RL Penalty Shooting

Group A34_14h_M

João Alves up202007614@up.pt

Marco André up202004891@up.pt

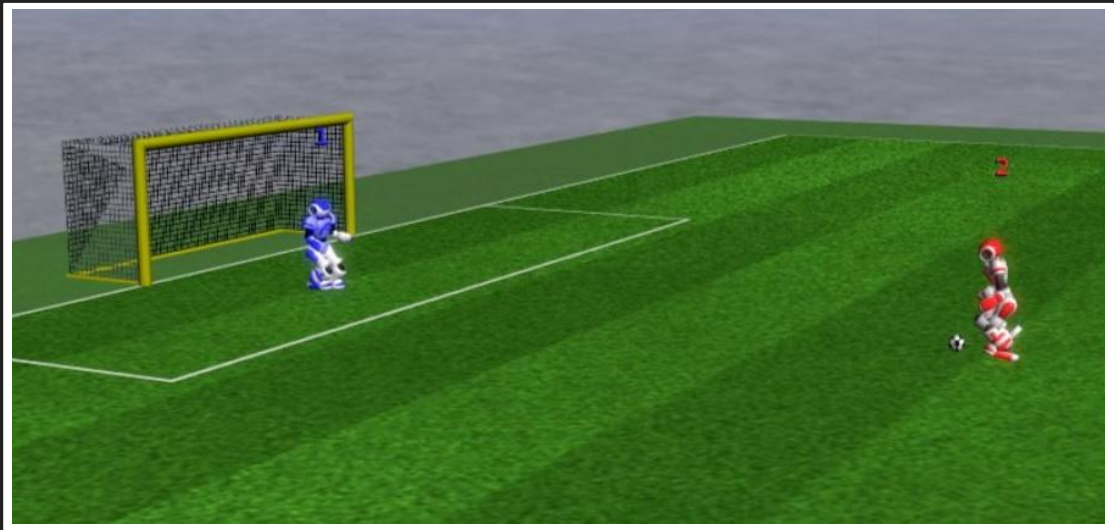
Rúben Monteiro up202006478@up.pt

RI @ FEUP | 2024/25

Project Description

Penalty Shooting Scenario using RL in *RoboCup 3D* Simulation

- ➔ Train an **attacker** player agent to do penalty kicks with RL
- ➔ Develop a basic heuristic-based **goalie** to defend the kicks



Objectives

Success is measured by meeting the following goals:

- ➔ Goalie behavior is relatively solid at defending basic shots
- ➔ Achieve a high percentage of successful goals with attacker agent via RL
- ➔ Attacker can use different ball trajectories by taking into account the defender’s position

Used Software

- ➔ RoboCup3D (simulator)
- ➔ Roboviz (visualization)
- ➔ FC Portugal (codebase and documentation)
- ➔ Python (programming language)
- ➔ OpenAI Gym (reinforcement learning)

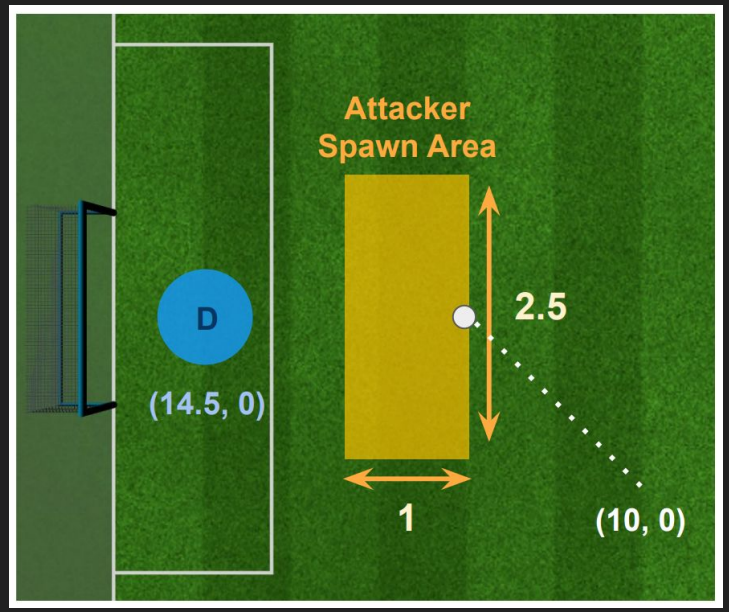
Developed Work

Start Position

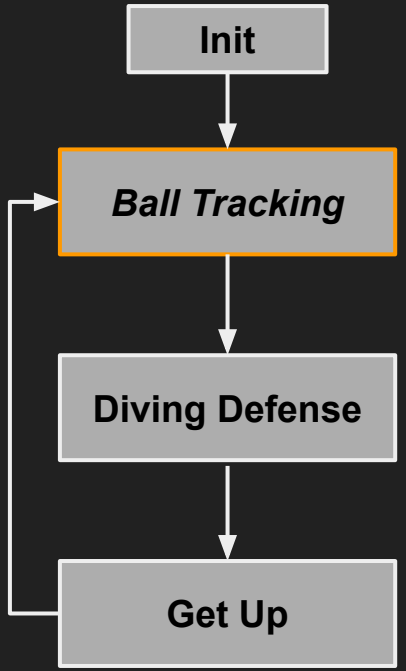
The goalie, *D*, always spawns in the same position: at the center of the goal.

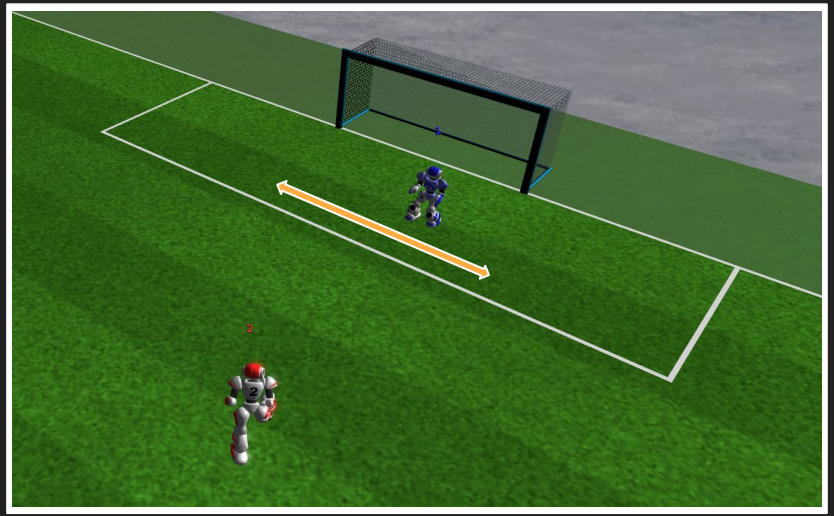
The attacker can randomly spawn in a predefined area near the goal, leading to:

- ➔ More robust agent behavior to starting position and pose
- ➔ Makes the training more challenging as the agents need to take relative positions, distances and angles into account



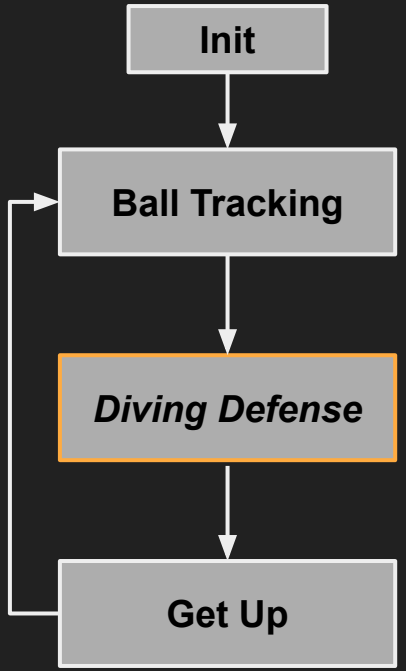
Defender State: Ball Tracking

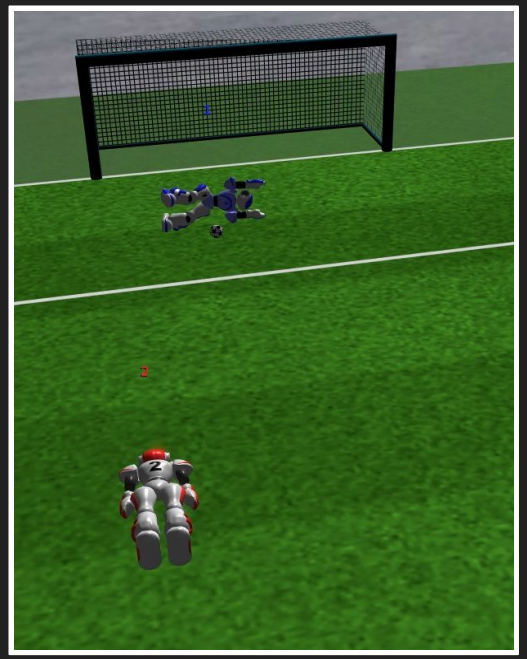




Default State: Agent moves sideways to place itself between the ball and the goal

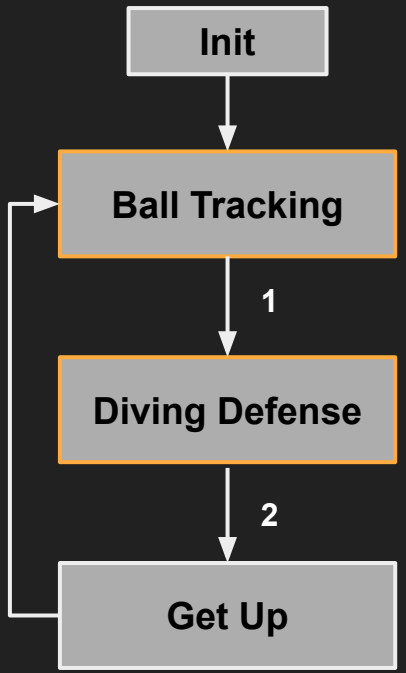
Defender State: Diving Defense





Agent dives sideways to defend based on the ball's speed and direction

Defender Transitions



Conditions for a transition to occur for the two main states:

1) If the ball is coming near (3m) with high speed (3m/s) and not straight at the goalie, it dives to the direction the ball is taking

2) After a dive, the agent waits a while before getting up to make sure it catches the ball.

Attacker Agent

Optimization Algorithm: OpenAI Gym with Proximal policy optimization (PPO)

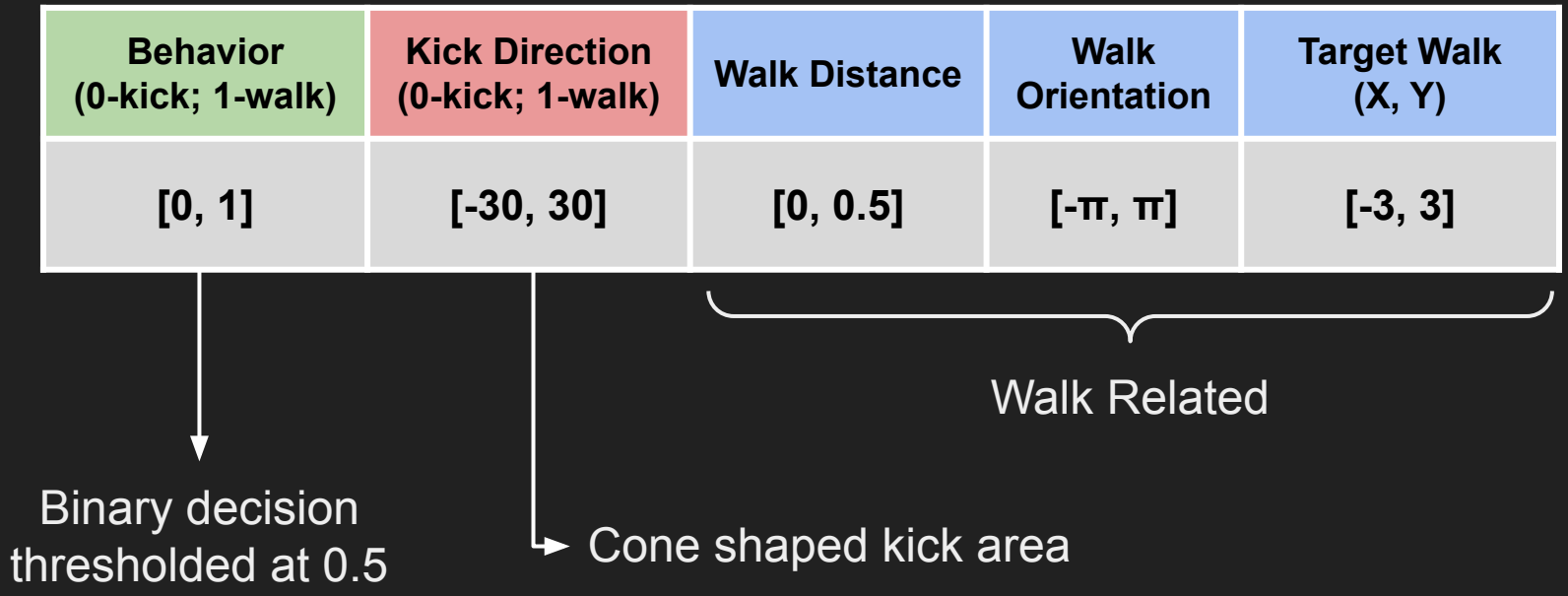
Observation Space: to reduce training time, only 8 values are known to agent:

Step Counter (Nº steps)	Last Action (0-kick; 1-walk)	Player Pos. (X, Y)	Ball Pos. (X, Y)	Player-Goal Vector (Width, Depth)

The values are all *float32*, from -infinity to +infinity.

Attacker Agent: Action Space

For the action space 6 values define the possible agent behaviors:



Attacker Agent: Reward Function

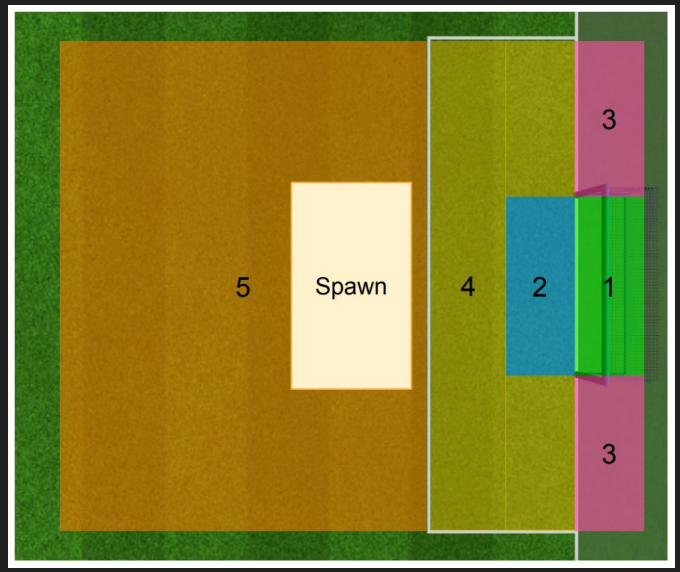
Some rewards are given to incentivize the action of kicking the ball as fast as possible:

- ➔ Default of -1 points to incentivize action
- ➔ Each simulation step not kicking ball is -5 points
- ➔ Staying close to ball yields +0.5 points per step
- ➔ Further penalize proportionally on ball-player distance if the agent moves away from ball without kicking
- ➔ If the agent falls without kicking the ball it receives a heavy punishment of -2000 points and the episode finishes early

Attacker Agent: Reward Function

The terminal episodes are defined from the highlighted areas as follows, from highest positive to most negative:

1: Goal	2500 + bonus the closer the ball is to the center
2: Almost Goal	500 * scaler based on distance to goal (max 1000)
3: Out of Bounds	-150 * horizontal to goal offset
4 / 5: Time Limit	Heavy punishments of -1500 * ball distance to goal



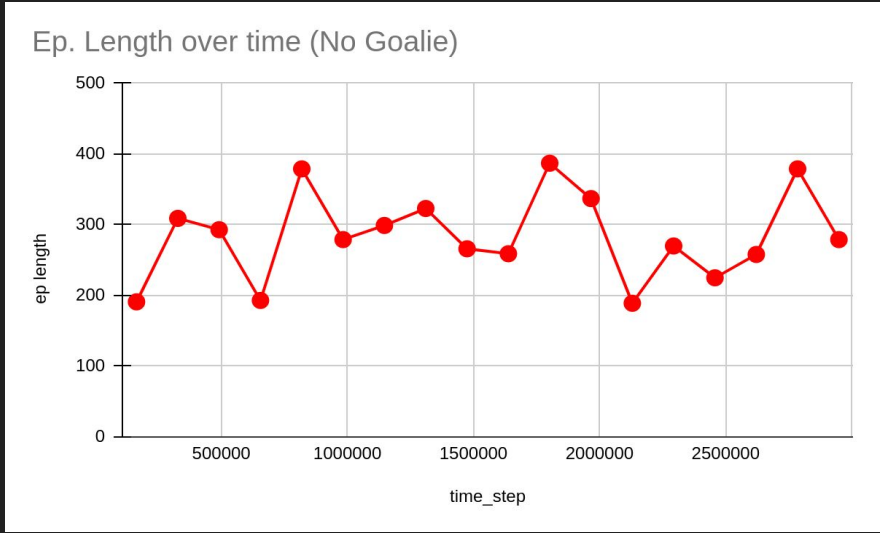
Results Evaluation

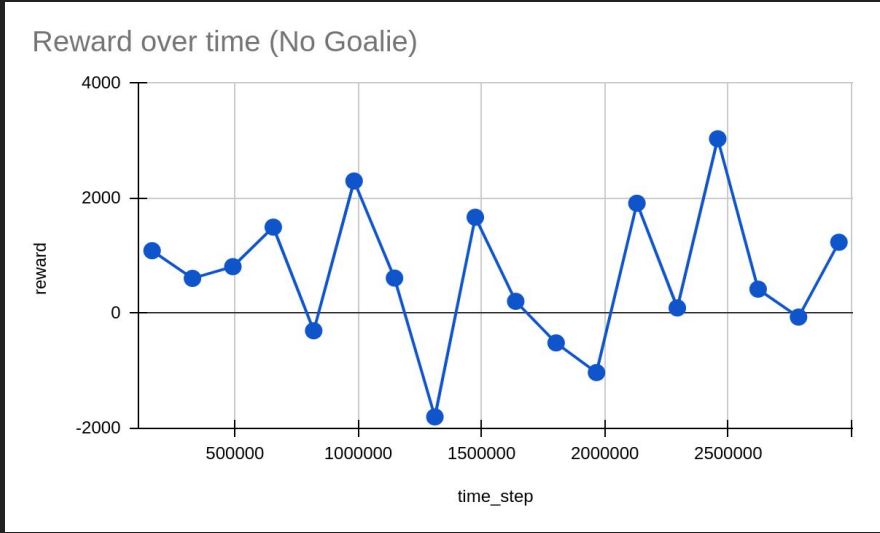
The team separately studied and evaluated three scenarios:

- ➔ Attacker without goalie
- ➔ Attacker with goalie (no RL adaptation to goalie)
- ➔ Attacker with goalie (with attacker RL adaptation)

The results were collected over many iterations to achieve more stable metrics. Due to relatively long RL training (each setup took ~4 hours to train), the exploration was limited

Attacker without Goalie Training

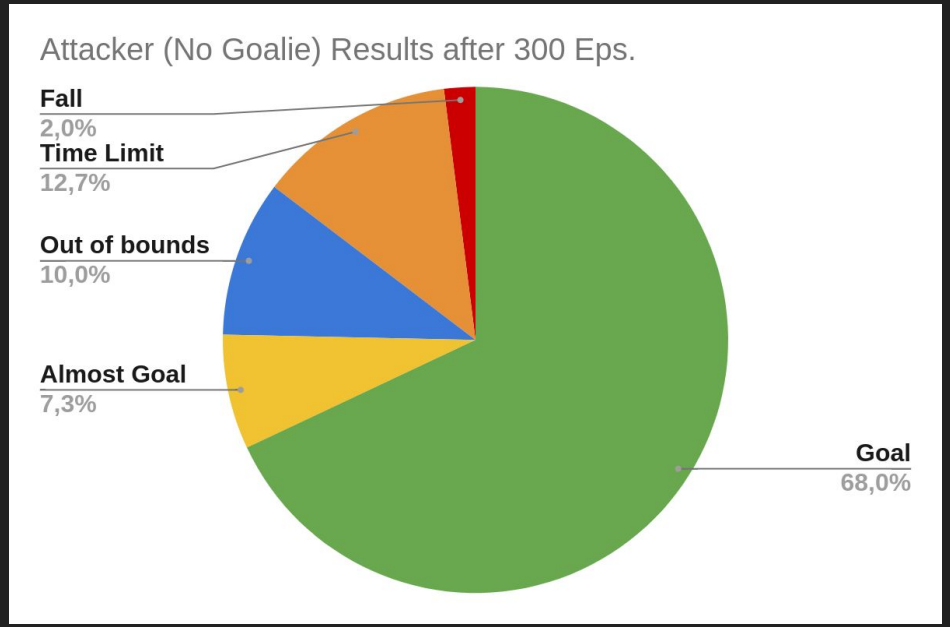




Total steps	3,000,000
Nº iterations	245
Train time	~ 4h

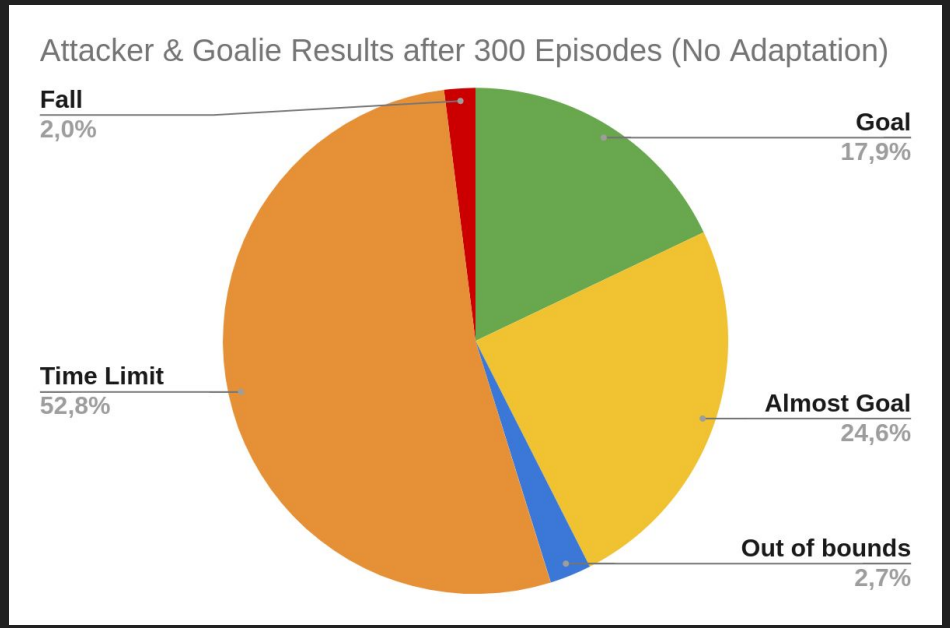
Attacker without Goalie Results

- ➔ The attacker scores or almost scores 75% of the time
- ➔ Less than 15% of the episodes were very punishing (fall or time limit)
- ➔ In 17% of the cases, the robot has the “right intention” but either the kick force or direction was not the appropriate one



Attacker with Goalie (No Adaptation)

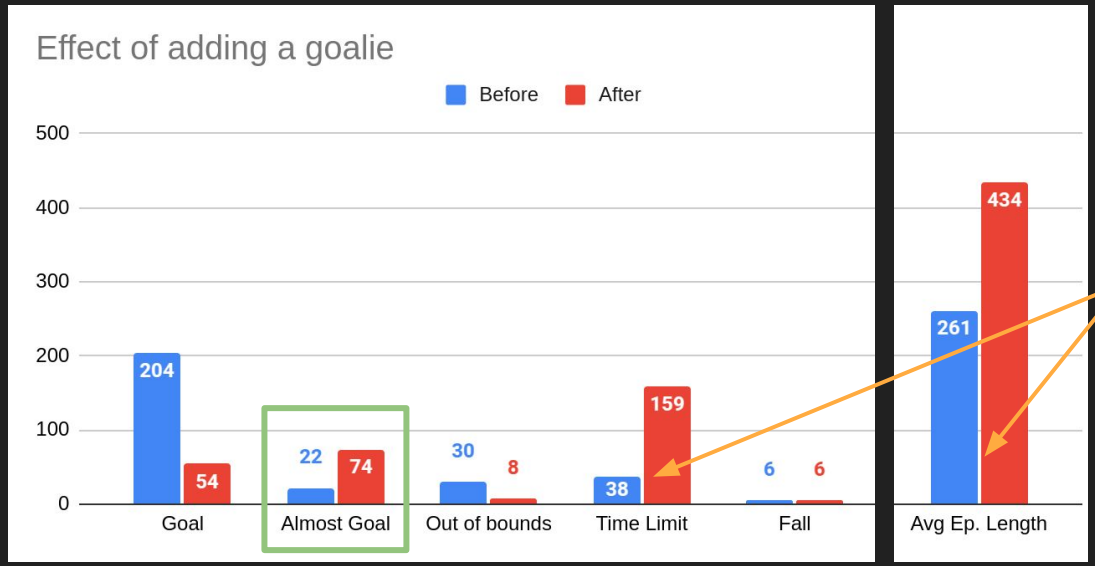
- ➔ We also experimented putting the attacker facing the goalie without further adaptation to its presence
- ➔ At first, the goalie was too efficient because the shots were mostly perpendicular to the goal and so virtually no goals were scored
- ➔ As such, we removed the goalie's option to not dive to defend a kick and got the results on the right



Note: The default codebase attacker has a success rate of ~70% against the same goalie

The effect of adding a goalie

The impact of adding a goalie to which the attacker wasn't prepared is very noticeable as the goal rate dropped by 75%, being replaced by **near misses**.



Because we didn't stop the episode after a defense to avoid false positives, naturally there was a big increase in episodes ended due to time limit.

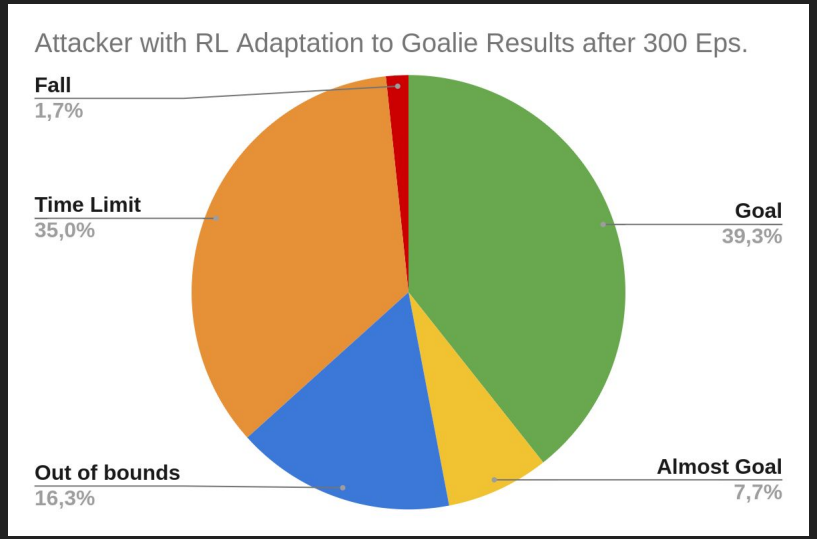
Training the Attacker with Goalie

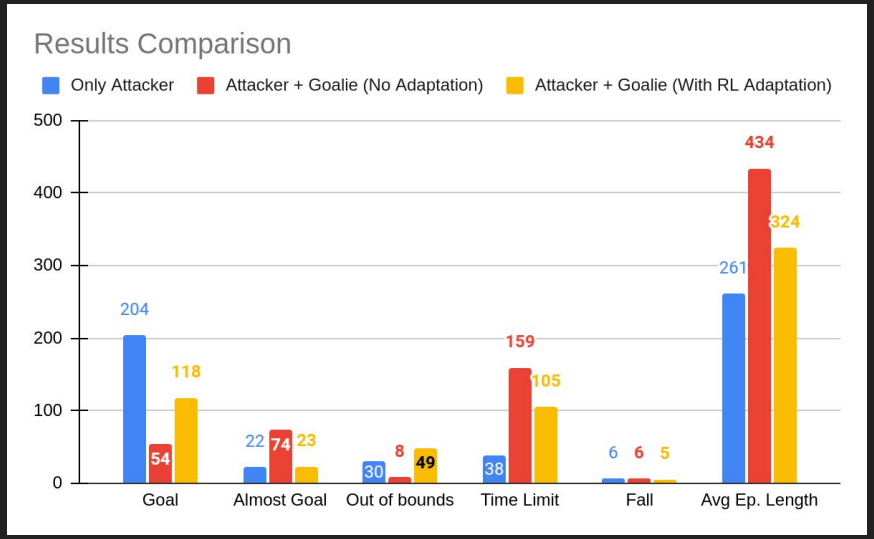
The following changes were made to train the attacker with the goalie's effect in consideration:

- **Observation Space**
  - Added the (X, Y) of the goalie to the observation space, making it 10 values in total
- **Rewards**
  - Aggravated some penalties for not kicking the ball to further reduce wandering around
  - Alleviated "the out of bounds" punishment
  - Attenuated "end due to time" punishment if the ball was within the the 2 poles in width
  - Added a big goal bonus reward the further the ball was from the goalie when crossing

Training the Attacker with Goalie

The results show that the robot managed to adapt relatively well to the goalie, recuperating a significant part of its performance back from when it had no defender. However, this was mostly achieved by exploiting the NAO's physical model design.





Development Limitations

During the development of project the following constraints were felt:

- ➔ Lack of powerful computational resources for training
- ➔ Limited time for testing each desired training setup
- ➔ Lost time trying to work with complex codebase
- ➔ Finicky adjustment of observation and action spaces and rewards for the agent
- ➔ The starting training state yielded dramatically different results

Conclusions and Future Work

Despite this being our first experience with a RL project and facing a big codebase, we deem the achieved results positive.

The final RL agent was capable of 40% goals against a moderately capable goalie, showing the viability of RL for many real-world tasks if trained properly.

Ideally we would like to have more time to experiment different setups and have the agent trained for longer periods to stabilize the results.

Future work could also include experimenting with different algorithms besides PPO and adopt more sophisticated RL strategies.

Demo Video Link:
https://youtu.be/fjvVxMVGUtA