Optimizing Goalkeeper Positioning

Mattia Danese and Evan Loconto

What problem did we address?

- RL in soccer mostly regards outfield play, but what about the goalkeeper..?
- The goalkeeper is arguably the most important position in a soccer team and a goalkeeper's effectiveness comes down to a few key skills:
  - Hands (i.e. catching)
  - Agility (i.e. diving and quickness)
  - Positioning
- Knowing where to be all times puts the goalie in the best position to save a shot from the opposing team
  - Having impeccable positioning can sometimes even make up for subpar catching or agility
- Given the position of an attacker, what is the best position for a goalkeeper to be in?

What RL methods did we use?

- Multi-Armed Bandit
  - Agent has *n* arms (actions)
  - Observes a state from the environment and will either:
    - Exploit → Pick the current best arm
    - Explore → Pick a random arm
  - Receives a reward based on the optimality of the arm picked
  - Stores *estimated* value of each arm (per state)
    - Essentially a Q-table
  - Updates *estimated* values based on reward and a constant step-size

$$Q_{n+1} = Q_n + \alpha[R_n - Q_n]$$

- Best fit for the structure of the problem
  - Episodic, but each episode is comprised of just one time step

What is the environment?

- Two perpendicular sides of a cube
  - The vertical side (XZ plane) represents the goal
  - The horizontal side or base (XY plane) represents the goal area
- State Space
  - Initial position of the attacker
- Action Space
  - All possible positions the goalie can be in
  - In theory: any position within the goal area
  - In practice: we discretized the goal area into 24x18 matrix (each cell is one square foot)
- Reward Function
  - Save → 0
  - Goal → -1

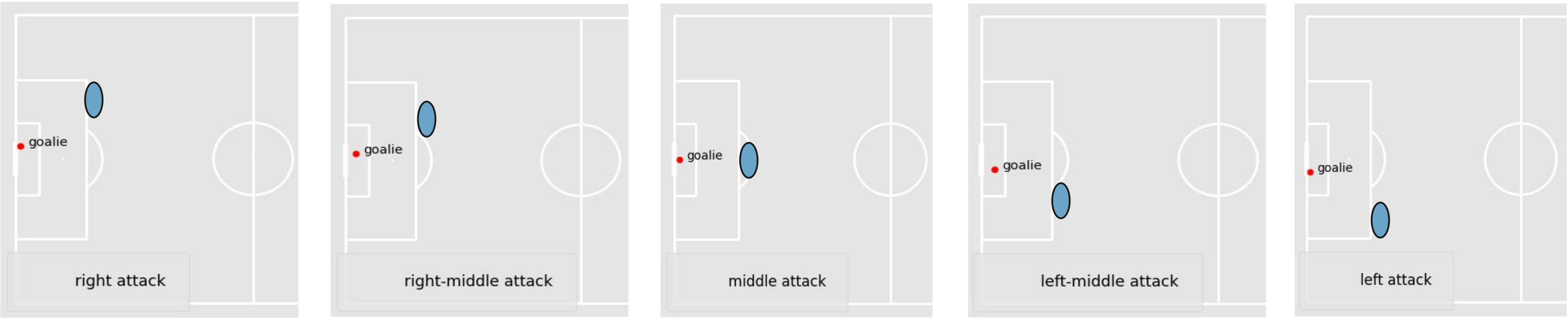
How did we train the agent?

```
Initialize attacker S, goalkeeper G, and step-size α:
     Q(a,s)← 0
     α ∈ (0,1]
Loop forever:
     P ← S.newPosition()
     G.newPosition(P)
     L ← S.newShotLocation()
     R ← G.saveOrNot(L)
     Q(a, P) ← Q(a, P) + α[R - Q(a, P)]
```

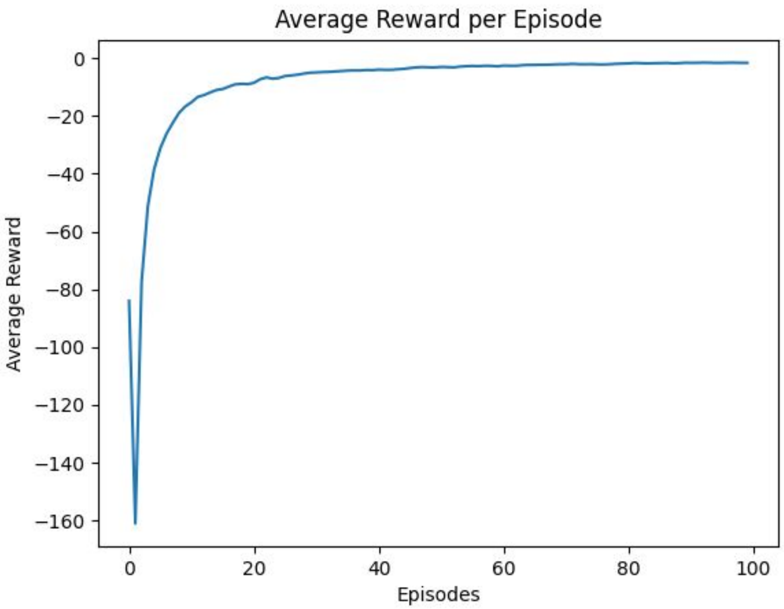
How did we train the agent?

- Attacker position randomized between 5 different locations
  - Each location has its own shot distribution
- Attacker shot location sampled from respective shot distribution
- Goalkeeper picks action (i.e. what position to be in) solely based on attacker position
  - Does not know shot distributions
- Based on Goalkeeper position, *savable area* of goal is calculated and used to assess if shot was saved or not
  - Goalkeeper casts shadow onto goal
  - If Goalkeeper is far out, the shadow increases in width but decreases in height
  - If Goalkeeper is closer to the goal line, the shadow increases in height but decreases in width

What were our results?



What were our results?



Where do we go from here?

- Quick Recap
  - The scope of our project was to optimize goalkeeper positioning
  - Implemented agent as a Multi-Armed Bandit
  - Results clearly show that the agent learned which positions are better (and worse)
- Next steps
  - Extend this experiment to 3D via RoboCup Simulator
  - Expand environment to additionally have more than one attacker or defenders

Thank You!

Any questions?