learn low level skills which control the robot's wheel speeds and kicker. A trained neural network was used to predict whether a shot would be successful, improving the number of goals scored by the attacking behavior by 84 to 186%. The reinforcement learning methodologies used in this work produced behaviors which were efficient in speed, beating manually programmed behaviors in time taken, but can benefit from future refinements to improve accuracy in shooting towards goal.

Keywords: AI, Deep Reinforcement Learning, SSL, Robot Soccer.

1. Introduction

In Small-Size League (SSL)[1] robotics soccer, decision making is integral to a team's performance. This work focuses on two independent methods of applying machine learning (ML) to decision making; using neural network (NN) predictors to predict the success of a goal and using deep reinforcement learning (RL) to learn skills, or low-level behaviors, which control a robot's low level actions.

Robots playing offense may be faced with choosing between shooting to score or passing to a teammate. Plays oriented around passing are recognized as effective strategies in SSL games, highlighting the significance of this decision[2]. Classical methodologies used by SSL teams[3, 2] to determine whether a robot should shoot the ball towards a goal utilize analytical geometry; leveraging functions and algorithms which may make use of the distance from the ball towards the goal and the area between goal posts unobstructed by opponents. Examples of these classical methods may be visualized in Figure 1. ML methodologies, such as linear logistic regressors (LLR) or NNs, can be applied to a collection of shots to learn functions or train models for predicting the outcome of a shot. This work focuses on training a NN for predicting the success of a goal, with similar accuracies to related works[4]. Additionally, this work provides a comparison against a naive LLR and analyzes the performance impact of deploying a trained NN into an attacking behavior.

RL has been applied to SSL to learn multi-agent control strategies[5], where the action space consists of picking between predefined behaviors, as well as low-level skills[6]. This work follows the methodology proposed by Schwab et al.[6], employing Deep RL to train models which control the robot's low-level actions, such as velocities and kicking mechanisms.

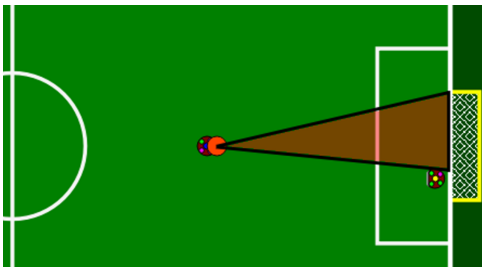

Figure 1: Visualization of the classical methods utilized in shooting towards goal.

2. Related Works

In the work of Naito et al.[4], the KIKS SSL team used NN predictors for predicting the success of passing and shooting towards goal, achieving an accuracy of 84% on testing data. Their data set consists of more than one hundred data points sampled from grSim, although it is not specified if each sample is taken from an independent shot. Their testing set contains 50 samples. The NN architecture deemed most successful by KIKS has 3 fully connected layers, with batch normalization and affine layers between the dense layers, with its input being the velocity of the ball, distance between the ball and an opponent, as well as the angle between the ball and an opponent. A NN may be advantageous to use in place of a LLR in order to learn from non-linear relationships between the features of the data, however, the work of KIKS does not provide a comparison of the performance of their NN against any other ML schemes.

Schwab et al.[6] utilized the Deep Deterministic Policy Gradient (DDPG)[7] algorithm in order to learn low-level skills for SSL soccer. In their work, they were able to train NN models which learned skills for traveling towards a ball, as well as aiming and shooting towards goal. Schwab et al.[6] adopted the methodology of burning in the replay buffer from DDPG from Demonstrations (DDPGfD)[8]; accomplished by filling up the replay buffer with samples obtained from demonstrations of a good behavior. DDPGfD employs other modifications to DDPG, such as the use of a prioritized replay buffer and n-step returns[8]. Other variations of DDPG, such as Distributed Distributional DDPG (D4PG)[9], share the use of these additions and have shown success in learning, even when faced with sparse rewards. The work of Schwab et al.[6] does not mention adopting any additional modifications to the original DDPG other than replay buffer burn-in.

The DDPG algorithm is an actor critic algorithm used for deep RL in continuous action spaces[7]. DDPG is trained off-policy with a replay buffer; therefore, the networks can be updated using experiences obtained at any time during its training. Throughout the training process, the agent acts within an environment and stores experiences inside of the replay buffer. The experience consists of a state, s, action, a, reward obtained at the following state, r, and the following state s'. Thus, the experience can be defined as (s,a,r,s'). During the training process noise may be added to the actions. Additionally, early on in the training process entirely random actions may be taken in order to encourage exploration[10]. The critic, Q, is updated using the mean squared Bellman error function in Equation (2). DDPG makes use of target networks, which have delayed updates to improve stability in learning, in order to approximate the Bellman equation, defined in Equation (1)[7].

$$y_t = r + \gamma Q_{\text{targ}}(s', \mu_{\text{targ}}(s')) \tag{1}$$

$$\text{loss} = \frac{1}{|B|} \sum_{(s,a,r,s') \in B} (Q(s,a) - y_t))^2 \tag{2}$$

The actor, μ, is updated with the loss function defined in Equation (3), performing gradient ascent on the output of the critic.

$$\text{loss} = -\frac{1}{|B|} \sum_{s \in B} (Q(s, \mu(s)) \tag{3}$$

At a high level, the critic learns to estimate all future rewards from taking an action in a given state, while the actor learns to take actions in order to maximize future rewards.

rSoccer[11] provides an open-source RL framework for SSL and Very Small Sized Soccer (VSSS) leagues with the goal of allowing for more research into this area. rSoccer facilitates the applications of RL to SSL by providing a SSL simulator suited for RL, as well as providing framework for creating OpenAIGym[12] environments.

3. Method

This work was done using the RoboBulls software system with grSim[13], a SSL soccer simulator, as the simulated environment for data collection for the NN predictor, as well as final benchmarks. The deep RL models were trained using the rSoccer[11] framework, and loaded into the RoboBulls software system as TorchScript modules. Similarly, the NN predictors were trained using TensorFlow in python, and loaded via the Tensorflow C++ API.

In this work, the RoboBulls Attack Main behavior was used for comparison against the behaviors with ML applications. The baseline behavior determines whether it can shoot by breaking the goal area into discrete segments, visualized in Figure 2, and then checking whether each segment is deemed a clear shot. Each segment is determined clear or not clear by evaluating if an opponent robot is within a certain distance from the straight line between the ball and goal segment. This can be visualized in Figure 3. After conducting this process on each segment, the largest group of adjacent segments is chosen, and the mid-point is used as the target point for the attacking robot to shoot. If there is no clear segment, the robot will not shoot. In order to improve upon the methodology of the baseline, other factors, such as the distance and angle to the goal, may be used in order to predict whether or not the shot will be successful[2].

In order to train a model for predicting the success of a shot, a data-set of 542 shots was collected for training, with 201 shots

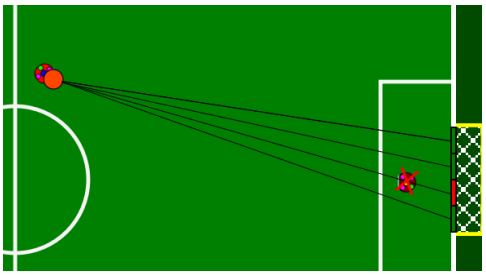

Figure 2: Visualization of checking clear shots by segments, using a small number of segments for clarity[14].

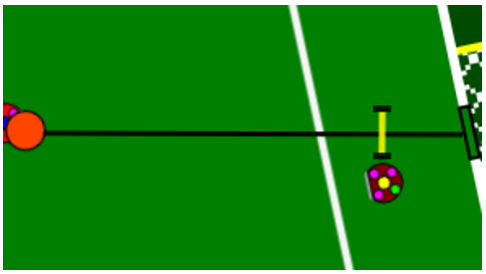

Figure 3: Visualization of distance tolerance for detecting a clear shot[14].

collected for testing. These shots were collected utilizing a single attacker and a single goalie. Information about the orientations of the two robots, as well as the target goal point was obtained when the attacking robot issued the kicking command. The success of the goal was stored when the ball entered the goal area or made contact with the goalie. In order to speed up the data collection process, due to the RoboBulls goalie being proficient at defending against a single attacker, the goalie was restricted to operating at half-speed. Two scenarios were used to collect the data in order to collect shots taken with a variety of positions. In the first scenario, the attacking robot started on its home side of the field with the ball, and was allowed to travel freely towards the opposing team's goal area, shooting when it came within a determined range of the goalie. Within this scenario, two ranges were utilized: mid-range and close-range. Due to potential biases in the robot's chosen path, collecting shots only utilizing this scenario could result in the data containing only shots taken from similar positions. In the second scenario, the robot was placed in the opposing team's side of the field within varying sections of the field, and was allowed to shoot as long as it did not leave its section. In the second scenario, the robot remained relatively stationary. More details regarding the data collection process can be found alongside the full analysis of this research[14].

After collecting the data-set a NN architecture similar to KIKS[4] was utilized, which can be seen in Table 1. The model was trained using 33% of the training data as a validation set for early stopping. The NN would take the features in Equation (4) as input.

$$\text{input} = (\theta_{\text{robot}-\text{opp}}, d_{\text{goal}}, d_{\text{opp}}, P'_{x_{\text{opp}}}, P'_{y_{\text{opp}}}) \tag{4}$$

$\theta_{robot-\text{opp}}$ is the difference between the orientation of the robot and the angle to the opponent robot, d_{goal} is the distance between the ball and target goal point, d_{opp} is the distance between the shooting and opponent robots, and $P'_{x_{\text{opp}}}$ and $P'_{y_{\text{opp}}}$ is the location of the opponent relative to the shooting robot. In addition to training a NN, stratified 10-fold cross validation[15] was repeated 10 times alongside the corrected resampled paired t-test[16] in order to provide

Table 1: Neural network architecture.

Layer	# Units	# Activation
Batch Normalization	-	-
Dense	6	ReLU
Batch Normalization	-	-
Dense	3	Swish
Output	1	Sigmoid

a comparison against a LLR. The NN architecture was compared against the LR twice, using 33% and 25% of its allocated training split for early stopping. Reducing the size of each split used for early stopping was done to mitigate the issues of the NN starving itself of data, as it would already be trained on limited data.

The trained model was then deployed into the Attack Main behavior, creating the AttackNN behavior. In addition to the distance tolerance check used in the baseline behavior, the NN predictor was used to evaluate segments deemed clear, disqualifying any clear segment that the NN deemed would result in an unsuccessful goal.

Two low-level skills were learned by using DDPG: Go-To-Ball and Shoot-Towards-Goal. The state, action, and reward definitions were used and modified from CMU's work[6]. The equations utilized variables obtained in relation to the robot. Spinning Up's[10] implementation of DDPG was utilized, with modifications to allow for exporting TorchScript modules.

The Go-To-Ball skill's state definition, seen in Equation (5), utilizes the difference in orientation between the robot and the ball, θ_{ball}, the distance to the ball, d_{ball}, and the linear and angular velocities of the robot, $V_{x_{\text{robot}}}, V_{y_{\text{robot}}}$, and $V_{\theta_{\text{robot}}}$. The action space, defined in Equation (6), consists of controlling the robot's linear and angular velocities. The reward function, in Equation (10), consists on a reward based upon the distance to the ball, whether or not the robot is within 200 meters of the ball, and the difference between the robot's orientation and the angle to the ball. The definitions for the aforementioned equations are in Equations (7, 8, 9). The episode ended when the robot successfully came within 0.2 meters of the ball or when 9.9 seconds had elapsed.

$$s = (sin(\theta_{\text{ball}}), cos(\theta_{\text{ball}}), d_{\text{ball}}, V_{x_{\text{robot}}}, V_{y_{\text{robot}}}, V_{\theta_{\text{robot}}}) \tag{5}$$

$$a = (V_x, V_y, V_\theta) \tag{6}$$

$$r_{\text{contact}} = \begin{cases} 100 & d_{\text{ball}} \leq 0.2 \\ 0 & d_{\text{ball}} > 0.2 \end{cases} \tag{7}$$

$$r_{\text{distance}} = \frac{5}{\sqrt{2\pi}} exp(\frac{-d_{r-b}^2}{2}) - 2 \tag{8}$$

$$r_{\text{orientation}} = \frac{1}{\sqrt{2\pi}} exp(-2\frac{\theta_{r-b}}{\pi^2}) \tag{9}$$

$$r_{\text{total}} = r_{\text{contact}} + r_{\text{distance}} + r_{\text{orientation}} \tag{10}$$

The Shoot-Towards-Goal state space, defined in Equation (3), contains the position of the ball, velocity of the ball, distance to the mid-point of the goal area, as well as the orientation difference between the top and bottom goalposts. The action space, seen in Equation (11), contains the angular velocity of the robot and whether or not the robot should kick. The reward function contains a reward for facing the goal, as well as a sparse reward for kicking the ball towards goal, and penalties to avoid holding onto the ball. α represents the angle between the top and bottom goalposts in relation to the robot's position. β represents the difference in orientation between the robot and whichever goalpost results in a larger value. For Equation (12), the robot receives a reward in proportion to the ball's velocity when it is facing the goal, as well a penalty in proportion to the ball's velocity when facing away from the goal. Equation (13) was introduced to mitigate issues involving the local maxima present in Equation (12), where the robot will be penalized for any motion when facing away from the goal. Without this addition, learned skills would never orient the robot towards the goal, in order to avoid receiving a penalty. The robot receives a reward when it has kicked towards the goal, but is penalized when the robot is not facing the goal, or is facing the goal but is still holding onto the ball. The episode would end when the robot has successfully shot the ball into the goal area, unsuccessfully shot the ball outside of the field, or 3 seconds elapse.

Table 2: Results of the corrected resampled paired t-test. $\bar{a}$ is the mean accuracy of the NN, $\bar{b}$ is the mean accuracy of the LLR, $\bar{x}$ is the mean of the paired difference. The percentage of the NN's allocated training data used for early stopping is shown.

% Early Stopping	$\bar{a}$	$\bar{b}$	$\bar{x}$	s^2	t	p
33%	87.4024	84.4512	2.9512	18.5950	1.9666	0.0520
25%	87.5478	84.5923	2.9556	17.0759	2.0552	0.0425

$$s = (P_{x_{\text{ball}}}, P_{y_{\text{ball}}}, V_{x_{\text{ball}}}, V_{y_{\text{ball}}}, V_{\theta_{\text{robot}}}, d_{r-g}, sin(\theta_{\text{top}}), cos(\theta_{\text{top}}), sin(\theta_{\text{bottom}}), cos(\theta_{\text{bottom}}))$$

$$a = (V_\theta, K) \tag{11}$$

$$r_{\text{face-goal}} = \begin{cases} 0.05(\alpha - \beta)|V^B| & \alpha \geq \beta \\ (\alpha - \beta)|V^B| & \alpha < \beta \end{cases} \tag{12}$$

$$r_{kick} = \begin{cases} 10 & \alpha \geq \beta \wedge \text{Just Kicked} \\ \text{-0.25} & \alpha < \beta \vee \neg \text{ Kicked} \end{cases} \tag{13}$$

$$r_{\text{total}} = r_{\text{face-goal}} + r_{\text{kick}} \tag{14}$$

In order to test the performance of the learned skills, the skills were combined, with the addition of a manually programmed intermediate state for picking up the ball, to create the Go-To-Ball-And-Shoot behavior. The new behavior was tested against the Attack Main behavior and a human participant operating a robot with a video game controller. The robot was placed in its home side goalkeeper area, and would travel towards the ball and shoot towards the opposing team's unattended field goal. The ball was placed in two positions encompassing two trials; Trial 1, where the ball was placed in the center of the field, and Trial 2, where the ball was placed at the coordinates (0, 1.5), in meters.

4. Experiments and Results

The results of the corrected resampled paired t-test can be seen in Table 2. The comparison that utilized a NN which would hold aside 33% of its data for early stopping did not achieve a statistically significant difference between the performance of the NN and the LLR. However, the NN that would set aside 25% of its allocated training data did achieve a statistically significant difference.

The trained model which was selected achieved an accuracy of 85.07% on the testing data, with the confusion matrix seen in Table 3. Deploying the trained NN into the AttackNN behavior resulted in the attacking robot achieving 80 goals out of 100 attempted

Table 3: Neural network confusion matrix for results on the test set.

		Predicted	
		Failure	Success
Actual	Failure	102	20
	Success	10	69

Table 4: The average time taken for each skill, comparing RoboBulls baseline behaviors to the reinforcement learning implemented behaviors.

(a) Trial 1

Behavior	GoToBall		Shoot		Total		# Goals
	μ	s^2	μ	s^2	μ	s^2	
DDPG	2.9217	0.0008	0.2604	0.0027	3.1821	0.0048	23
Basline	3.4998	0.0040	0.2665	0.0006	3.7662	0.0044	25
Human	3.483	10.9620	0.4032	0.1709	3.8862	12.8650	25

(b) Trial 2

Behavior	GoToBall		Shoot		Total		# Goals
	μ	s^2	μ	s^2	μ	s^2	
DDPG	3.1262	0.0196	0.4031	0.0012	3.5294	0.0196	21
Baseline	4.9846	0.0141	0.4713	0.0057	5.4560	0.0193	25
Human	3.1691	0.3725	1.1236	0.3481	4.2927	1.1300	14

shots against the half-speed goalie. This is an 186% improvement in the amount of successful shots as compared to the baseline, which only achieved 28 of its 100 attempted goals. When facing the full-speed goalie, the AttackNN behavior had an 84% improvement, achieving 35 of 50 shots compared to the baseline only achieving 19 of 50.

The results of the trials to compare the RL learned behavior can be seen in Tables 4a and 4b. The RL learned behaviors consistently outperformed the baseline behavior and human participant in terms of time taken to complete each skill. The RL learned behaviors suffered in accuracy towards shooting towards goal, missing 5 goals as compared to the baseline which successfully scored each attempted shot. However, the RL behavior was more succesful than the human, which missed 11 shots. The variance in time taken for the RL behaviors are more similar to the hand-programmed behaviors than the human operator.

5. Conclusions

The adoption of a NN into the RoboBulls attacking behavior increased the number of successful goals against the half-speed goalie used in the data collection process by 186%, and the full-speed goalie by 84%. Comparing the performance of the NN approach against a naive LLR using repeated 10-fold cross validation and the corrected resampled paired t-test suggests that there may be advantages of using an NN predictor to learn non-linear relationships within the data, but future works should attempt to gather more data with less bias, and feature selection should be revisited in order to allow the NN to take advantage of additional inputs, such as the velocity of the opposing robots. In this work, a single goalie was utilized in the data collection process. However, shots could be collected from games or scenarios with more opponents to gather data from more dynamic situations.

The behaviors learned by the DDPG algorithm resulted in fast behaviors, outperforming baseline in time taken, but achieving lower shooting accuracies. Future works should employ the variations of DDPG that adopt n-step returns and prioritized replay buffer, in order to facilitate learning in more advanced scenarios that may use sparse rewards[8, 9]. In order to produce skills which may be utilized in SSL games, environments will need to be crafted around realistic scenarios in order to learn skills which take into account obstacle avoidance, to avoid colliding with robots on the field, and shooting against an opponent goalie. The reward definition for the Shoot-Towards-Goal skill has local maxima present in CMU's definition[6]; however, they likely mitigated this issue by leveraging demonstrations in the learning process. This suggests the use of DDPGfD[8] in future works to learn in the absence of meticulously engineered reward functions.

Acknowledgements

This work was funded in part by NSF IIS Robust Intelligence research collaboration grant #1703225 at the University of South Florida entitled "Experimental and Robotics Investigations of Multi-Scale Spatial Memory Consolidation in Complex Environments".

References

[1] "Robocup small size league." `https://ssl.robocup.org/`. Accessed Feb. 16, 2022 [Online].

[2] M. Geiger, C. Carstensen, A. Ryll, N. Ommer, D. Engelhardt, and F. Bayer, "Tigers mannheim (team interacting and game evolving robots) extended team description for robocup 2017," 2017.

[3] U. Robobulls, "Home." `http://usfrobobulls.org/`. Accessed Feb. 16, 2022 [Online].

[4] Y. Naito, S. Ohno, Y. Imaeda, A. Odanaka, Y. Tsuruta, R. Mitsuoka, T. Tane, M. Watanabe, and T. Sugiura, "Kiks extended team description for robocup 2020," 2020.

[5] F. Ollino, M. Solis, and H. Allende, "Batch reinforcement learning on a robocup small size league keepaway strategy learning problem," 03 2019.

[6] D. Schwab, Y. Zhu, and M. Veloso, "Learning skills for small size league robocup," in *RoboCup 2018: Robot World Cup XXII* (D. Holz, K. Genter, M. Saad, and O. von Stryk, eds.), (Cham), pp. 83–95, Springer International Publishing, 2019.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[8] M. Vecerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. A. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *CoRR*, vol. abs/1707.08817, 2017.

[9] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. Tb, A. Muldal, N. Heess, and T. Lillicrap, "Distributed distributional deterministic policy gradients," *arXiv preprint arXiv:1804.08617*, 2018.

[10] J. Achiam, "Spinning up in deep reinforcement learning," 2018.

[11] F. B. Martins, M. G. Machado, H. F. Bassani, P. H. M. Braga, and E. S. Barros, "rsoccer: A framework for studying reinforcement learning in small and very small size robot soccer," 2021.

[12] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[13] V. Monajjemi, A. Koochakzadeh, and S. S. Ghidary, "grsim – robocup small size robot soccer simulator," in *RoboCup 2011: Robot Soccer World Cup XV* (T. Röfer, N. M. Mayer, J. Savage, and U. Saranlı, eds.), (Berlin, Heidelberg), pp. 450–460, Springer Berlin Heidelberg, 2012.

[14] J. Rodney, "Analyzing decision-making in robot soccer for attacking behaviors," Master's thesis, 2022.
[15] R. Bouckaert and E. Frank, "Evaluating the replicability of significance tests for comparing learning algorithms," in *Advances in Knowledge Discovery and Data Mining*, vol. 3056, pp. 3–12, 01 2004.
[16] C. Nadeau and Y. Bengio, "Inference for the generalization error," *Machine Learning*, vol. 52, pp. 239–281, 01 2003.