Evaluating State Action Space in Soccer using Deep Reinforcement Learning

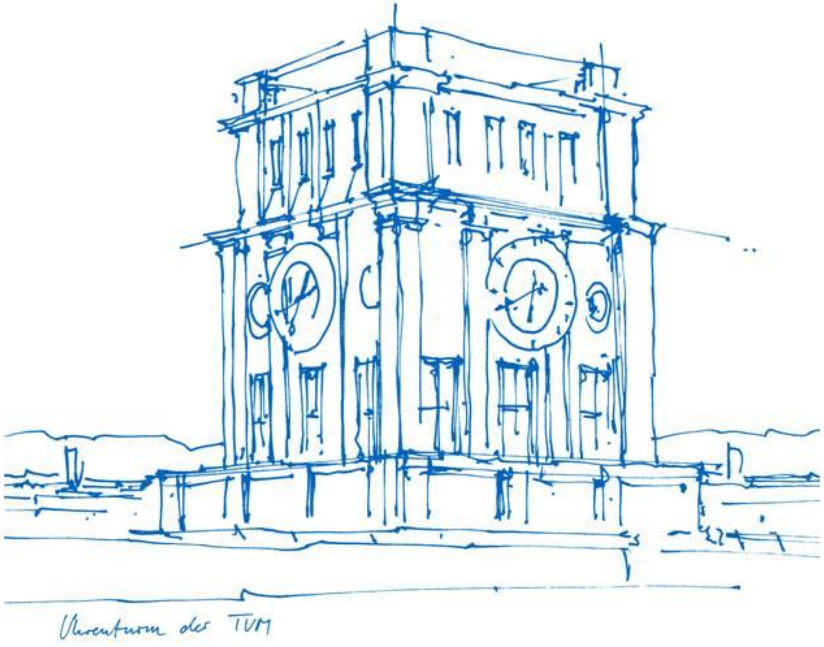
Guided Research Project

Md Siyam Sajeeb Khan

Technische Universität München

Supervised by

Marc Schmid and Marko Jovanović

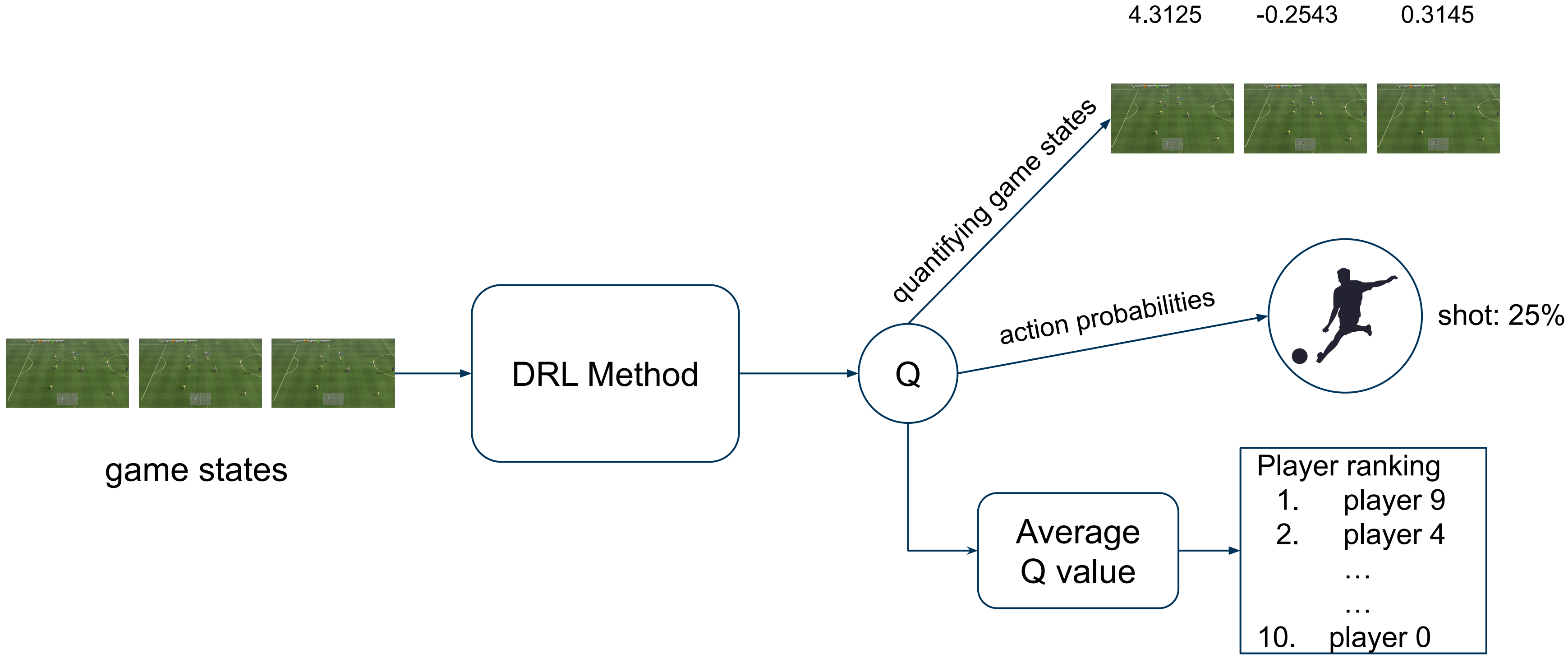


Outline

- Motivation
- Related work
- Dataset and environment
- Methodology
- Experiments and results
- Contribution and future work

Motivation

- Quantifying game states and actions in Soccer
- Use Deep Reinforcement Learning (DRL) to compute an action-value Q function
- Evaluate the quality of the state-action space using the Q-function
- Rating player performances and rank them



What is a game state?

Game State:

- Representation of an event of the game such as a normal pass, shot, tackle, etc. with tracking data including but not limited to
  - Player and ball positions
  - Player and ball directions
  - Active player, etc.

example game state: an attacker running with the ball

0	1	2	…	…	…	114
0.23	0.45	0.12	…	…	…	0.05

representation of the game state using 115-dimensional tracking data

Related work

- Expected Goals (xG)[1]
  - Incorporates shot information to rate shots based on the probability of goal
  - Considers angle to goal, player position, etc. for the calculation
- Expected Possession Value (EPV)[2]
  - Deep learning based solution
  - Measures the impact of individual Soccer players in different game scenarios
  - Requires tracking data
- Valuing Actions by Estimating Probabilities (VAEP)[3]
  - Considers all on-the-ball actions and their effects on the game
  - Not suitable for measuring off-the-ball movements

[1] Patrick Lucey, Alina Bialkowski, Mathew Monfort, Peter Carr, and Iain Matthews. 2015. quality vs quantity: Improved shot prediction in soccer using strategic features from spatiotemporal data. (2015).
[2] Javier Fernández, Luke Bornn, and Dan Cervone. 2019. Decomposing the immeasurable sport: A deep learning expected possession value framework for soccer. In 13th MIT Sloan Sports Analytics Conference.
[3] Decroos, Tom, Lotte Bransen, Jan Van Haaren, and Jesse Davis. "Actions speak louder than goals: Valuing player actions in soccer." In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 1851-1861. 2019.

Related work

- xThreat[4]
  - A discrete Markov model
  - Divides the football pitch into different zones and measures the changes in the expected scoring within these positions
  - Considers only two actions: passes and dribbles
- Deep Soccer Analytics[5]
  - Utilizes a Deep Reinforcement Learning model to learn an action-value Q-function
  - Development of soccer Goal Impact Metric (GIM) from the learned Q-function
  - Uses play-by-play event data

[4] Singh, K. 2019. Introducing expected threat. https://karun.in/blog/expected-threat.html. Accessed: 2022-07-06.
[5] Guiliang Liu, Yudong Luo, Oliver Schulte, and Tarak Kharrat. 2020. Deep soccer analytics: learning an action-value function for evaluating soccer players. Data Mining and Knowledge Discovery 34, 5 (2020), 1531–1559.

Dataset & RL environment

Reinforcement learning environment

- Google Research Football (GRF)
  - A novel, multi-player, stochastic RL environment for soccer based on OpenAI Gym
  - Implementation of the standard football rules
  - Different state representations: floats (115-dimensional vector) is used which includes:

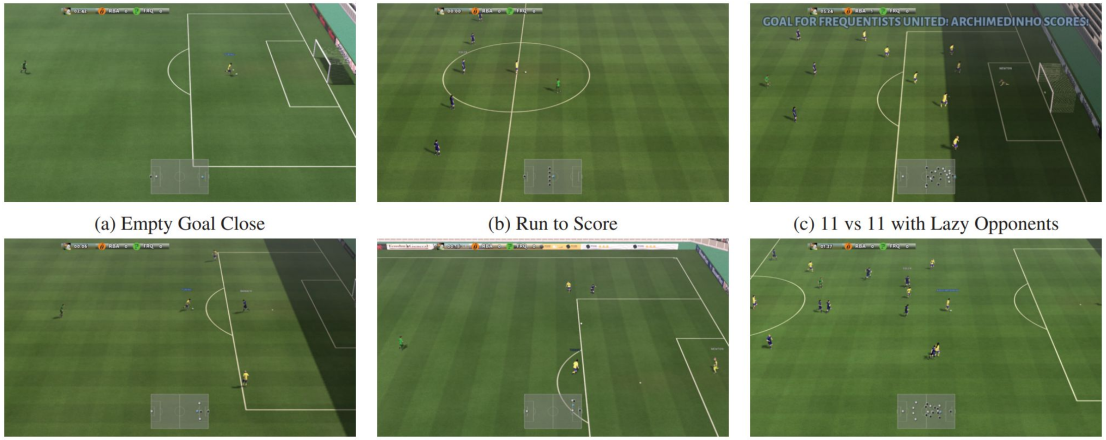
22 - (x,y) coordinates of left team players
22 - (x,y) direction of left team players
22 - (x,y) coordinates of right team players
22 - (x, y) direction of right team players
3 - (x, y and z) - ball position
3 - ball direction
3 - one hot encoding of ball ownership
11 - one hot encoding of which player is active
7 - one hot encoding of game_mode

one hot encoding of game_mode:
0 = Normal
1 = KickOff
2 = GoalKick
3 = FreeKick
4 = Corner
5 = ThrowIn
6 = Penalty

one hot encoding of ball ownership:{-1, 0, 1}, -1 = ball not owned, 0 = left team, 1 = right team.

Reinforcement learning environment

- 19 actions including shot, pass, sprint, tackle and different directional actions
- Two types of rewards:
  - Scoring: sparse goal scoring reward of +1 (when scored) and -1 (when conceded)
  - Checkpoint: agent awarded by reaching certain zones near the opposition half and shooting
- Scenarios:
  - Football benchmark: 11 vs 11 full Football game of 90 minutes (varying levels of difficulties)
    - ➢ a full game consists of 3000 frames
  - Football academy scenarios for testing RL algorithms, works as a unit test



(a) Empty Goal Close (b) Run to Score (c) 11 vs 11 with Lazy Opponents
(d) 3 vs 1 with Keeper (e) Pass and Shoot (f) Easy Counter-attack

Different football academy scenarios

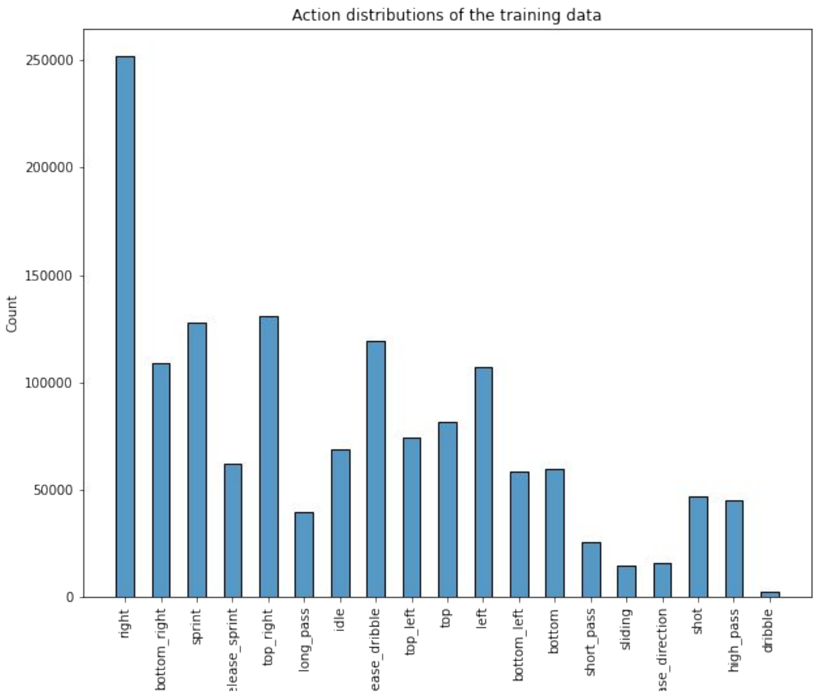
Data preparation and preprocessing

- Used 589 game replay files from the 4800 available game replays in Kaggle
  - collected from different participants' games against AI agent
- Total data points: 1,391,225 (60-20-20 split)
  - (Train, Val, Test) = (834,735, 278,245, 278,245)
- Data points converted to floats 115 representation in the format of *(observation, action)*

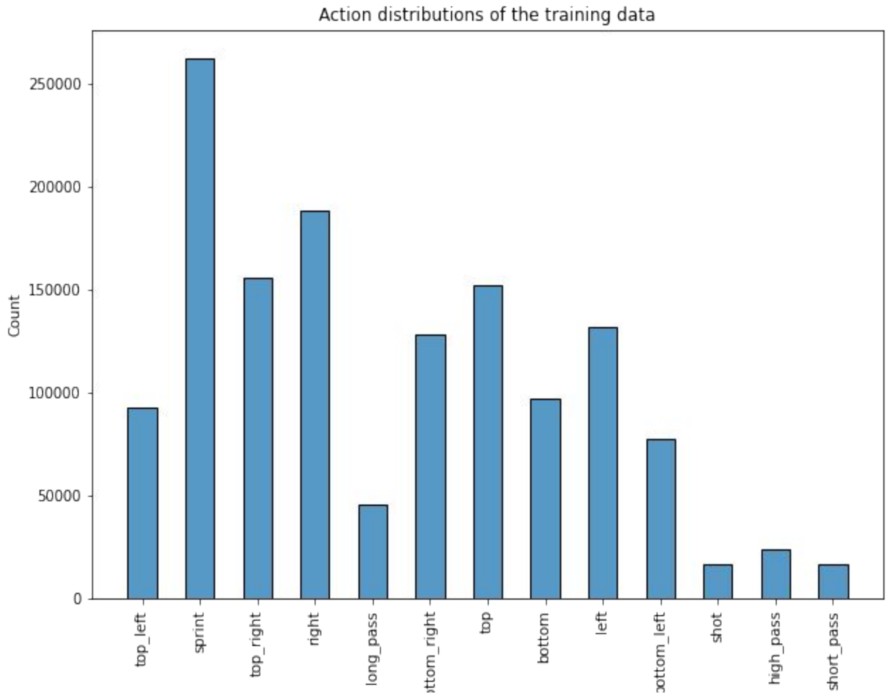
0	**1**	**2**	**…**	**…**	**…**	**114**	**115**
0.23145	0.4576	0.123	…	…	…	0.0054	1

- Ignored the following actions from a set of 19 actions:
  - idle, release_direction, release_sprint, dribble and release_dribble
  - Sticky actions such as dribble, release_direction affects the training process

Data preparation and preprocessing



Action distribution with all the actions



Action distribution after removing six actions

Methodology

Why DRL and action-value Q function?

Why DRL?

- Using a simulated football environment
- Model-free RL (no pre built model of the environment)
- An end-to-end approach
- Generalize to large dataset

Why Action-value Q function?

- Action-value Q-function *Q(s, a)*
  - is defined for states and actions
  - estimates how good it is to take an action ***a*** in a state ***s*** under a certain policy
  - serves our purpose of evaluating actions taken in different game states

Action-value Q-function

- Action-value Q-function *Q(s, a)* for a time step *t* is defined as,

$$Q(s,a) = \boxed{\mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s = s, a = a\right]} \longrightarrow \text{Q-value}$$

  - expected return of starting from sate *s* at time *t*, taking action *a* and then following a policy afterwards.

- Optimal Q-function *Q*(s, a)*: the optimal policy has an optimal *Q(s, a)* function which is defined as,

$$Q^*(s,a) = \max_{\pi} Q_{\pi}(s,a)$$

  - gives the maximum possible Q-value for a state-action pair for all possible policies π
  - satisfies the following Bellman optimality equation

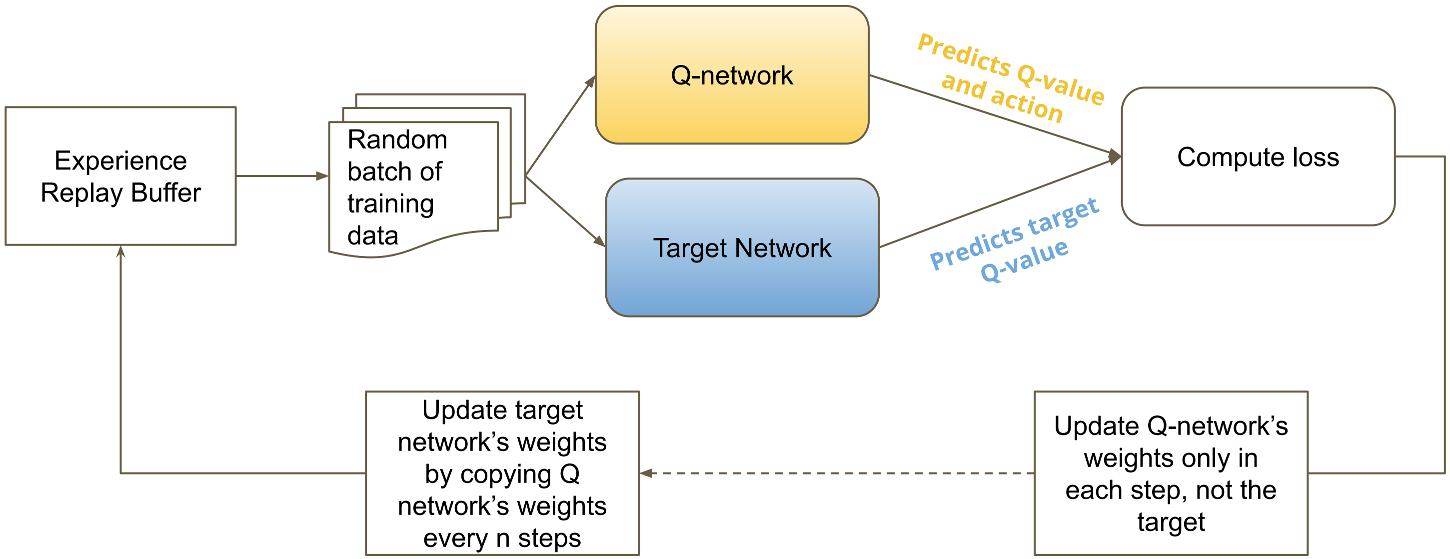
$$Q^*(s,a) = \mathbb{E}\left[\underbrace{\boxed{r}}_{\text{expected reward of taking action } a \text{ in state } s} + \underbrace{\boxed{\gamma \max_{a'} Q^*(s', a')}}_{\text{maximum expected discounted return achieved from the next state action pair } (s', a')}\right]$$

Q-learning & Deep Q-Network

- Goal:
  - To find an optimal policy by learning the optimal $Q^*(s,a)$ for each state-action pair
  - Iteratively update the Q-values using dynamic programming so that $Q(s,a) \rightarrow Q^*(s, a)$
  - Needs to store the Q-values of a state-action pair in a table
  - Feasible for small set of actions

- Deep Q-Network (DQN): off policy deep learning approach to approximate the optimal action-value function $Q^*(s', a')$
  - Q-learning stores the $Q(s,a)$ values in a table which is infeasible for an environment with lots of actions
  - Uses deep learning to approximate the optimal $Q^*(s, a)$ function

Deep Q-Network



Overview of the working flow of DQN

Experience Replay buffer

- A memory buffer where transitions from the environment are saved
- Separates the learning process from gaining experience
- Helps make data i.i.d

Target Network

- breaks the correlation of the successive observations
- a copy of the Q-network
- weights are updated from time to time to keep its predictions reliable
- stabilizes the training process

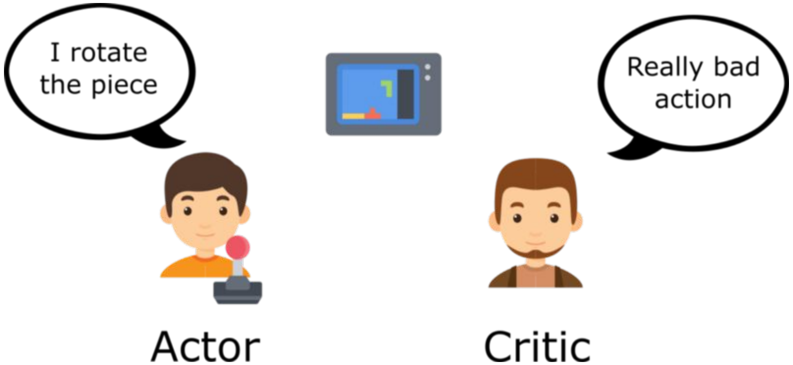
Double DQN

- Issues with DQN:
  - Overestimation of the Q-values: predicted Q values are greater than the target Q values
  - Propagation of estimation errors and overestimation of future rewards
  - Occurs due to the same Q network predicting action and estimating its Q-value

- Double DQN (DDQN)
  - Decouples the action selection from the value estimation
  - Q-network performs the action selection
  - Target network estimates the value for the action
  - Counters the overestimation of values problem of DQN

Proximal Policy Optimization

- Proximal Policy Optimization (PPO): an on-policy algorithm of the actor-critic family
  - Actor
    - ➢ Responsible for choosing the actions according to a policy and updating it
    - ➢ Takes the observations and creates a probability distribution of actions
    - ➢ An action is sampled from the probability distribution
  - Critic
    - ➢ Corresponds to the action value function Q(s,a) or the state value function V(s)
    - ➢ Produces a single number as the value of a given observation



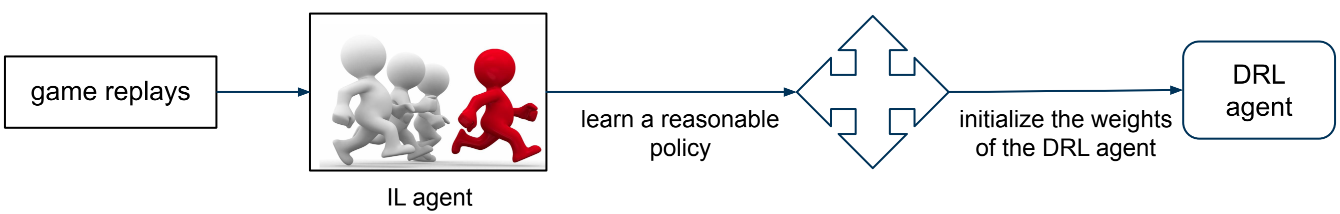
Imitation learning

- Behavioral Cloning: supervised learning technique where the agent learns from an expert

  1: Collect demonstrations (τ^* trajectories) from the expert.
  2: Treat the demonstrations as i.i.d. state-action pairs $(s_0^*, a_0^*), (s_1^*, a_1^*), \cdots$
  3: Learn policy π_θ using supervised learning by minimizing the loss function $L(a^*, \pi_\theta(s))$.

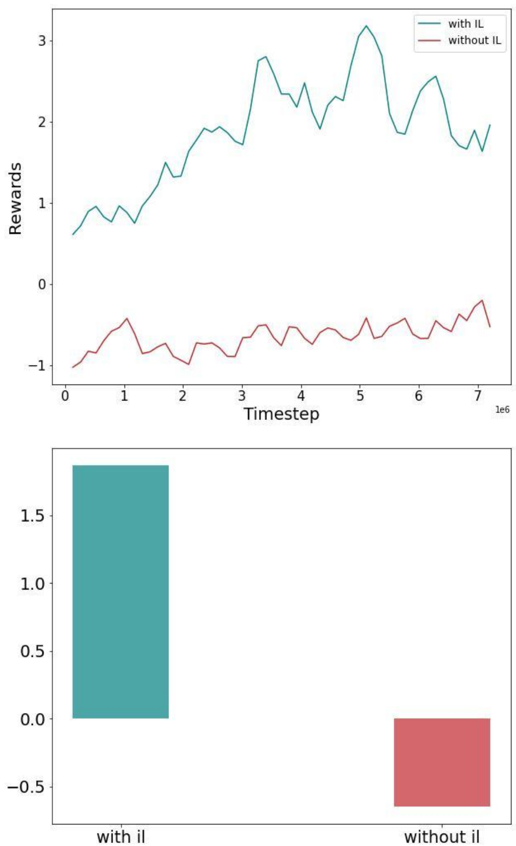
  - A 3-layered MLP network was used with dropout

- Goal:
  - Learn a reasonable performing policy
  - Initialize the weights of the DRL methods for faster convergence

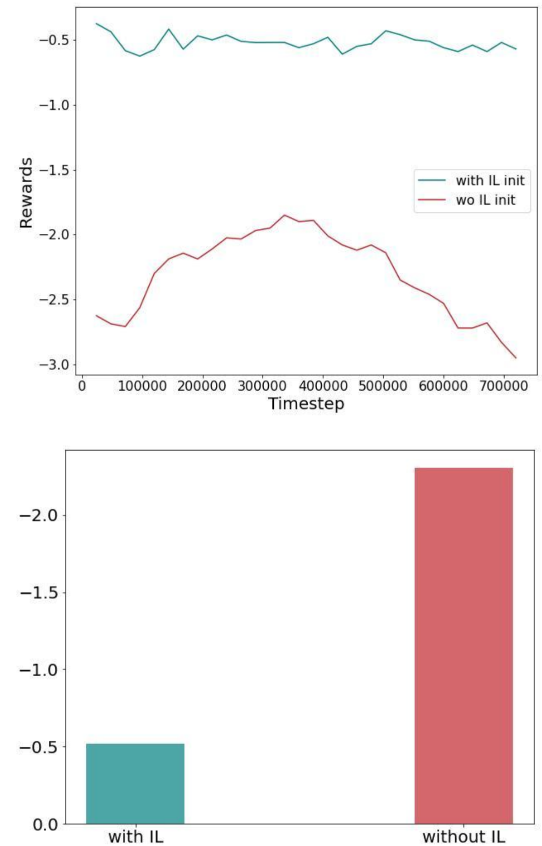


Experiments & Results

Effect of IL weight initialization on DRL methods



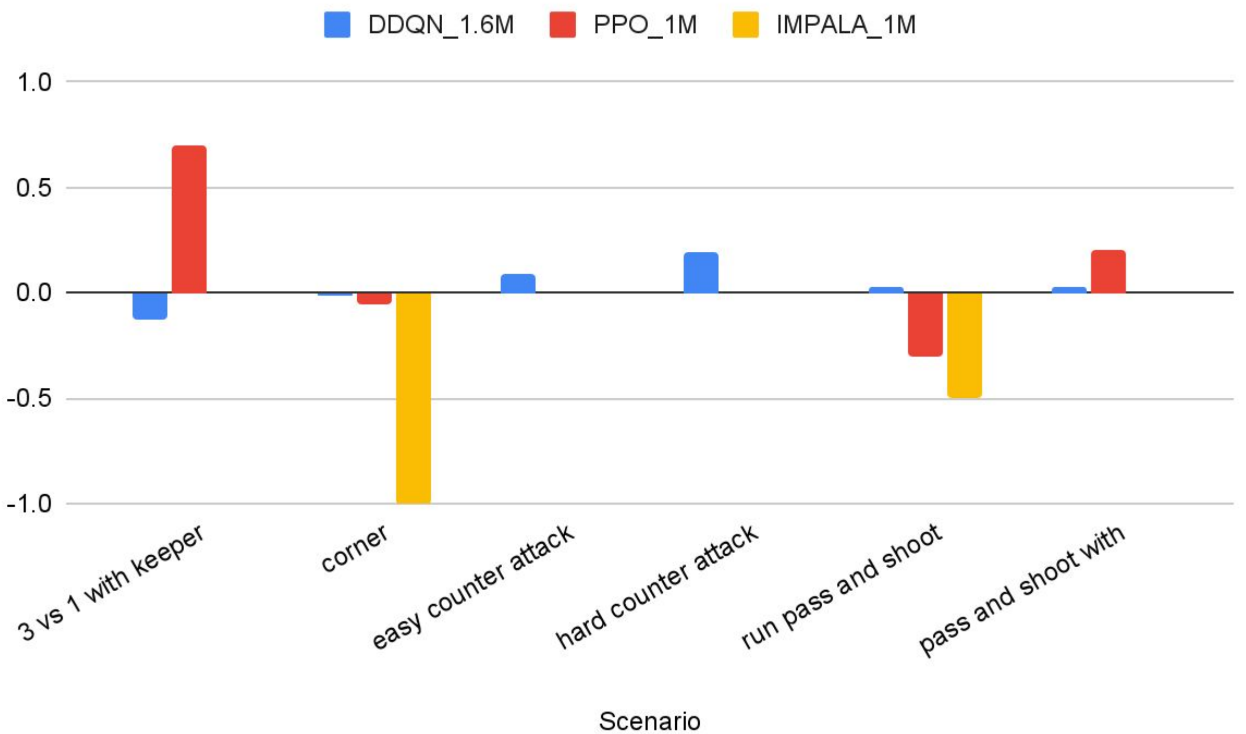
PPO



DDQN

IL weight initialization is useful for both methods, produced higher rewards than non IL initialized models.

Effect of IL weight initialization on DRL methods



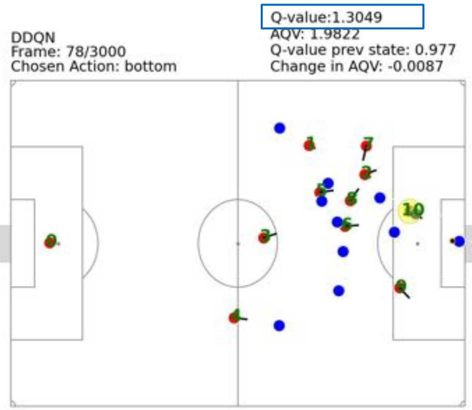
IMPALA: distributed deep RL method

Comparison among DDQN (with IL init and 1.6M timesteps), PPO and IMPALA with 1M timesteps on different football academy scenarios (rewards are scaled by a factor by 10 for ease of visualization). DDQN performs better in difficult scenarios like corner, hard counter attack, easy counter attack

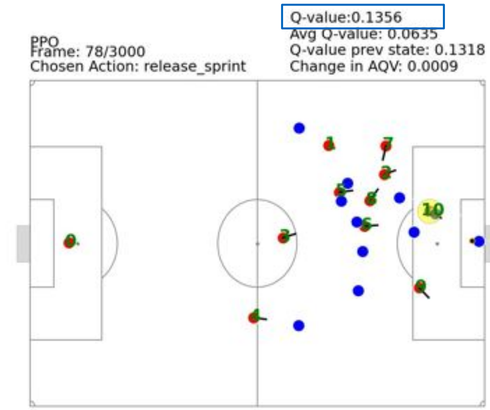
Disclaimer: the values of PPO and IMPALA have been approximated from the figure of google football research paper.

Comparison between PPO and DDQN

States of a game played by the DDQN agent (won by 3-1) are chosen to compare how the PPO agent performs on these same states.

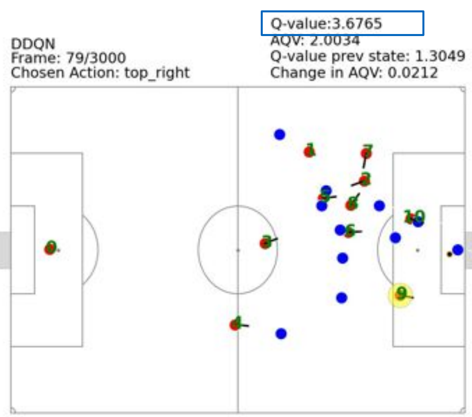


player 10 moves towards bottom direction

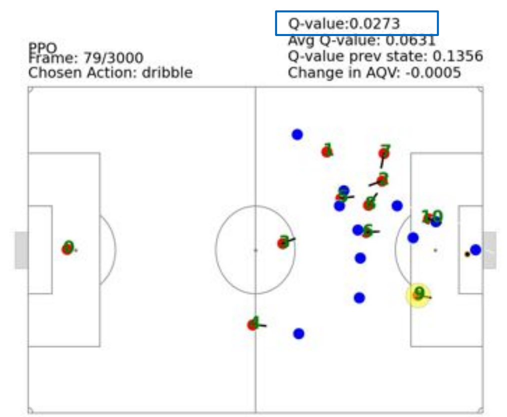


player 10 releases sprint

DDQN agent's actions are the most reasonable compared to PPO agent's actions.

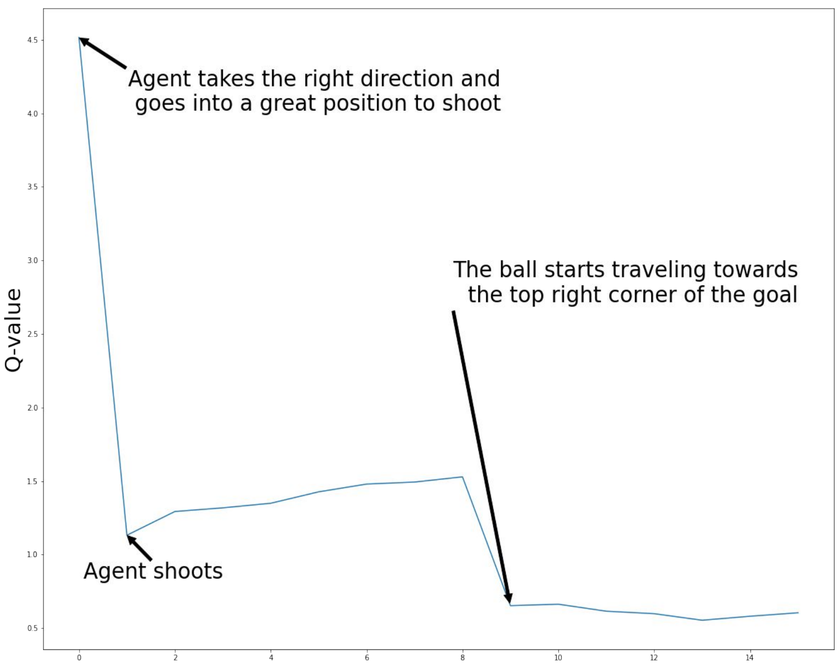


control is transferred to player 9 who moves into top_right direction

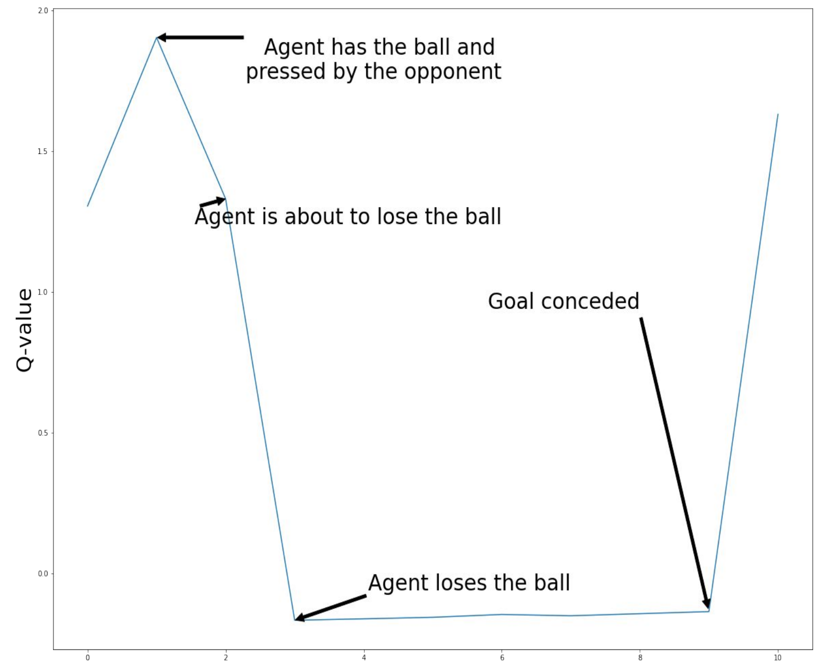


player 9 dribbles

Change of Q-values in different scenarios



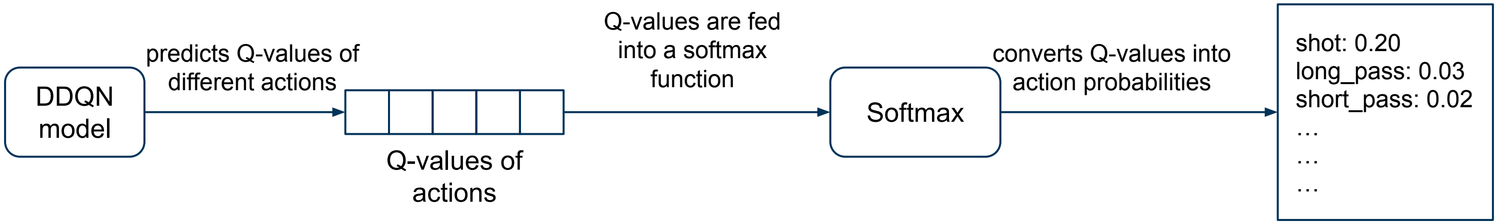
During scoring a goal

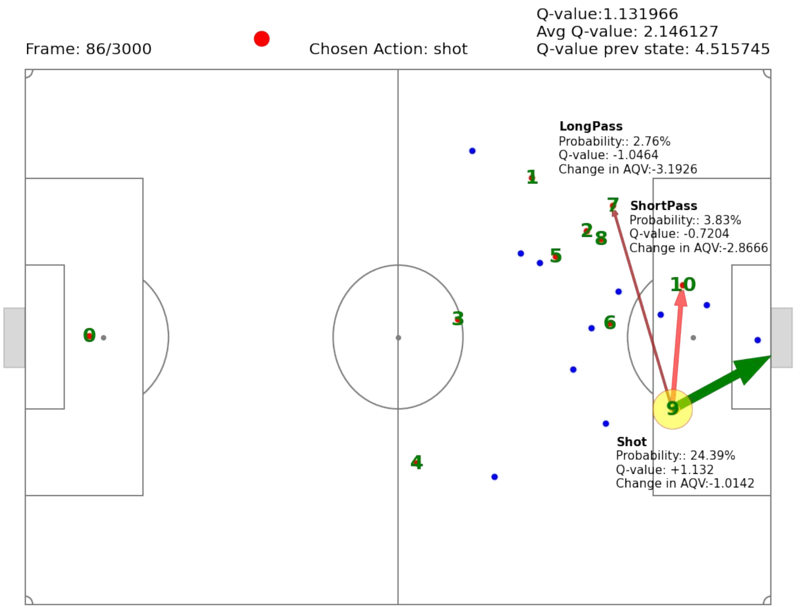


During conceding a goal

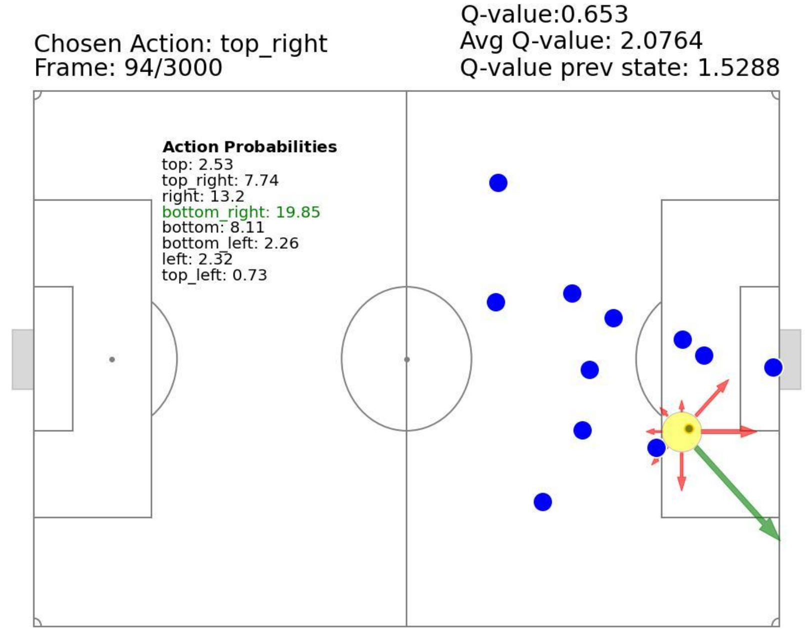
Different transitions in the game are reflected in the increase or decrease of the Q-values.

Visualization of action probabilities from a state





Probabilities of different actions

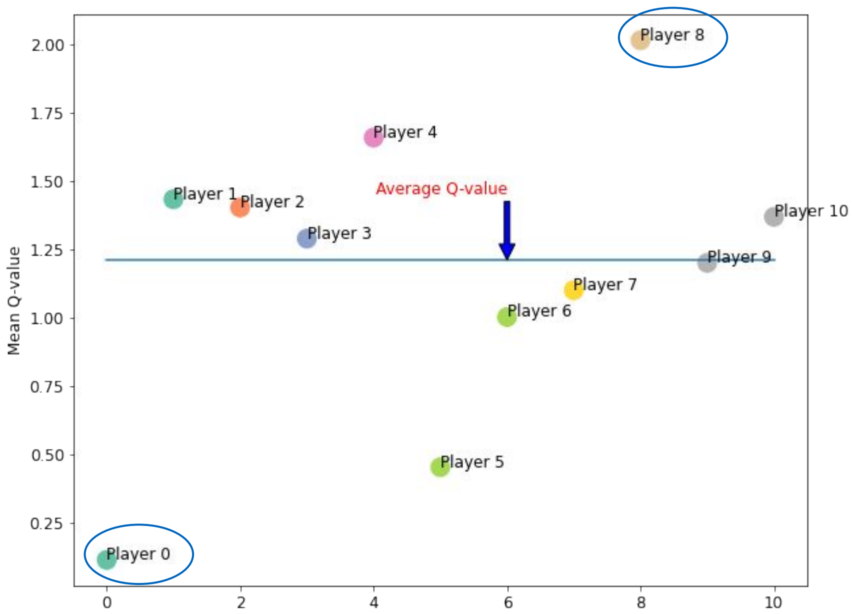


Probabilities of different directional actions

The game is played from the left to right and the directions are seen from a bird’s eye view

Rating player performances with Q-values

Player	Aggregated Q-Values	# Active	Mean Q-Value
0	15.283028	131	0.11666
1	586.607882	409	1.4342
2	270.985868	193	1.4041
3	370.35831	287	1.2904
4	413.215005	249	1.659498
5	123.836441	272	0.455281
6	430.700648	429	1.003964
7	177.414456	161	1.101953
8	352.634005	175	2.015051
9	429.074016	357	1.201888
10	461.734931	337	1.370133



- According to the aggregated Q-values: player 1 is the best performer
- Players who were active the most no. of times will have higher aggregated Q-values
- Mean Q-value is useful to reflect the quality of actions. Player 8 had 175 touches (low compared to others) but had the highest Mean Q-value. The player performed quality actions

Conclusion & Future work

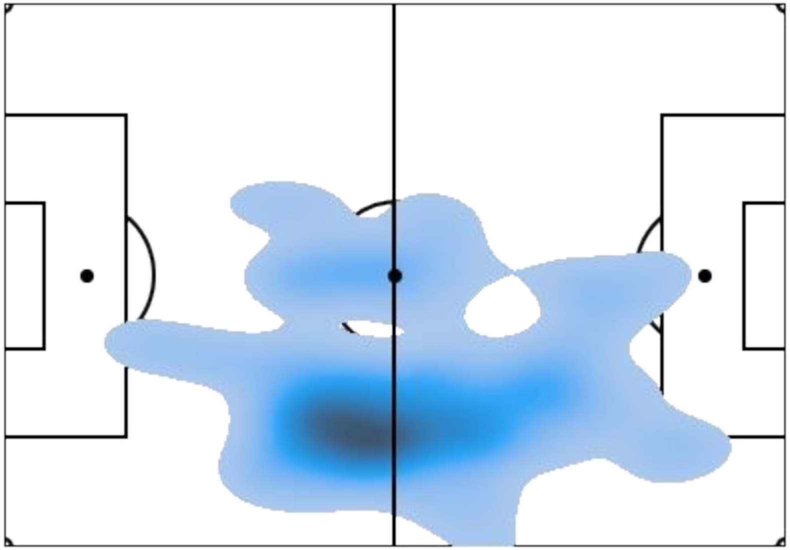
Contributions

- A robust DRL based method to design an action-value Q-function
- Successfully Quantified game states and player actions with the Q-values
- Explained and rated players’ performances
- Proposed a simulation based alternative to tackle the limitation of the availability of real world Soccer data
- Successfully combined IL with DRL to produce promising results with limited computing resources
- Compares two DRL methods (DDQN and PPO) on their ability to explain game states

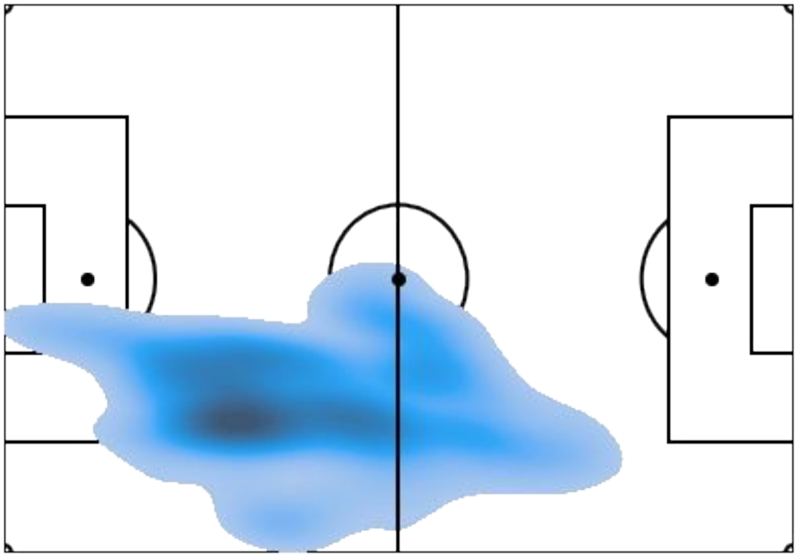
More exciting explorable research directions

- Exploring the possibilities of Multi-agent RL (MARL) in a collaborative game like Soccer
- Testing the effectiveness of the models with real-world data
  - Testing offline DDQN in its full glory
- Quantifying off-the-ball player movements
- Tweaking the DRL methods and try other methods to beat the hardest agent in GRF and also in Kaggle
- Exploring the options of transformers and graph neural networks

Evaluating state action spaces - Heatmaps of players with high and low average Q-value



Heatmap of player 8



Heatmap of player 1

Motivation

- Learning a value function using deep reinforcement learning
- Quantifying game states and player actions using the values obtained from the value function
- Explaining different game scenarios and player actions (on-the-ball and off-the-ball)
- Rating player performances

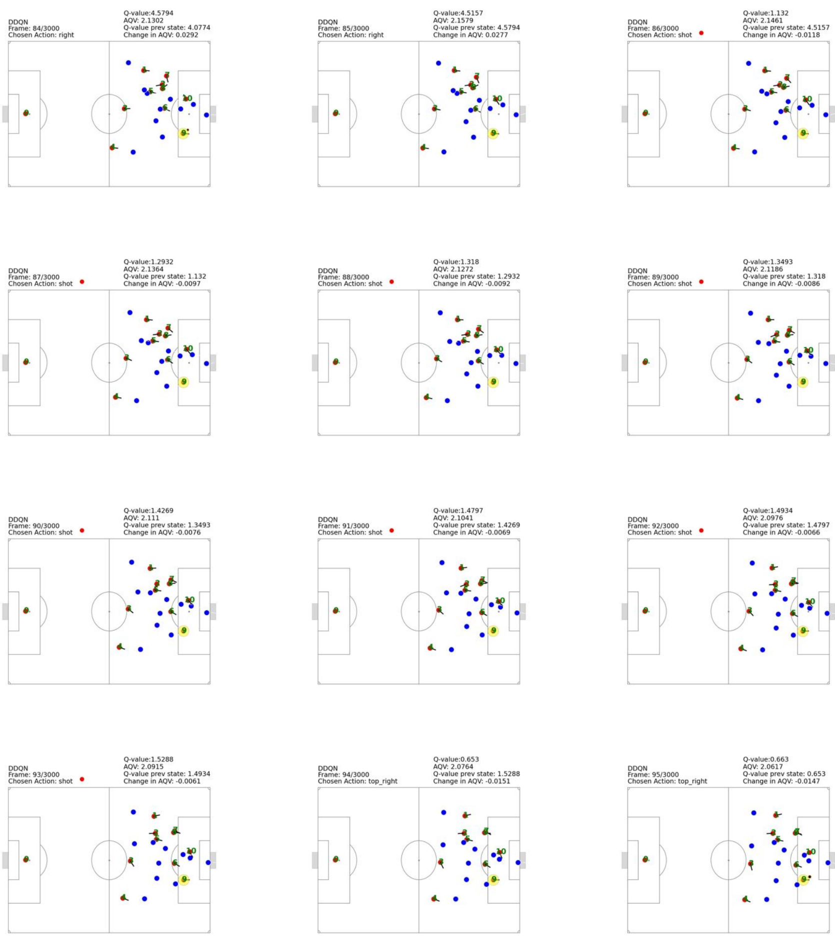
Methodology - Deep Reinforcement Learning (DRL)

- Deep Q-Network (DQN): off policy deep learning approach to approximate the optimal action-value function Q*(s', a')

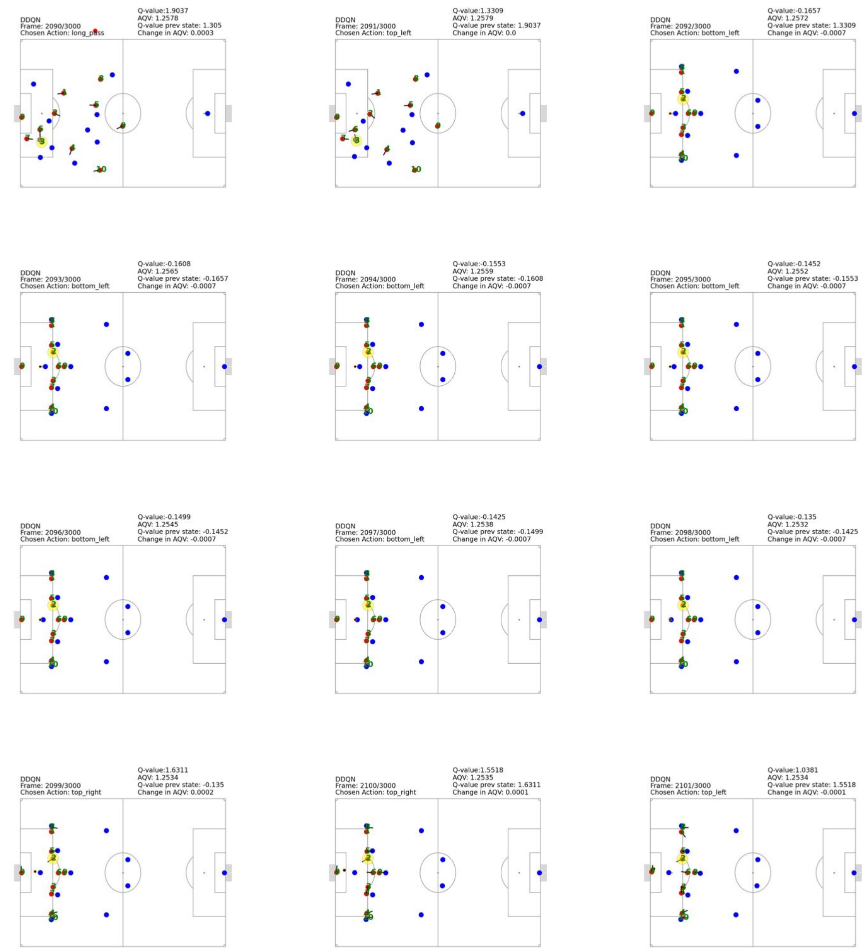
$$Q^*(s,a) = \mathbb{E}[r + \gamma max_{a'} \ Q^*(s',a')]$$

  - Target network:
    - A second network beside the original Q-network
    - Weights from the Q-network are copied from time to time to the target network
    - Used for countering the problem of correlated samples
    - Stabilizes the training process

Evaluating state action spaces with Q-values - change of Q-values



During scoring a goal



During conceding a goal

Deep Q-Network

- Deep Q-Network (DQN): off policy deep learning approach to approximate the optimal action-value function $Q^*(s', a')$
    - Q-learning stores the $Q(s,a)$ values in a table which is infeasible for an environment with lots of actions
    - Uses deep learning to approximate the optimal $Q^*(s, a)$ function
- Target network: A second network beside the original Q-network
    - Weights from the Q-network are copied from time to time to the target network
    - Used for countering the problem of correlated samples and stabilizing the training
- Experience Replay buffer: a buffer where transitions from the environment are stored
    - Breaks the correlation between successive samples
    - Allows to learn from individual experiences multiple times

Reinforcement learning environment

- Google Research Football (GRF)
  - A novel, multi-player, stochastic RL environment for soccer based on OpenAI Gym
  - Implementation of the standard football rules
  - Different state representations: floats (115-dimensional vector) is used which includes:
    - ➢ Coordinates and directions of left and right team players
    - ➢ Ball position and direction, ball ownership, active player and game mode
- 19 actions including shot, pass, sprint, tackle and different directional actions
- Two types of rewards:
  - Scoring: sparse goal scoring reward of +1 (when scored) and -1 (when conceded)
  - Checkpoint: agent awarded by reaching certain zones near the opposition half and shooting
- AI opponent of varying levels of difficulties
- Different football academy scenarios for testing RL algorithms, works as a unit test
  - simple scenarios such as corner, 3 vs 1 with keeper, run to score against the GK, etc.