Data-Driven Action-Value Functions for Evaluating Players in Professional Team Sports

by

Guiliang Liu

B.Sc., South China University of Technology, 2016

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

Declaration of Committee

Name:	**Guiliang Liu**
Degree:	**Doctor of Philosophy**
Thesis title:	**Data-Driven Action-Value Functions for Evaluating Players in Professional Team Sports**

Abstract

As more and larger event stream datasets for professional sports become available, there is growing interest in modeling the complex play dynamics to evaluate player performance. Among these models, a common player evaluation method is assigning *values to player actions.* Traditional action-values metrics, however, consider very limited game context and player information. Furthermore, they provide directly related to goals (e.g. shots), not *all actions*. Recent work has shown that reinforcement learning provided powerful methods for addressing quantifying the value of player actions in sports. This dissertation develops deep reinforcement learning (DRL) methods for estimating action values in sports. We make several contributions to DRL for sports.

First, we develop neural network architectures that learn an action-value Q-function from sports events logs to estimate each team's expected success given the current match context. Specifically, our architecture models the game history with a recurrent network and predicts the probability that a team scores the next goal. From the learned Q-values, we derive a Goal Impact Metric (GIM) for evaluating a player's performance over a game season. We show that the resulting player rankings are consistent with standard player metrics and temporally consistent within and across seasons.

Second, we address the interpretability of the learned Q-values. While neural networks provided accurate estimates, the black-box structure prohibits understanding the influence of different game features on the action values. To interpret the Q-function and understand the influence of game features on action values, we design an interpretable mimic learning framework for the DRL. The framework is based on a Linear Model U-Tree (LMUT) as a transparent mimic model, which facilitates extracting the function rules and computing the feature importance for action values.

Third, we incorporate information about specific players into the action values, by introducing a deep *player representation framework*. In this framework, each player is assigned a latent feature vector called an embedding, with the property that statistically similar players are mapped to nearby embeddings. To compute embeddings that summarize the statistical information about players, we implement a Variational Recurrent Ladder Agent Encoder (VaRLAE) to learn a contextualized representation for when and how players are likely to act.

We learn and evaluate deep Q-functions from event data for both ice hockey and soccer. These are challenging continuous-flow games where game context and medium-term consequences are crucial for properly assessing the impact of a player's actions.

Keywords: Action-Value Q-function; Deep Reinforcement Learning; Mimic Learning; Regression Tree; Deep Representation Learning; Variational Auto-Encoder.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Oliver Schulte, for his consistent support and and invaluable guidance throughout my Ph.D. study. All of his patience, motivation, enthusiasm, and immense knowledge encouraged me to finish my research. His insightful vision and profound knowledge led me to become an independent researcher. The great role he sets for me will always inspire me to pursue the real knowledge in the rest of life.

I am very grateful to all the other committee members, including Prof. Tim Swartz as the supervisor, Prof. Kangkang Yin as the internal examiner, Prof. Jesse Davids as the external examiner, and Prof. Nick Summer as the chair. They provided constructive comments and helpful feedback to my Ph.D. thesis, which effectively encourages my future study and research.

My sincere thanks also goes to all of those with whom I have had the pleasure to work during this project and other related work, especially Fangxin Wang, Yudong Luo, Xiangyu Sun and Mohan Zhang. Each of them has provided me additional personal and professional guidance and I learned a lot about both scientific research and life in general from them.

Last but not least, nobody has been more important to me in the pursuit of this degree than my family members, in particular Grace Bi. I would like to thank my parents, whose love and guidance are always with me in whatever I pursue.

Table of Contents

Declaration of Committee **ii**

Abstract **iii**

Acknowledgements **v**

Table of Contents **vi**

List of Tables **x**

List of Figures **xiii**

1 Introduction **1**
- 1.1 Overview 1
- 1.2 Summary of Contributions 2
- 1.3 Thesis Organization 4

2 Background and Related Work **5**
- 2.1 A Review of Action-Value Metrics 5
  - 2.1.1 Single Action-Value Counts 5
  - 2.1.2 Multiple Actions-Value Counts 7
  - 2.1.3 Value-Above-Replacement 9
- 2.2 A Review of Sports Data 9
  - 2.2.1 Box Scores 9
  - 2.2.2 Play-by-Play Dataset 10
  - 2.2.3 Game Tracking Dataset 11
  - 2.2.4 Video Dataset 12

3 Preliminaries **13**
- 3.1 Markov Model for Sports Games 13
  - 3.1.1 Partially Observable Markov Decision Processes 13
  - 3.1.2 Markov Game Model 14
  - 3.1.3 Play Dynamics in Sports Games 14

3.2 Domain Knowledge: an Introduction to the Game Rules 15
3.2.1 Ice Hockey 15
3.2.2 Soccer 16
3.3 The Play-by-Play Sports Dataset 17
3.3.1 The Ice Hockey Datatset 17
3.3.2 The Soccer Dataset 18

4 Learning an Action-Value function for Evaluating Ice Hockey players with Deep Reinforcement Learning 20
4.1 Introduction: Valuing Actions of Ice Hockey Players 20
4.2 Task Formulation and Approach 21
4.3 Play Dynamics in NHL 22
4.4 Learning Q values with DP-LSTM Sarsa 23
4.4.1 Network Architecture 23
4.4.2 Illustration of Temporal Projection 24
4.4.3 Illustration of Spatial Projection 24
4.5 Player Evaluation Metric Based on Q values 25
4.5.1 Goal Impact: Deriving Action Values from Q-values. 26
4.5.2 Q Value Above Average Replacement 27
4.5.3 Rank Players with GIM 28
4.6 Empirical Evaluation 29
4.6.1 Comparison Player Evaluation Metrics 30
4.6.2 Season Totals: Correlations with standard Success Measures 30
4.6.3 Round-by-Round Correlations: Predicting Future Performance From Past Performance 31
4.6.4 Future Seasons: Predicting Players' Salary 32
4.6.5 Cross-Season Correlations: Measuring the consistency of GIM between seasons. 33
4.7 Summary 35

5 Extending the Action-Value Function for Evaluating Soccer Players 36
5.1 Introduction: Valuing Actions of Soccer Players 36
5.2 Learning Q Values: Model Architecture and Training 37
5.2.1 Model Architecture: Function Approximation with Neural Network 37
5.2.2 Weight Training 37
5.3 Model Validation: Q Values 39
5.3.1 Illustration of Temporal and Spatial Projection. 39
5.3.2 Calibration Quality for the learned Q-function 42
5.4 Player Ranking: Case Study 44
5.4.1 All-Actions Assessment 44

5.4.2 Action-Specific Assessment 45
5.5 Player Ranking: Empirical Evaluation 46
5.5.1 Comparison Player Evaluation Metrics 46
5.5.2 Season Totals: Correlations with Standard Success Measures 47
5.5.3 Round-by-Round Correlations: Predicting Future Performance From Past Performance 48
5.6 Summary 49

6 Understanding the Action Values with Interpretable Mimic Learning 51
6.1 Introduction: Mimic a Deep Reinforcement Learner 51
6.2 Background and Related Work 52
6.2.1 Mimic Learning 52
6.2.2 U-Tree Learning 53
6.3 Mimic Learning for Sports Environment 53
6.3.1 Mimic Learning with a Regression Tree 53
6.3.2 Mimic Learning with a Linear Model U-Tree 59
6.4 Mimic Learning for Active Deep Reinforcement Learning 67
6.4.1 Mimic Learning Framework for DRL 67
6.4.2 Empirical Evaluation 69
6.5 Summary 74

7 Embedding Player Information with Representation Learning 76
7.1 Introduction 76
7.2 Related Work 77
7.3 Player Representation Framework 78
7.3.1 Contextual Variables for Ice Hockey Players 78
7.3.2 Player Representation via Player Generation 79
7.4 A Variational Agent Encoder for Learning Agent Representations 80
7.5 Empirical Evaluation 82
7.5.1 Experiment Settings 82
7.5.2 Generative Performance of Embedding Models: On-Puck Player Identification 83
7.5.3 Predictive Performance On Application Tasks 84
7.5.4 Posterior Collapse 88
7.6 Summary 88

8 Conclusion, Discussion, and Future Directions 89
8.1 Summary of the Dissertation 89
8.2 Discussion 90
8.2.1 The Sparsity of Goals 90
8.2.2 Model Convergence 91

8.2.3 Learning Contextualized Representations. 91
8.2.4 Shrinkage Effect 91
8.2.5 Limitations 91
8.3 Future Directions 92
8.3.1 Model Game History with Attention Mechanism 92
8.3.2 Information Bottleneck Method for Mimic Learning 93
8.3.3 Multi-Agent Embedding for Player Representations 93

Bibliography **94**

A Appendix **102**
A.1 Proof of Proposition 1 102
A.2 VAEP Implementation 103
A.3 Ranking for the Premier League 103
A.4 Actions Details 105
A.5 Generating the Spatial Projections 105
A.6 A Spatial Illustration for the Shot Attempts 105

List of Tables

Table 2.1 A summary of the previous +/- based approaches. 6

Table 2.2 A summary of the previous XG based approaches. 7

Table 2.3 A summary of the multiple actions-value counts approaches. 7

Table 2.4 An example of box scores for NBA players. It summarizes the player performance of a game and reports the basketball statistics including playing Minutes (Min), points (PTS), field goals made/attempted/percentage (FGM/FGA/FG%) and three-point field goals made/attempted/percentage (3PM/3PA/3P%). . . 10

Table 2.5 An example of box scores for NBA teams. It summarizes the team performance of a game and reports the basketball statistics including playing Minutes (Min), points (PTS), field goals made/attempted/percentage (FGM/FGA/FG%) and three-point field goals made/attempted/percentage (3PM/3PA/3P%). . . 10

Table 2.6 An example of a play-by-play data sample featuring team scoring: a sequence of events where home team scores and then away team scores. The rewards [1,0,0] and [0,1,0] indicate the scoring event of the home team and away team respectively. We skip some events in the middle due to space issues. 11

Table 3.1 Dataset statistics for the Ice Hockey dataset. The basic unit of this dataset is an **event**, which describes the game context and the on-the-ball action of a player at a time step. 18

Table 3.2 The complete feature list for an observation, where the manpower situation is marked by Event Strength (ES), Short-Handed (SH) and Power Play (PP). 18

Table 3.3 Dataset Example 18

Table 3.4 Derived Features 18

Table 3.5 Complete feature list. For the feature manpower situation, negative values indicate short-handed, positive values indicate power play. 19

Table 3.6 Dataset statistics. The basic unit of this dataset is an **event**, which describes the game context and the on-the-ball action of a player at a time step. 19

Table 4.1 2015-2016 Top-20 Player Impact Scores 29

Table 4.2 Correlation with standard success measures. 31

Table 4.3 Correlation with Players' Contract 32

Table 4.4 The pearson correlation coefficient of the players' GIMs between two continuous seasons. 35

Table 5.1 Calibration results for the spatial illustration of tackle values. 41
Table 5.2 Calibration Results. TT_Home and TT_Away report the average scoring probability $\hat{Q}^{team}(A)$ estimated by our TTDP-LSTM model. Here we compare only Q values for pass and shot as they are frequent and well-studied actions. TT_MAE is the Mean Absolute Error (MAE) between estimated scoring probabilities from our model and empirical scoring probabilities. For comparison, we also report a Markov_MAE which applies the estimates from a discrete-state Markov model [85]. 43
Table 5.3 2017-2018 season top-10 Player Impact Scores for players in EFL Championship game season. 45
Table 5.4 Top-10 soccer players with largest shot impact in 2017-2018 EFL Championship game season. 45
Table 5.5 Top-10 soccer players with largest pass impact in 2017-2018 EFL Championship game season. 45
Table 5.6 Correlation with standard success measures for all the players. We bold the highest correlations and underline the lowest ones for penalties. 47
Table 5.7 Correlation with standard success measures for players in the EFL Championship. We bold the highest correlations and underline the lowest ones for penalties. 48

Table 6.1 Performance of General Mimic Regression Tree (RT) with different Minimum Samples in each Leaf node (MSL). We apply ten-fold cross validation and report the regression result with format: Mean Square Error (Variance) . 54
Table 6.2 Top 10 features for Q values. 55
Table 6.3 Feature importance for the impact of shot. 55
Table 6.4 Feature importance for the impact of pass. 55
Table 6.5 Exceptional Players Based on Tree Discretization 59
Table 6.6 Partition Cell 60
Table 6.7 Fidelity to Deep Model: RMSE on Test Set 64
Table 6.8 Top 10 Features for the Shots for Ice Hockey Players. 65
Table 6.9 Top 10 Features for the Shots for Soccer Players. 65
Table 6.10 Result of Mountain Car 71
Table 6.11 Result of Cart Pole 71
Table 6.12 Result of Flappy Bird 71
Table 6.13 Game Playing Performance 73
Table 6.14 Feature Influence 73

Table 7.1 Results for acting players identification. We report both Accuracy and Log-Likelihood (LL). 84
Table 7.2 Expected goal results applying different player embeddings. 86
Table 7.3 The test set game prediction error between predicted and final score differences for the entire game. 87

Table A.1 2017-2018 season top-10 Player Impact Scores for players in Premier League season. 104
Table A.2 Top-10 soccer players with largest shot impact in 2017-2018 Premier League season. 104
Table A.3 Top-10 soccer players with largest pass impact in 2017-2018 Premier League season. 104

List of Figures

Figure 2.1 A tree diagram to position previous works about action-value based player evaluation metric. An important factor is whether a metric considers all actions or only a subset of them. 5

Figure 2.2 An example of video tracking: at each video frame, the tracking system dynamically locates the position of on-the-ball player (in blue frame) with the object detection algorithm and records the player action. The system might require manually labeling the position of the ball (or the puck) for the sports with a large pitch (or rink). 12

Figure 3.1 Ice Hockey Rink. The hockey rink It measures 200 feet (60.96 m) by 85 feet (25.91 m), which can be divided into three zones by two blue lines. The zone between two blue lines is referred to as a neutral zone. The zone with the team's own goal is called a defensive zone. The zone toward which a team's play flows (or attacks) is the offensive zone. 16

Figure 3.2 Soccer pitch layout with adjusted coordinates. Coordinates are adjusted so that for the home/away team performing an action, its offensive zone is on the right 17

Figure 4.1 System Flow for Player Evaluation 22

Figure 4.2 Our design is a 5-layer network with 3 hidden layers. Each hidden layer contains 1000 nodes, which utilize a relu activation function. The first hidden layer is the LSTM layer, the remaining layers are fully connected. Temporal-difference learning looks ahead to the next goal, and the LSTM memory traces back to the beginning of the play (the last possession change). 24

Figure 4.3 Temporal Projection of the method. For each team, and each game time, the graph shows the chance the that team scores the next goal, as estimated by the model. Major events lead to major changes in scoring chances, as annotated. The network also captures smaller changes associated with **every** action under different game contexts. 25

Figure 4.4 Spatial Projection for the shot action: The probability that the home team scores the next goal after taking a shot at a rink location, averaged over possible game states. Figure 4.5 shows the positions of shots behind the goal, which explains the distribution of Q values around the goal. 25
Figure 4.5 Shots behind the opponent's goal in the 2015-16 NHL game season. . . . 26
Figure 4.6 Correlations between round-by-round metrics and season totals. 33
Figure 4.7 Player GIM vs. Value of new contracts in the 2016-17 (left) and 2017-18 (right) NHL season. 34
Figure 4.8 The scatter plot of players' GIMs between 15-16 and 16-17 NHL season. . 34
Figure 4.9 The scatter plot of players' GIMs between 16-17 and 17-18 NHL season. . 34
Figure 4.10 The scatter plot of players' GIMs between 17-18 and 18-19 NHL season. . 35

Figure 5.1 The architecture of our Two-Tower Dynamic Play LSTM (TTDP-LSTM). The figure shows how the model processes two generic time instances, one where the home team acts—which is analyzed by the home tower—and another where the away team acts—analyzed by the away tower. 38
Figure 5.2 Temporal Projection of the learned Q-function. The game is between Fulham (Home) and Sheffield Wednesday (Away), which has happened on Aug.19^{th}, 2017. 39
Figure 5.3 Spatial Projections for estimated Q values: $\hat{Q}^{Home}(s, shot)$, $\hat{Q}^{Home}(s, pass)$, $\hat{Q}^{Home}(s, cross)$ and $\hat{Q}^{Home}(s, tackle)$ over the entire soccer pitch. We use the adjusted coordinate described in Section 3.3.1. 40
Figure 5.4 Spatial illustration of all goal-scoring shots under the adjusted coordinates (with our dataset). 41
Figure 5.5 Spatial illustration of tackle under the adjusted coordinates (with our dataset) in 100 randomly sampled game. We label these tackles by whether the team that performs the tackle manages to score the next goal. 42
Figure 5.6 Correlations between round-by-round metrics and season totals for all players. 49
Figure 5.7 Correlations between round-by-round metrics and season totals for the players in EFL Champion. 49

Figure 6.1 Interpretable Mimic Learning Framework 54
Figure 6.2 Regression tree for the impact of shot. 56
Figure 6.3 Regression tree for the impact of pass. 57
Figure 6.4 The impact values for passes v.s. Game Time **Remain**. We randomly sample 10 games and plot the impact values for all passes. 57
Figure 6.5 Partial Dependence Plot for Time Remaining (left), X Coordinate (middle) and X Velocity (right) 58
Figure 6.6 An example of Linear Model U-Tree (LMUT). 60

Figure 6.7 Model Tree Example With 4 Layers for Action Values of Shots in Soccer . 66
Figure 6.8 Rule Example 1 for Action Values of Shots in Ice Hockey. The model tree for ice hockey produces a prediction for the Q-probability that the home team scores the next goal after a shot. 66
Figure 6.9 Rule Example 2 for Action Values of Shots in Ice Hockey 66
Figure 6.10 Rule Example for Action Values of Shots in Soccer 67
Figure 6.11 Experience Training Setting 68
Figure 6.12 Active Play Setting. 69
Figure 6.13 Coverage v.s. Optimality 71
Figure 6.14 Consecutive Testing of LMUT 72
Figure 6.15 Super pixels in Flappy Bird 73
Figure 6.16 Examples of Rule Extraction for Mountain Car and Cart Pole. 74
Figure 6.17 Flappy Bird input images with Super-pixels (marked with red stars). The input order of four consecutive images is left to right. 75

Figure 7.1 Our VaRLAE model includes a conditional Ladder-VAE at every RNN cell. VaRLAE applies a top-down dependency structure ordered as the sports causal relationship (Section 7.3.1). The thick/dash lines denote logical-functions /stochastic-dependence. The shaded nodes are given. 80
Figure 7.2 Embedding visualization. Each data point corresponds to a player embedding conditioning on the game context at the current time step t. The player embeddings are labelled 1) player positions (on the *left* column, including Center (C), Defense (D), Left-Wing (LW) and Right-Wing (RW) and Goalie (G)) 2) 5 selected defence men (on the *middle* column) and 3) player locations (on the *right* column, including Defence Zone (DZ), Neutral Zone (NZ) and Offensive Zone (OZ)). The embeddings are computed by VaRLAE (*top plots*) and CAERNN (*bottom plots*) respectively. 85
Figure 7.3 Embedding visualization. Each data point corresponds to a player embedding conditioning on the game context at the current time step t. The player embeddings are labelled by the action types. The embeddings are computed by VaRLAE (*top plots*) and CAERNN (*bottom plots*). 85
Figure 7.4 Temporal illustrations of the absolute error between predicted score differences and final score differences. The plots report mean (left) and mean$\pm$variance of the differences (right). Figure 7.5 shows the separated plots for each method. 87

Figure 7.5 Temporal illustrations of the absolute error between predicted score differences and final score differences. We report mean$\pm$variance of the error at each time step for all compared methods. 87

Figure 7.6 The KLD between the posteriors and the priors during training for VaRLAE , CVRNN and CVAE (from left to right). 88

Figure A.1 The spatial illustration of shot attempts on a hockey rink. We apply the adjusted coordinate and the play always flows from left to right. Blue circles represent unsuccessful shots and red stars indicate successful shot. 106

Chapter 1

Introduction

1.1 Overview

The past decades have seen booming interest in evaluating the player performance in professional team sports. Recently, with the advancement of high-frequency optical tracking systems and object detection systems, more and larger event stream datasets have become available for a variety of professional team sports. These datasets elaborate both spatial and temporal features (e.g., player locations and game time) of game events at dense time intervals. The rich yet complicated data sources bring numerous opportunities for large scale machine learning algorithms to model complex sports dynamics. Such models significantly facilitate computing or learning metrics for player evaluation. Among the recently proposed metrics, the most common approach has been to quantify players' influences on the scoring probabilities and winning chance with action values [85, 80, 84, 21, 63, 29, 32]. These values have strong real-world impacts, for example, they provide professional teams an empirical basis for deciding which players to trade or draft during a game season.

Previously proposed metrics for evaluating the actions of players are commonly limited in the following aspects:

Limited Game Context: When assigning values to the action of players, traditional works often consider a limited context of the acting player. For example, a previous work [80] built a Markov model for representing the ice hockey game context. The states of the Markov model were defined by only three features: period, goal differential, and manpower differential. However, in professional sports, the relevant context is very complex, including specific game time, positions of players, speed of the ball and duration of the actions, etc. The states in the previous Markov models contained only the current observations whereas the influence of action often relies on game history in the real sports match, for example, a block after a shot should be more important than a pass-blocking. Omitting the contextual features and the game history will significantly compromise the performance of the underlying sports models.

Insufficient Action Look-Ahead: Many traditional sports models assess only the actions that have an immediate impact on goals (e.g. shots), but not the actions that create these scoring opportunities (e.g. pass, reception). A typical example is the Plus-Minus (+/-) metric. +/- awards a player +1 if a goal is scored by the player's own team when the player is on the pitch, and -1 if the other team scores. Some more advanced +/-'s [86, 67, 88, 52] modeled the game context and weighted goals by their importance on game-winning, but they considered only goal scoring, not other actions that also have major impacts on scoring (e.g., assist, pass and block, etc.). Some recent works [29, 82] have proposed assessing an action by whether it can lead to a goal within a fixed window size (e.g., 20 seconds). Their systems can assign values to the actions before a scoring event (e.g., pass, assist, etc.,), but they have limited ability to evaluating many defensive actions that have a medium-term effect on goal scoring.

Opaque Model Structure: Many recent works have applied a black-box model (e.g., neural model and random forest) to represent the play dynamics in team sports. For example, a recent work [32] trained a deep neural network to compute the Expected Possession Values (EPVs) for evaluating the actions of players in soccer games. The opaque neural model structure, however, prohibits understanding 1) how a game feature influences the expected action values and 2) which are the most important game features for improving the impact of an action. The difficulty of understanding the knowledge learned by the black-box sports models significantly influences the confidence of coaches and fans on these models and the corresponding action values.

Omitting the In-game Influence of Individual Players on Action Values: When estimating action values, previous sports models based on reinforcement learning often pool observations of different players without capturing the specific roles and behaviors of each athlete who is performing the action. Neglecting the special characteristics of individual players significantly compromises the model performance for many application tasks, such as predicting the game-winner and estimating the expected goals [68]. A previous work [33] proposed embedding a player's information with a continuous value vector following an Auto-Encoder structure. Their deterministic player embedding, however, only considered very limited game context which prohibits their embedding model from generalizing to a large number of player in a sports league.

1.2 Summary of Contributions

In this dissertation, we introduce our approaches to handling the above challenges. This section summarizes the contributions in the following aspects.

Learning the Q-function with Deep Reinforcement Learning (DRL): To model the complex game context, we build a Markov Game Model [61] for professional sports games and compute a Q-function, representing the chance that a team scores the next goal, for *all* actions. The Q-function assigns values to actions on the same scale by looking ahead to expected outcomes and effectively reflect the match context of evaluated action. Unlike the previous MDP models [84, 85] that required pre-discretizing game features, we utilize a deep reinforcement learning (DRL) approach to learn an action-value Q function for capturing the current match context. The neural network representation can easily incorporate continuous quantities like rink location and game time.

Evaluating Players using the Deep Q-function: Given a Q-function, we define the *impact* of action as the change in Q-value due to the action and introduce a novel Goal Impact Metric (GIM) to evaluate a player's performance by aggregating the impact of all actions of a player. We evaluate the performance of GIM in both ice hockey and soccer environments by studying its correlation with standard success measures and consistency within an entire game season. To our knowledge, this is the first player evaluation metric based on DRL. The context-aware Q-function can capture the spatio-temporal complexity of the home and away teams separately in a sports game. Based on the Q-function, the GIM metric measures both players' offensive and defensive contribution to goal scoring.

Interpreting the Q-function with Mimic Learning: A promising method for interpreting deep models is mimic learning. To understand the influence of game features on players' action values, we introduce a mimic learning framework for DRL and examine the approach of training regression trees as mimic models for interpreting the Q-function. Our baseline mimic model is a traditional Classification And Regression Tree (CART), with which we explain our approach of computing the feature importance and finding exceptional players. We then introduce a novel Linear Model U-Tree (LMUT), which adds a linear model at each leaf node to improve its representative power and generalization ability. LMUT learning is based on an online U-tree [99, 69] algorithm specially designed for Q-values in reinforcement learning . We not only evaluate LMUT as a general DRL interpreter under the virtual game environment but also study its performance for ice hockey and soccer. To our best knowledge, this is the first work that extends interpretable mimic learning to Reinforcement Learning.

Learning Player Representation with a Variational Encoder: To incorporate player information into sports statistics, we propose a novel player representation framework that learns player representations via player generation. The basic idea is to model which player is likely to be the acting player given a current match state/context and a current action. The generation process follows the design of Conditional Variational Auto-Encoder (CVAE): we learn a context-specific prior as a representation for game context, which allows deriving an approximate posterior as a contextualized representation for the observed acting player. Based on this framework, we introduce a Variational Recurrent Ladder Agent En-coder (VaRLAE). VaRLAE applies a ladder structure and learns a hierarchy of latent variables distributions to represent the player information . Unlike the traditional CVAE model that considers only the current observation, VaRLAE models the game history with the RNN design. We evaluate the performance of our VaRLAE on a massive National Hockey League (NHL) dataset containing over 4.5M events. To study how much the learned player representations improve downstream applications, we evaluate two major tasks in sports analytics: estimating expected goals and predicting final match score differences. Our empirical evaluation shows the improvement in predictive accuracy after incorporating the embeddings generated by VaRLAE.

1.3 Thesis Organization

The remainder of this dissertation is organized as follows:

- Chapter 2 introduces the background and related work on action-value metrics for player evaluation. We introduce the metrics based on single/ multiple action-value counts as well as Value-Above-Replacement approaches in Section 2.1. Section 2.2 also provides a review for the available dataset for sport analytic, including the box scores, the play-by-play dataset, the game tracking dataset, and the videos.

- Chapter 3 contains the preliminaries in this dissertation, covering the Markov model for professional team sports (Section 3.1), the domain knowledge for the evaluated game environments (Ice Hockey and Soccer, Section 3.2), as well as a detailed introduction to the sports dataset (Section 3.3) applied in this dissertation.

- In Chapter 4, we introduce our approach of learning the action-value Q-function with the Deep Reinforcement Learning (DRL) approach. We compute an action impact from the learned Q-function, based on which we define a Goal Impact Metric (GIM) (Section 4.5) for evaluating the player performance for the entire game season.

- In Chapter 5, we extend the DRL model and learn the Q-function for the soccer game, which has a large pitch with more player and sparser rewards than Ice Hockey. We present a fine-tuning method to model the league-specific player performance.

- Chapter 6 introduces the mimic learning framework for interpreting the Q-function learned by the DRL model. We present a novel Linear Model U-Tree (LMUT) as a mimic model, from which we can extract the knowledge and understand the influence of game features on the Q-function. We evaluate the LMUT under both the virtual game (Section 6.4) and sport environments (Section 6.3).

- Chapter 7 proposes our player representation framework (Section 7.3) and our Variational Recurrent Ladder Agent Encoder(VaRLAE , Section 7.4). We validate the performance of VaRLAE in two practical applications: estimating the expected goals and predicting the final score differences.

- Chapter 8 summarizes the major contributions of this dissertation (Section 8.1). We discuss the advantages and limitations of the methods (Section 8.2) in this dissertation. We describe some future directions (Section 8.3) for improving our action-value function, the mimic learning framework, and the player representation model.

Chapter 2

Background and Related Work

2.1 A Review of Action-Value Metrics

The most common application of action values is evaluating players' performance. In this section, we provide a review of the action-value metrics for players evaluation. A recent work [2] provides several up-to-date survey articles on player evaluation. As it is shown in Figure 2.1, the action-value based player evaluation metrics can be generally divided into two categories: 1) Action-Value Counts and 2) Value-Above-Replacement:

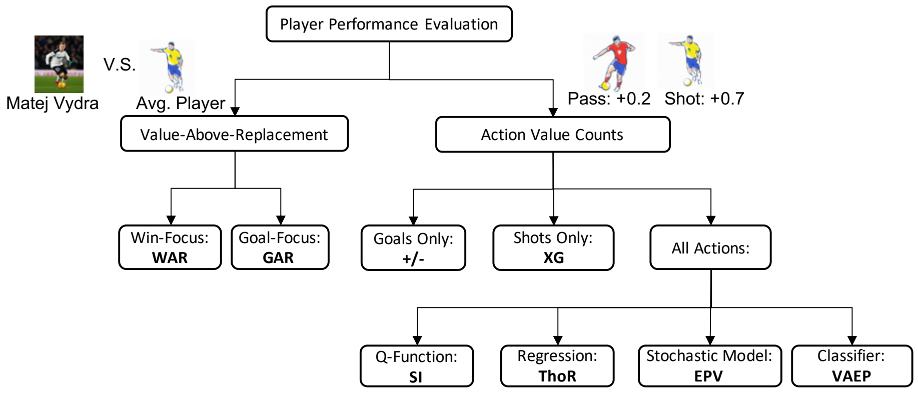


Figure 2.1: A tree diagram to position previous works about action-value based player evaluation metric. An important factor is whether a metric considers all actions or only a subset of them.

2.1.1 Single Action-Value Counts

Probably one of the simplest approaches to qualifying a player's contribution is counting the number of goals that he or she manages to score during a sports game. Following this intuition, to evaluate a player's performance, traditional player evaluation metrics focus only on the actions that have an immediate impact on the scoring probabilities, for example shooting and passing. Among all the single-action evaluation metrics proposed in recent years, the most popular and well-studied metrics are Plus-Minus and Expected Goal:

Plus/Minus (+/- or PM). +/- is a commonly applied player evaluation metric that is applicable to almost every professional sport. It qualifies the influence of a player's presence on the goal-scoring opportunity for his or her team. The basic version awards a player +1 if a goal is scored by the player's own team when the player is on the pitch, and -1 if the other team scores.

Some recent works proposed several effective modifications to the basic plus-minus metric. Instead of assigning equal importance to each goal, [86] weighted the goals basing on opponents' strength and the importance of a particular goal on expected win probability (e.g. game time and game frequency). A previous work [67] also proposed an adjusted plus-minus statistic for NHL players by applying a weighted least squares regression to estimate an NHL player's effect on his team's success in scoring and preventing goals. The weighting approach incorporates the influence of team performance and game environment into the basic +/-, and thus provides a more comprehensive evaluation of a player's overall performance. Another previous work [88] enhanced the adjusted +/- by ridge regression in which the data is combined with a prior belief regarding reasonable ranges for the parameters. In order to produce more accurate +/-, a recent work [52] applied machine learning and survival models to estimate both expected goals and expected points to assess a player's defensive and offensive influence. Table 2.1 shows a summary of these modifications.

Approaches	Summary
Basic +/-	+1/-1 if a goal is scored / lost.
Weighted +/- [86]	Weight the goals by their importance and opponents' strength.
Adjusted +/- [67]	Adjust +/- statistics with weighted least squares regression.
Enhanced +/- [88]	Enhance the adjusted +/- statistics with ridge regression.
Expected Goals +/- [52]	Estimate expected goals for computing +/-.
Expected Points +/- [52]	Estimate expected points for computing +/-.

Table 2.1: A summary of the previous +/- based approaches.

Expected Goals (XG). XG quantifies the value of a shot by the probability of the shot leading to a goal. A common method to compute the XG values is training a goal prediction model with shot features (e.g. x,y coordinates, or angle to goal) and assigning the soft outputs (expected goals) to players' shots. Players can be ranked by the summation of their total expected goals [6]. A recent work [68] extended the traditional XG by applying statistics (e.g. faceoffs and hits) as predictor variables in addition to goals, shots, missed shots, and blocked shots, to predict goals. They used ridge regression to estimate a player's contribution to his team's XGs.

Passing Quality. To study the effect of actions other than shots, several recent works have extended the application of traditional XG and evaluated players by measuring the quality of their

Approaches	Summary
Basic XG [6]	A goal prediction model that assigns values to shots.
Extended XG [68]	Estimate the XGs with more statistics by ridge regression.

Table 2.2: A summary of the previous XG based approaches.

passes. Passing is one of the most strategic and frequent actions in many team sports, such as soccer and basketball. For each pass, a previous work [17] measured its value as the estimated probability of resulting in a successful shot. Another recent work [15] evaluated each pass by the difference between the goal-scoring probability before and after the pass.

2.1.2 Multiple Actions-Value Counts

The single action-value based metrics evaluate the players by measuring only one type of action with an immediate impact on the scoring. However, to evaluate a player's overall performance, we should not only measure players' offensive influence (e.g. increasing their team's scoring chance) but also consider their defensive contribution (e.g. preventing their opponent scoring). Several recent works have proposed some more advanced action-value metrics that assess players by evaluating **all** their actions, including both offensive actions (e.g. shot, assist) and defensive actions (e.g. block and tackle). We provide a summary of the major multi-action-value metrics:

Approaches	Summary
EPV [21, 51]	Modele the evolution of a sports possession with a stochastic process model.
Deep EPV [32]	Build a neural model to compute the expected possession values for all players.
VAPE [29]	Evaluate an action by its impact on scoring probabilities within a few future steps.
THoR [82]	Evaluate an action by the probability that it leads to a goal (within 20s).
SI [80]	Compute action impacts with the Markov model and the dynamic programming.
SI-Loc [84, 85]	Improve SI by incorporating location information of players into evaluation.

Table 2.3: A summary of the multiple actions-value counts approaches.

Expected Possession Value (EPV). EPV [21] evaluates all the players' actions within a continuous possession by estimating the expected number of points that will be score during the current possession. They modeled the evolution of a complete sports possession with a stochastic process model, which was implemented at multiple levels of resolution, differentiating between continuous, infinitesimal movements of players, and discrete events such as shot attempts and turnovers. A previous work [51] applied a similar EPV-based framework to estimate the expected point value for starting possessions in different field locations during a professional rugby league match-play. They also compute the mean expected points for each subsequent play during the possession. Based on the EPV framework, a recent work [32] built a deep model from the full resolution spatial-temporal data of 22 soccer players on the pitch and computed the expected possession values for all actions during a game. They study the action impacts of individual soccer players under different game situations. Computing the EPV-based often requires tracking data that has the complete observability of all players. Many play-by-play datasets, however, provide only partial observability of game context: they record only actions of the players who possess the ball at a given time.

Valuing Actions by Estimating Probabilities (VAEP). The VAEP framework [29] evaluates all on-the-ball actions of soccer players based on their influence on the game outcome. They also proposed a new language SPADL for describing individual player actions on the pitch. Applying SPADL, their model considers a set of hand-crafted action features from the recent game history and evaluates an action by its impact on scoring probability within a constant number of future steps. To quantify a player's overall performance, their model computes the change in scoring and conceding probability after a player's action and summarizes the probabilities overall the entire season. Total Hockey Rating (THoR) [82] applied a similar method and they determined the impact of each play by the probability that it leads to a goal for a player's team (or their opponent) in the subsequent 20 seconds. Their rating that accounts for all on-the-ice players as well as the impact of where a shift starts and of every non-shooting events such as turnovers and hits.

Scoring Impact (SI) SI [80] measures a player's performance by summarizing their impact over a game. The impact is the difference of scoring probabilities before and after the player controls the ball (or puck in ice hockey). To compute the scoring probabilities, [80] formulated an ice hockey play-by-play dataset into a Markov Game Model [61], where actions record the player movements and states capture the game context. They modeled event data of the form $s_0, a_0, r_1, s_1, a_1, \ldots, s_t, a_t,\ r_{t+1}, s_{t+1}, a_{t+1}$: environment state s_t (represented by three features: Goal Differential, Manpower Differential and Period) occurs, an action a_t is chosen, resulting in a reward r_{t+1} and state s_{t+1}. At the next time step, another action a_{t+1} is chosen. The data are often separated into local **transitions** of the form $\mathcal{T}\{s, a, r', s', a'\}$. Under this setting, a model-based Reinforcement Learning (RL) approach [94] has been applied to calculate the action values for each players. The RL approach computes the expected scoring probabilities of player actions under different game context by a Q-function using dynamic programming [76] based on the Bellman equation and calculate the impacts of player actions by:

$$Q(s,a) = \mathbb{E}_{s',a'}[r' + Q(s',a')|s,a] = \sum_{r'} p(r'|s,a)r' + \sum_{s'} \mathcal{P}_a(s',s) \sum_{a'} \pi(a'|s')Q(s',a') \quad (2.1)$$

$$\text{impact}(s,a) = Q(s,a) - V(s) \text{ and } V(s) = \sum_{a} \pi(a|s)Q(s,a) \quad (2.2)$$

The recurrence allows us to estimate the Q value at a current context s, a given an estimate for the next Q values and transition probabilities. Applying similar approaches, continuing works[84, 85] further added location information to the Markov model (by discretizing a hockey rink according to action distributions). They applied maximum likelihood estimates for the resulting discrete transition probabilities. The discretization leads to loss of information and undesirable spatial-temporal discontinuities in the Q-function. These drawbacks prohibit the model from generalizing to the unobserved part of state space.

2.1.3 Value-Above-Replacement

Value-Above-Replacement (VAR). VAR evaluates players by measure how much they outperform the average player in their league. The most common VARs include Goals Above Replacement (GAR) and Wins Above Replacement (WAR) which measures the player's contribution to his or her team by estimating the difference of team's scoring or winning chances when the target player plays, vs. replacing him or her with an average player. GAR and WAR have been applied as player evaluation metrics for team sports like baseball [1], basketball [2], ice-hockey [3] and American football, there has not been an agreed-upon definition for soccer [4], because soccer has a more complex game context including a large football pitch over 7k square meters and 22 on-the-pitch players.

2.2 A Review of Sports Data

In this section, we provide a detailed review of the available sports datasets by 1) describing the data format, 2) showing some data examples, and 2) discussing the approach of applying them for Sports Analytics.

2.2.1 Box Scores

Box scores (or player/team statistics) are commonly applied in team sports like baseball, basketball, football, and ice hockey, showing a structured summary of the results from professional sports games. A box score often summarizes the game score as well as individual and team achievements in the game or during a game season (e.g. goal score, assistants, etc.).

Figure 2.4 and Figure 2.5 show the examples of box scores for players and teams respectively. Box score directly uses counting numbers to demonstrate teams' and player's performance in each game. These numbers can be aggregated along an entire game season to analyze a team or a player's general performance. The main drawbacks of box scores are 1) instead of concluding a player's overall performance with one metric, box score often applies different numbers to represent a player's ability in different aspects (e.g., defensive and offensive performance). 2) Box scores, based on action counting, assign the same value to each action or goal and neglects the influence of manpower

[1] https://library.fangraphs.com/misc/war/

[2] https://www.nbastuffer.com/analytics101/wins-above-replacement-player-warp/

[3] https://hockey-graphs.com/2019/01/16/wins-above-replacement-history-philosophy-and-objectives-part-1/

[4] https://www.americansocceranalysis.com/home/2019/1/11/points-above-replacement

or game time (some previous [86, 67] works about weight/adjusted +/- metrics have discussed the similar issue). 3) As the summary of the player's or team's success, box scores often omit the context features and thus lose much key information.

Player	Team	Date	MATCHUP	W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	3P%
Trae Young	ATL	03/01/2019	ATL vs. CHI	L	56	49	17	33	51.5	6	13	46.2
Damian Lillard	POR	03/18/2019	POR vs. IND	W	40	30	9	16	56.3	4	9	44.4
Bradley Beal	WAS	01/13/2019	WAS vs. TOR	L	55	43	17	36	47.2	6	12	50.0
James Harden	HOU	01/03/2019	HOU @ GSW	W	44	44	13	32	40.6	10	23	43.5
Bradley Beal	WAS	12/22/2018	WAS vs. PHX	W	54	40	17	33	51.5	4	12	33.3
De'Aaron Fox	SAC	11/01/2018	SAC @ ATL	W	34	31	9	14	69.2	3	4	75.0

Table 2.4: An example of box scores for NBA players. It summarizes the player performance of a game and reports the basketball statistics including playing Minutes (Min), points (PTS), field goals made/attempted/percentage (FGM/FGA/FG%) and three-point field goals made/attempted/percentage (3PM/3PA/3P%).

Team	Date	MATCHUP	W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	3P%
DEN	10/17/2019	DEN vs. POR	W	241	110	41	89	46.1	8	28	28.6
MEM	03/20/2019	MEM vs. HOU	W	265	126	41	89	46.1	13	37	35.1
HOU	03/08/2019	HOU vs. PHI	W	239	107	42	85	49.4	13	41	31.7
IND	02/23/2019	IND @ WAS	W	240	119	47	83	56.6	11	25	44.0
TOR	02/05/2019	TOR @ PHI	W	239	119	41	89	46.1	11	31	35.5
DEN	01/13/2019	DEN vs. POR	W	239	116	45	81	55.6	9	22	40.9

Table 2.5: An example of box scores for NBA teams. It summarizes the team performance of a game and reports the basketball statistics including playing Minutes (Min), points (PTS), field goals made/attempted/percentage (FGM/FGA/FG%) and three-point field goals made/attempted/percentage (3PM/3PA/3P%).

2.2.2 Play-by-Play Dataset

A Play-by-play dataset consists of the logs of discrete action events specifying various properties of the action (e.g. action type, acting player, time, and location). The dataset tracks action events around the ball from the beginning until the end of a game. An action event often records the actions of on-the-ball players as well as the spatial and the temporal context features.

Table 2.6 shows an example of the play-by-play soccer dataset, where each event records context features including X-Y coordinate, manpower, score differential, and game time remains. Compared to the box scores, a play-by-play dataset provides more detailed context features for each action. The fine-grained data allows a more advanced machine learning algorithm to model the spatial and temporal features. For example, a previous work [85] built a Markov model where they used the context features (including manpower, score differential, and game time) to define the states and computed the transition probabilities between those states according to the occurrences of these state in the play-by-play dataset. However, a major drawback of the play-by-play data is that it provides only partial information. Instead of recording the actions and the locations of *all* the players on the court, the data often records only the action of the on-the-ball player (player who controls the ball) and neglect the position and movement of other players. This issue becomes more serious for the team sport with a complex game context (e.g. soccer). To alleviate the partial observability, some previous works [29] have attempted to include the play history into the evaluation, but what information should be included and how to include this information still remain unsolved.

MP=Manpower, GD=Goal Difference, OC = Outcome, S=Succeed, F=Fail, H=Home, A=Away, T=Team who performs action, GTR = Game Time Remain, ED = Event Duration

GTR	X	Y	MP	GD	Action	OC	Velocity	ED	Angle	T	Reward
35m44s	87	26	Even	1	simple pass	S	(2.2, 1.7)	11.0	0.19	H	[0,0,0]
35m42s	90	17	Even	1	standard shot	F	(1.5, -4.5)	2.0	0.11	H	[0,0,0]
35m42s	99	44	Even	1	save	S	(0, 0)	0.0	0.06	A	[0,0,0]
35m9s	100	1	Even	1	cross	S	(0.0, -1.3)	33.0	0.0	H	[0,0,0]
35m7s	85	56	Even	1	simple pass	S	(-7.3, 27.6)	2.0	0.39	H	[0,0,0]
35m5s	92	67	Even	1	simple pass	S	(3.6, 5.4)	2.0	0.28	H	[0,0,0]
35m4s	97	50	Even	1	corner shot	S	(5.1, -16.2)	1.0	1.74	H	[0,0,0]
35m4s	100	50	Even	1	goal	S	(0, 0)	0.0	0.0	H	[1,0,0]
....	...	...	....	...	....	...	....	...	.....	.	......
3m41s	62	96	Even	2	long ball	F	(4.5, 9.3)	9.0	0.08	A	[0,0,0]
3m39s	19	89	Even	2	clearance	S	(-21.5, -3.2)	2.0	0.07	H	[0,0,0]
3m35s	24	100	Even	2	throw in	S	(1.3, 2.7)	4.0	0.09	A	[0,0,0]
3m33s	27	96	Even	2	simple pass	S	(1.1, -2.2)	2.0	0.1	A	[0,0,0]
3m31s	12	95	Even	2	cross	S	(-7.5, -0.5)	2.0	0.07	A	[0,0,0]
3m28s	6	46	Even	2	simple pass	S	(-1.7, -16.3)	3.0	0.79	A	[0,0,0]
3m26s	14	48	Even	2	standard shot	S	(3.8, 1.3)	2.0	0.44	A	[0,0,0]
3m26s	0	50	Even	2	goal	S	(0, 0)	0.0	0.0	A	[0,1,0]

Table 2.6: An example of a play-by-play data sample featuring team scoring: a sequence of events where home team scores and then away team scores. The rewards [1,0,0] and [0,1,0] indicate the scoring event of the home team and away team respectively. We skip some events in the middle due to space issues.

2.2.3 Game Tracking Dataset

A game tracking dataset is constructed by video-tracking techniques (e.g. activity recognition [46]), recording the locations of players (including the players controlling or not controlling the ball/puck) and the game time. Compared to the play-by-play data, the game tracking data records the location of each player at more dense time intervals. Figure 2.2 shows an example of tracking the location of on-the-ball player from the broadcast video frames. The broadcast video, however, contains only part of the players on the pitch, and thus provides only partial observability of the game. To overcome this limitation, SPORTLOGiQ [5] provides a multi-camera hardware solution. By installing multiple cameras around stadiums and extracting players' action data from the collected video with computer vision techniques (e.g. object detection [46]), they achieve full observability of all 22 soc-

[5]https://sportlogiq.com/en/

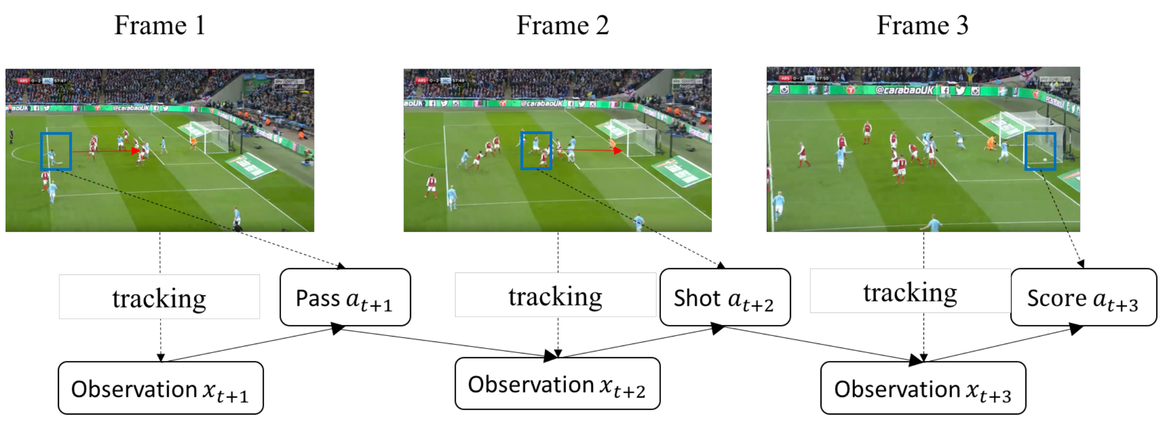


Figure 2.2: An example of video tracking: at each video frame, the tracking system dynamically locates the position of on-the-ball player (in blue frame) with the object detection algorithm and records the player action. The system might require manually labeling the position of the ball (or the puck) for the sports with a large pitch (or rink).

cer player on the pitch. The tracking data fuels many models that requiring the full observability of a game at every time step (e.g. EPV).

2.2.4 Video Dataset

Video dataset consists of broadcast videos (records part of the players on the court, for example, the video frames in Figure 2.2) or multi-camera videos (records all the players on the court). These videos contain the richest information about a sports game, but computing the action values from videos is challenging, because of the large numbers and the huge dimension of raw images. Existing action-value metrics often require an extra data preprocessing step that extracts human-understandable game features from these videos, during which much important information is lost. In future work, we discuss the approach of learning a deep latent representation from raw images with Variational Auto-Encoder. The learned latent representation is interpretable with significantly fewer dimensions than raw images. This approach enables a complex machine learning model to directly run on the complex video data.

Chapter 3

Preliminaries

3.1 Markov Model for Sports Games

We use the classical framework of Markov games [61] to model the sports game. A Markov game extends the general formulation of Partially Observable Markov Decision Processes (POMDP) to the multi-agent setting. In this chapter, we provide an introduction to the POMDP and the approach of extending it to a Markov Game Model.

3.1.1 Partially Observable Markov Decision Processes

A Markov Decision Process (MDP) [45] is a discrete-time stochastic control process that models the relationship between an agent and its environment. In a sports game, a POMDP can be defined by a six-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \gamma \rangle$. At each time step t, an agent performs an action $a_t \in \mathcal{A}$ at a game state $s_t \in \mathcal{S}$ after receiving an observation $\boldsymbol{o} \in \Omega$. The observation received by the agent describes only partial information of a game state, for example, a player can only observe the players around him (without knowing the complete actions and locations of all other players). This process generates the reward [1] $r_t \sim R(s_t, a_t)$ and the next state $s_{t+1} \sim \mathcal{P}_{a_t}(s_{t+1}, s_t)$ ($\mathcal{P}$ defines the conditional transition probabilities between states: $p(s_{t+1}|a_t, s_t)$). $\gamma \in (0, 1]$ is discount factor for a player's total return. In this dissertation, we assume the goals have equal importance, so we set $\gamma = 1$. A DRL agent determines actions according to a policy. The policy π of a player can be defined as a distribution over actions given states: $\pi : \mathcal{A} \times \mathcal{O} \rightarrow [0, 1]$. Under the general control setting, the goal of a player is to find a policy π mapping a given game state to a current choice of action to maximize an expected cumulative of discounted reward, which is given by an action-value function

[1]In this dissertation, $\sim$ indicates "is sampled from" or "has the distribution of".

$Q^{\pi}(s_t)$. The Q function can be given by:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_r\Big\{\sum_{j=0}^{\infty} \gamma^j r_{t+j} | s_t, a_t\Big\} \tag{3.1}$$

Given a transition of the form $[s, a, r', s', a']$, we can estimate the value function by applying dynamic programming [76] based on the Bellman equation:

$$V^{\pi}(s) = \sum_{a} \pi(a|s) Q^{\pi}(s, a) \tag{3.2}$$

$$V^{\pi}(s) = \sum_{a} \pi(a|s)[\sum_{r'} p(r'|s, a) r' + \sum_{s'} \mathcal{P}_a(s', s) V^{\pi}(s')] \tag{3.3}$$

3.1.2 Markov Game Model

A Markov game [61] extends the general formulation of POMDP to the multi-agent setting. A Markov Game can be represented as a six-tuple $G = \langle \mathcal{S}, \boldsymbol{\mathcal{A}}, \mathcal{P}, \boldsymbol{\mathcal{R}}, \Omega, \gamma \rangle$, where $\mathcal{S}$ is a finite set of states, $\boldsymbol{\mathcal{A}} = (a_1, \ldots, a_K)$ is a collection of finite action sets, one for each agent $1, \ldots, K$. At a discrete time t, a Markov game model defines a real-valued reward for each agent: $\boldsymbol{r}_{1,\ldots,K,t} \sim \mathbf{R}(s_t, a_{1,t}, \ldots, a_{K,t})$. The discount factor $\gamma = 1$ are shared across different agents. The transition function captures the dynamics in sports environment: $s_{t+1} \sim \mathcal{P}a_{1,t}, \ldots, a_{K,t}(s_{t+1}, s_t)$. Under the traditional controlling environment, the goal of an agent k is maximizing the expected cumulative of discounted reward, which is estimated by a value function:

$$V_{k,t} = \mathbb{E}\{\sum_{j=0}^{\infty} \gamma^j r_{i,t+j}\} \tag{3.4}$$

3.1.3 Play Dynamics in Sports Games

We introduce our approach of formulating the sports game into a Markov Model. For each game, based on the Markov Game model, we consider event data of the form $[(\boldsymbol{o}_0, pl_0, a_0, r_0), (o_1, pl_1, a_1, r_1), \ldots, (\boldsymbol{o}_t, pl_t, a_t, r_t), \ldots]$: at time t, after observing $\boldsymbol{o}_t$, player pl_t takes a turn (possesses the puck) and chooses an action a_t, which produces a reward r_t denoting whether a goal is scored. The $\boldsymbol{o}$, a and r are determined by:

- In each sport game, we represent an **observation** $\boldsymbol{o}_t$ by a feature vector for discrete time step t that specifies a value for the features listed in Table 3.2 (for ice-hockey) and Table 3.5 (for soccer). To alleviate the partial observability, similar to [41], We use the complete sequence $s_t = [\boldsymbol{o}_t, a_{t-1}, \boldsymbol{o}_{t-1}, a_{t-2}, \boldsymbol{o}_{t-2}, ...]$ as the **state** representation at time step t [71].

- The **action** a_t denotes the movements of players who control the ball. Our model applies a discrete action vector using one-hot encoding, where we represent the action by a 0-1 vector. The number of vector dimension equals the number of action types. We label 1 for the represented action and 0 for others.

- The **reward** r_t is a vector of goal values g_t that specifies which team ($Home$, $Away$) scores. We introduce an extra $Neither$ indicator for the eventuality that neither team scores until the end of a game. For readability, we use $Home$, $Away$, $Neither$ to denote the team in a 1-of-3 vector of goal values $r_t = [g_{t,Home}, g_{t,Away}, g_{t,Neither}]$ and $g_{t,Home} = 1$ indicates the home team scores at time t (see Table 2.6).

- The **player** pl_t records the identity of the current acting player. We represent a discrete player vector with the one-hot encoding.

- In this dissertation, we explore different options of defining **agents**: 1) In the Chapter 4, Chapter 5 and Chapter 6, we only consider two agents, $Home$ and $Away$, representing the teams of players. 2) In the Chapter 7, we expand this definition and consider modeling the behavior of each individual players. Under this setting, the number of agents equals the number of player in a professional league.

3.2 Domain Knowledge: an Introduction to the Game Rules

This dissertation mainly studies two sports games: Ice Hockey and Soccer. In this section, we provide an introduction to the rules for these sports domains.

3.2.1 Ice Hockey

Ice hockey is a fast-paced team sport, where two teams of skaters must shoot a puck into their opponent's net to score goals. This dissertation mainly studies professional ice hockey in the National Hockey League (NHL) which is one of the major and the most popular ice hockey league in the world. In this section, we provide an introduction to the NHL game rules.

Game Environment. Figure 3.1 shows an ice-hockey rink where a game is played. An NHL game has a total of 60 minutes of game time. The entire game is split into three periods and each period has 20 minutes. During normal play, both sides (the visiting team and the home team) can have 6 players on the ice. The positions of these players are Centre (one man), Left/Right Wing (two men), defensemen (two men), and Goalie (one man).

Game Winning Rules. During an NHL game, a goal is scored if one team shots the puck (a disk made of vulcanized rubber) into the goal of their opposing team. To win the game, a team must score more goals than its opposing team. If a game is tied after the regular game time, there is overtime and whoever scores the first goal will win the game. If no goal is scored after the overtime,

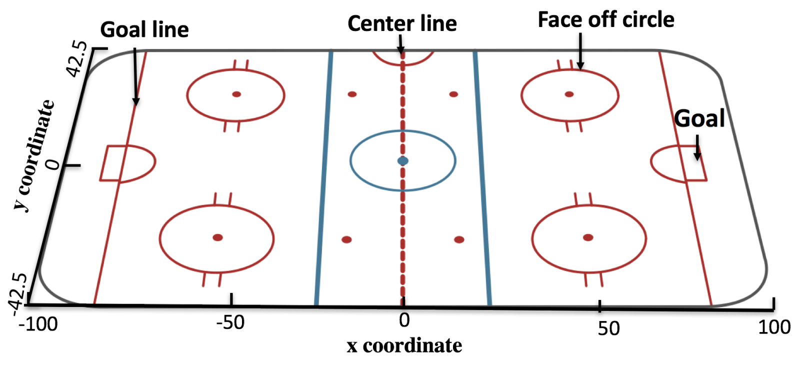


Figure 3.1: Ice Hockey Rink. The hockey rink It measures 200 feet (60.96 m) by 85 feet (25.91 m), which can be divided into three zones by two blue lines. The zone between two blue lines is referred to as a neutral zone. The zone with the team's own goal is called a defensive zone. The zone toward which a team's play flows (or attacks) is the offensive zone.

during the regular season, the game moves into a shootout: three players for each team in turn take a penalty shot. The team with the most goals during the three-round shootout wins the game. During playoffs, overtime is repeated until a goal is scored.

Penalty. NHL referee can call a penalty to punish infringement of the NHL rules. Most penalties are enforced by sending the offending player to a penalty box for a set number of minutes. The offending team may not replace the player on the ice, leaving them *Short-Handed* and their opposing team being on a *Power Play*. The power-play team can have one more player on the ice than the short-handed team.

3.2.2 Soccer

Soccer (or association football) is a team sport played with a spherical ball on a rectangular field (a pitch with a goal at each end). In this dissertation, we mainly study the professional soccer in the European Soccer Leagues and introduce the corresponding domain knowledge as follows:

Game Environments. Figure 3.2 shows a soccer pitch. A standard football match consists of two halves of 45 minutes each. After a full-time match, the referee may make an allowance for time lost through substitutions, injured players requiring attention, or other stoppages. Both teams (the visiting team and the home team) can have 11 players on the pitch, usually including a goalkeeper, two fullbacks, two center backs, five midfielders/wingers and a striker.

Game Winning Rules. Similar to the ice hockey game, to win a soccer match, the players must score more goals by shooting a ball to the goal of their opposing team. There is overtime or extended play period in league play.

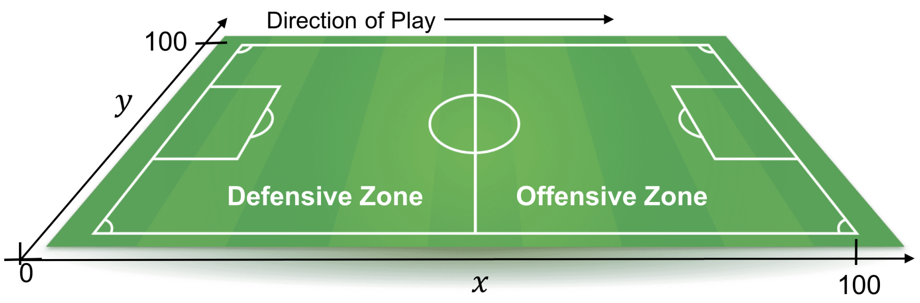


Figure 3.2: Soccer pitch layout with adjusted coordinates. Coordinates are adjusted so that for the home/away team performing an action, its offensive zone is on the right

Penalty. To make players play the game in a fair manner, the referee can call fouls. The penalty from a foul depends on the type and severity of the foul. The minor offenses award an indirect free kick to the opposing team. For the more serious offenses, the opposing team is awarded a direct free kick. This will become a penalty kick if it occurs within the penalty area. A yellow card can be given for repeated fouls. The second yellow results in a red and expulsion from the game (The player must leave the game and cannot be substituted for). A red card leaves a short-handed situation for the offending team and a power-play for their opposing team.

3.3 The Play-by-Play Sports Dataset

To study the action values for ice hockey and soccer, we utilize an NHL and a European soccer play-by-play dataset. In this section, we provide a brief introduction to these datasets.

3.3.1 The Ice Hockey Datatset

The ice hockey dataset that we utilize is constructed by SPORTLOGiQ [2] with computer vision techniques. It provides information about **game events** and **player actions**. Table 3.1 shows the statistics. Table 3.3 shows an excerpt. The data tracks events around the puck, and record the identity and actions of the player, with space and time stamps, and features of the game context. The unit for space stamps are feet and for time stamps seconds. We utilize adjusted spatial coordinates, where negative numbers refer to the defensive zone of the acting player, positive numbers to his offensive zone. Adjusted X-coordinates (XAdjcoord) run from -100 to +100, Y-coordinates (YAdjcoord) from 42.5 to -42.5, and the origin is at the ice center. We include data points from all manpower scenarios, not only even-strength, and add the manpower context as a feature. We did not include overtime data. Period information is implicitly represented by game time. The complete feature set in Table 3.2.

[2]https://sportlogiq.com/en/

Season	Events	Players	Teams
2015-16	3,502,317	1,164	30
2016-17	4,320,566	1,230	31
2017-18	4,643,224	1,271	31
2019-20	4,534,017	1,196	31

Table 3.1: Dataset statistics for the Ice Hockey dataset. The basic unit of this dataset is an **event**, which describes the game context and the on-the-ball action of a player at a time step.

Name	**Type**	**Range**
X Coordinate of ball/puck	Continuous	[-100, 100]
Y Coordinate of ball/puck	Continuous	[-42.5, 42.5]
Velocity of ball/puck	Continuous	(-inf, +inf)
Game Time Remain	Continuous	[0, 3600]
Score Differential	Discrete	(-inf, +inf)
Manpower Situation	Discrete	{ES, SH, PP}
Event Duration	Continuous	[0, +inf)
Action Outcome	Discrete	{Successful, Failure}
Angle between ball/puck and goal	Continuous	$[-3.14, 3.14]$
Home or Away Team	Discrete	{Home, Away}

Table 3.2: The complete feature list for an observation, where the manpower situation is marked by Event Strength (ES), Short-Handed (SH) and Power Play (PP).

GID=GameId, PID=playerId, GT=GameTime, TID=TeamId, MP=Manpower, GD=Goal Difference, OC = Outcome, S=Succeed, F=Fail, P = Team Possess puck, H=Home, A=Away, H/A=Team who performs action, TR = Time Remain, PN = Play Number, D = Duration

GID	PID	GT	TID	X	Y	MP	GD	Action	OC	P
1365	126	14.3	6	-11.0	25.5	Even	0	Lpr	S	A
1365	126	17.5	6	-23.5	-36.5	Even	0	Carry	S	A
1365	270	17.8	23	14.5	35.5	Even	0	Block	S	A
1365	126	17.8	6	-18.5	-37.0	Even	0	Pass	F	A
1365	609	19.3	23	-28.0	25.5	Even	0	Lpr	S	H
1365	609	19.3	23	-28.0	25.5	Even	0	Pass	S	H

Table 3.3: Dataset Example

Velocity	TR	D	Angle	H/A	PN
(-23.4, 1.5)	3585.7	3.4	0.250	A	4
(-4.0, -3.5)	3582.5	3.1	0.314	A	4
(-27.0, -3.0)	3582.2	0.3	0.445	H	4
(0, 0)	3582.2	0.0	0.331	A	4
(-30.3, -7.5)	3580.6	1.5	0.214	H	5
(0,0)	3580.6	0.0	0.214	H	5

Table 3.4: Derived Features

3.3.2 The Soccer Dataset

The soccer dataset we apply is an F24 play-by-play soccer game dataset provided by Opta [3]. The dataset records the play-by-play information of game events and player actions for the entire 2017-2018 game season from multiple soccer leagues, including English Premier League, Dutch Eredivisie, EFL Championship, Italian Serie A, German Bundesliga, Spanish La Liga, French Ligue 1

[3]https://www.optasports.com/

Name	**Type**	**Range**
Game Time Remaining	Continuous	[0, 100]
X Coordinate of ball	Continuous	[0, 100]
Y Coordinate of ball	Continuous	[0, 100]
Manpower Situation	Discrete	[-5, 5]
Goal Differential	Discrete	(-∞, +∞)
Action	Discrete	one-hot representation
Action Outcome	Discrete	{success, failure}
Velocity of ball	Continuous	(-∞, +∞)
Event Duration	Continuous	[0, +∞)
Angle between ball and goal	Continuous	$[-\pi, +\pi]$
Home or Away Team	Discrete	{Home, Away}

Table 3.5: Complete feature list. For the feature manpower situation, negative values indicate short-handed, positive values indicate power play.

Dataset	**F24**
Events	4,679,354
Players	5,510
Games	2,976
Teams	164
Leagues	10
Season	2017-18
Place	Europe

Table 3.6: Dataset statistics. The basic unit of this dataset is an **event**, which describes the game context and the on-the-ball action of a player at a time step.

and German Bundesliga Zwei. Table 3.6 shows more statistics. The dataset records the actions of on-the-ball players and the spatial and temporal context features. The complete feature set is listed in Table 3.5. Table 2.6 lists a series of events describing a goal sequence for the home and away teams. The dataset utilizes adjusted spatial coordinates. Both the X-coordinates and Y-coordinates are adjusted to [0, +100], where small numbers refer to the defensive zone of the acting player, and large numbers refer to the offensive zone. The adjusted soccer pitch is shown in Figure 3.2, where play flows from left to right for either team. To adjust coordinates, we reverse them when the team in possession attacks towards the left, so in this case $X_{Adjusted} = -rescale(X)$ and $Y_{Adjusted} = -rescale(Y)$. The adjusted coordinates accelerate model convergence during training and improve our model's fit for spatial features.

Chapter 4

Learning an Action-Value function for Evaluating Ice Hockey players with Deep Reinforcement Learning

In this chapter we show how Q-functions obtained from deep reinforcement learning can be used to assess the performance of professional hockey players. To make this chapter self-contained, we summarize the main background material. For more details, please see Chapters 2 and 3.

4.1 Introduction: Valuing Actions of Ice Hockey Players

With the advancement of high-frequency optical tracking and object detection systems, more and larger event stream datasets for sports matches have become available. There is an increasing opportunity for large-scale machine learning to model complex sports dynamics. Player evaluation is a major task for sports modeling that draws attention from both fans and team managers, who want to know which players to draft, sign, or trade. Many models have been proposed [29, 19, 67, 50]. The most common approach has been to quantify the value of a player's action, and to evaluate players by the total value of the actions they took [83, 70].

However, traditional sports models assess only the actions that have an immediate impact on goals (e.g. shots), but not the actions that lead up to them (e.g. pass, reception). And action values are assigned taking into account only a limited context of the action. But in realistic professional sports, the relevant context is very complex, including game time, positions of players, score and manpower differential, etc.

Recently, Markov models have been used to address these limitations. [80] used states of a Markov Game Model to capture game context and compute a Q function, representing the chance that a team scores the next goal, for **all** actions. [20] applied a competing risk framework with Markov chain to model game context, and developed EPV, a point-wise conditional value similar to a Q function, for each action. The Q-function concept offers two key advantages for assigning values to actions [84, 29]: 1) All actions are scored on the same scale by looking ahead to expected outcomes. 2) Action values reflect the match context in which they occur. For example, a late check near the opponent's goal generates different scoring chances than a check at other locations and times.

The states in the previous Markov models represent only a partial game context in the real sports match, but nonetheless, the models assume full observability. Also, they pre-discretized input features, which leads to loss of information. In this chapter, we utilize a deep reinforcement learning (DRL) model to learn an action-value Q function for capturing the current match context. The neural network representation can easily incorporate continuous quantities like rink location and game time. To handle partial observability, we introduce a possession-based Long Short Term Memory (LSTM) architecture that takes into account the current play history. Unlike most previous work on active reinforcement learning (RL), which aims to compute **optimal strategies** for complex continuous-flow games [41, 71], we solve a prediction (not control) problem in the passive learning (on policy) setting [93]. *We use RL as a behavioral analytics tool for real human agents, not to control artificial agents.*

Given a Q-function, the **impact** of an action is the change in Q-value due to the action. Our novel Goal Impact Metric (GIM) is computed as follows. We define the **impact** of an action as the change in Q-value due to the action, and aggregates the impact of all actions of a player. To our knowledge, this is the first player evaluation metric based on DRL. The GIM metric measures both players' offensive and defensive contribution to goal scoring. An alternative to the action-value approach is to compare a player to a random or league-average player (e.g.,[20]). This compares the expected success (e.g. the number of team wins) between the situations where the player is fielded and the situation if the player is replaced by a random or average player. We adopt this idea to introduce a new approach for play-by-play data that defines a natural Q-value-above-average-replacement metric for player performance measurement. Our main theorem states that a player's Q-value-above-average-replacement gives the same score as their total action impact value. This means that the DRL framework unifies the two fundamental approaches to player evaluation; the plausibility of the average replacement approach supports our total action-value metric (GIM). For player evaluation, similar to clustering, ground truth is not available. A common methodology [80, 75] is to assess the predictive value of a player evaluation metric for standard measures of success. Empirical comparison between 7 player evaluation metrics finds that 1) given a complete season, GIM correlates the most with 12 standard success measures and is the most temporally consistent metric, 2) given partial game information, GIM generalizes best to future salary and season total success.

4.2 Task Formulation and Approach

Player evaluation (the "Moneyball" problem) is one of the most studied tasks in sports analytics. Players are rated by their observed performance over a set of games. Our approach to evaluating players is illustrated in Figure 4.1. Given dynamic game tracking data, we apply Reinforcement

Learning to estimate the **action-value** function $Q(s, a)$, which assigns a value to action a given game state s. We define a new player evaluation metric called **Goal Impact Metric (GIM)** to value each player, based on the aggregated impact of their actions, which is defined in Section 4.5.1 below. Player evaluation is a descriptive task rather than a predictive generalization problem. So there is no separate test set and all metrics are computed on the training set only. As game event data does not provide a ground truth rating of player performance, our experiments assess player evaluation as an unsupervised problem in Section 4.6.

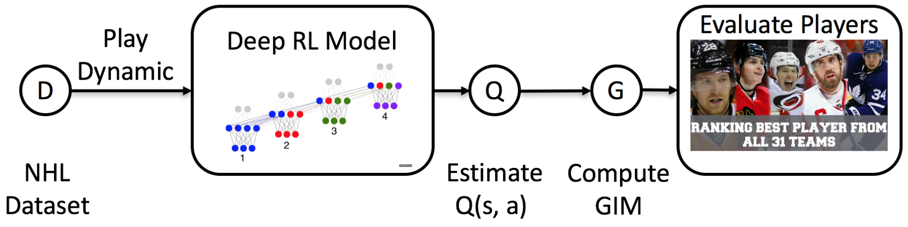


Figure 4.1: System Flow for Player Evaluation

4.3 Play Dynamics in NHL

The NHL Games Model. We apply the Markov Game Framework [61] to model the play dynamics for for NHL play. The basic building blocks of the model are described in preliminaries (section 3.1.3).

The Next-Goal Q-Function. Several value functions have been used to evaluate player actions. One option is to measure actions by whether they increase the winning chances [80]. More recent works focus on an action's more immediate impact regarding scoring points or goals [21, 85]. We formalize this idea in terms of the **next-goal Q function**, which is defined as follows.

We divide a sports game into **goal-scoring episodes**, so that each episode 1) starts at the beginning of the game, or immediately after a goal, and 2) terminates with a goal or at the end of the game. The next-goal Q-function represents the probability that the home/ away team *scores the goal at the end of the current goal-scoring episode* ($g_{Home} = 1/g_{Away} = 1$), or neither team scores ($g_{Neither} = 1$):

$$Q^{team}(s, a) = p(g_{team} = 1|s_t = s, a_t = a) \tag{4.1}$$

where $team$ is a placeholder for one of $Home, Away, Neither$. This Q-function represents **the probability that a team scores the next goal**, given current play dynamics in a sports game [84, 80]. For player evaluation, the next-goal Q-function has several advantages over win probabilities.

- Compared to final match outcome, the Q values model the probability of scoring the next goal that is a relatively short time away and thus easier to explain and understand.

- Increasing the probability that a player's team scores the next goal captures both offensive and defensive value. For example, a defensive action like tackling decreases the probability that the other team will score the next goal, thereby increasing the probability that the player's own team will score the next goal.
- The next-goal reward captures what a coach expects from a player. For example, instead of thinking about how the game will end, a coach prefers his players to focus on defending against their opponent's offensive play and creating the next scoring opportunities at the moment.

4.4 Learning Q values with DP-LSTM Sarsa

We take a function approximation approach and learn a neural network that represents the Q-function ($Q^{team}(s, a)$).

4.4.1 Network Architecture

Figure 4.2 shows our model structure. The inputs of our neural model are actions and states for ice hockey games (check a complete description of game feature in table 3.2). Three output nodes represent the estimates $\hat{Q}^{Home}(s, a)$, $\hat{Q}^{Away}(s, a)$ and $\hat{Q}^{Neither}(s, a)$. Output values are normalized to probabilities. The $\hat{Q}$-functions for each team share weights. The network architecture is a Dynamic LSTM that takes as inputs a current sequence s_t, an action a_t and a dynamic trace length tl_t.

We apply an on-policy Temporal Difference (TD) prediction method **Sarsa** [93, Ch.6.4], to estimate $Q^{team}(s, a)$ for the NLH play dynamics observed in our dataset. The model parameters θ are optimized by mini-batch gradient descent via back-propagation. We used batch size 32 (determined experimentally). The Sarsa gradient descent update at time step t is based on a squared-error loss function:

$$\mathcal{L}_t(\theta_t) = \mathbb{E}[(g_t + \hat{Q}(s_{t+1}, a_{t+1}; \theta_t) - \hat{Q}(s_t, a_t; \theta_t))^2] \tag{4.2}$$

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \mathcal{L}(\theta_t) \tag{4.3}$$

where g and $\hat{Q}$ are for a single team[1]. LSTM training requires setting a **trace length** tl_t parameter. This key parameter controls how far back in time the LSTM propagates the error signal from the current time at the input history. Team sports like Ice Hockey show a turn-taking aspect where one team is on the offensive and the other defends; one such turn is called a **play**. We set tl_t to the number of time steps from current time t to the beginning of the current **play** (with a maximum of 10 steps). A play ends when the possession of puck changes from one team to another.

[1]In this dissertation, $Q(\cdot; \theta)$ indicates that the Q function is parameterized by θ.

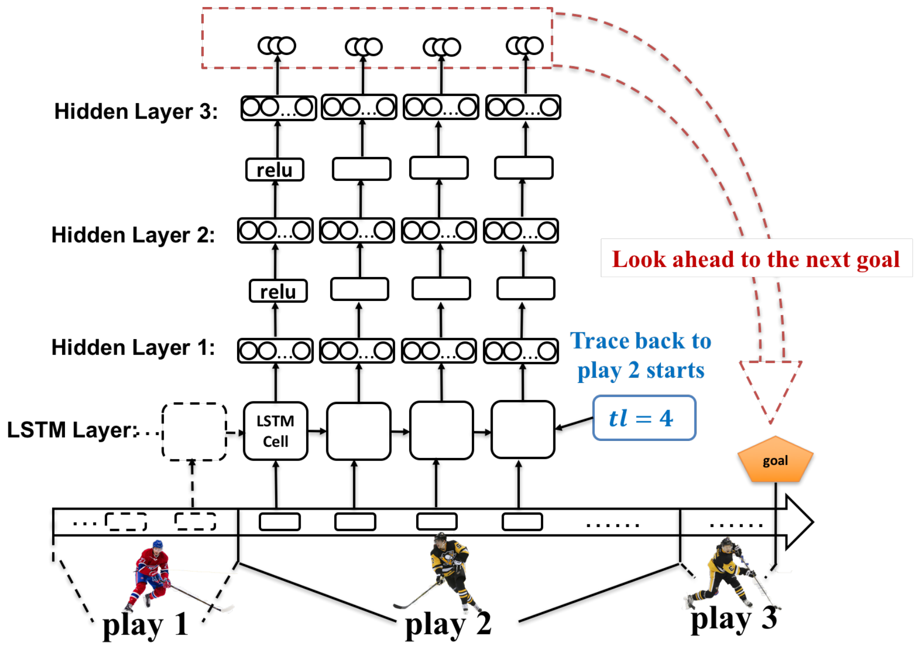


Figure 4.2: Our design is a 5-layer network with 3 hidden layers. Each hidden layer contains 1000 nodes, which utilize a relu activation function. The first hidden layer is the LSTM layer, the remaining layers are fully connected. Temporal-difference learning looks ahead to the next goal, and the LSTM memory traces back to the beginning of the play (the last possession change).

Using possession changes as break points for temporal models is common in several continuous-flow sports, especially basketball [20, 73]. We apply Tensorflow to implement training; our source code is published on-line.[2]

4.4.2 Illustration of Temporal Projection

Figure 4.3 shows a value ticker [30, 20] that represents the evolution of the Q function from the 3^{rd} period of a match between the blue Jackets (Home team) and the Penguins (Away team), Nov. 17, 2015. The figure plots values of the three output nodes. We highlight critical events and match contexts to show the context-sensitivity of the Q function. High scoring probabilities for one team decrease those of its opponent. The probability that neither team scores rises significantly at the end of the match.

4.4.3 Illustration of Spatial Projection

The neural network generalizes from observed sequences and actions to sequences and actions that have not occurred in our dataset. So we plot the learned smooth value surface $\hat{Q}^{Home}(s_\ell, shot(team))$ over the entire rink for home team shots in Figure 4.4. Here s_ℓ represents the average play his-

[2]https://github.com/Guiliang/DRL-ice-hockey

tory for a shot at location ℓ, which runs in unit steps from $x\_axis \in [-100, 100]$ and $y\_axis \in [-42.5, 42.5]$. It can be observed that 1) The chance that the home team scores after a shot is shown to depend on the angle and distance to the goal. 2) Action-value function generalizes to the regions where shots are rarely observed (At the lower or upper corner of the rink). The quasi-elliptical shape of the shot heatmap is plausible, capturing the importance of both distance and angle. The shape behind the net illustrates an interesting interpolation issue with the neural network model: While shots occur behind and very close to the net, almost no shots occur in the centre of the goal behind the net (Figure 4.5). In this case the neural network appears to interpolate the net centre value from the net corner values (Figure 4.4).

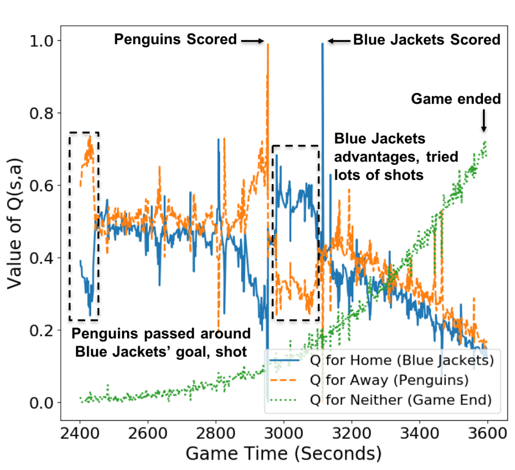


Figure 4.3: Temporal Projection of the method. For each team, and each game time, the graph shows the chance the that team scores the next goal, as estimated by the model. Major events lead to major changes in scoring chances, as annotated. The network also captures smaller changes associated with **every** action under different game contexts.

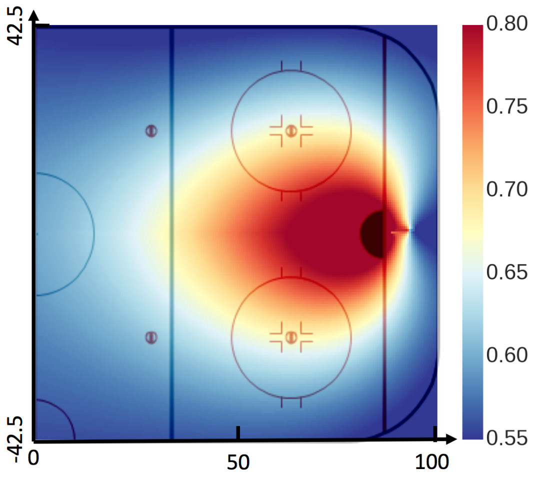


Figure 4.4: Spatial Projection for the shot action: The probability that the home team scores the next goal after taking a shot at a rink location, averaged over possible game states. Figure 4.5 shows the positions of shots behind the goal, which explains the distribution of Q values around the goal.

4.5 Player Evaluation Metric Based on Q values

In this section, we show how a player evaluation metric can be derived from the Q-function. Our approach to measuring player performance is assigning impact values (the difference between two consecutive Q values) to a player's action. To provide a theoretical foundation for our impact metric, this section introduces another Q-value-Above-Replacement metric to evaluate a player's action. By proving both metrics are equivalent, we show that Q-values unify the two main approaches to player evaluation.

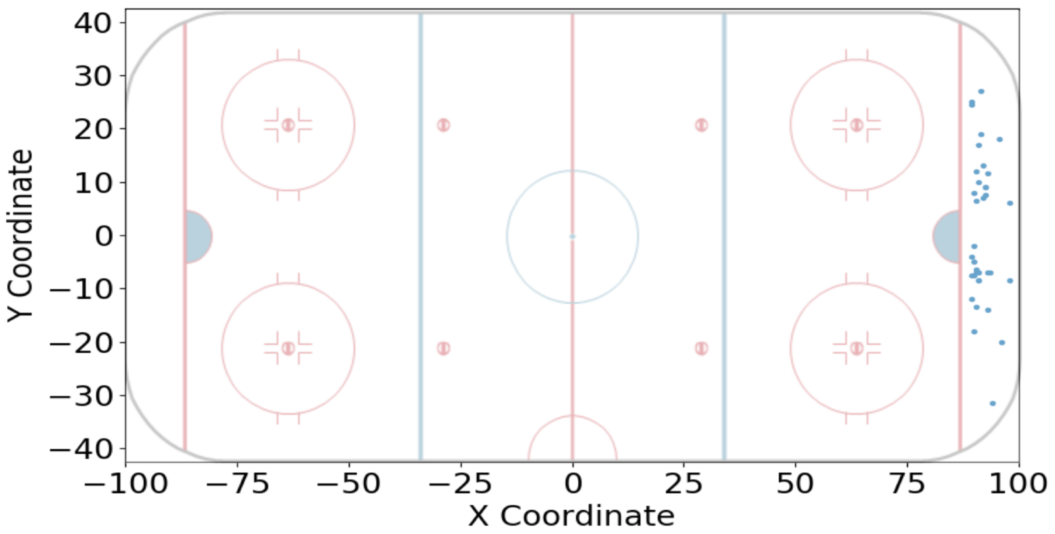


Figure 4.5: Shots behind the opponent's goal in the 2015-16 NHL game season.

4.5.1 Goal Impact: Deriving Action Values from Q-values.

Our Q-function concept provides a novel AI-based definition for assigning a value to an action. Similar to [85], we measure the quality of an action by how much it changes the expected total reward of a player's team: *the difference in expected total reward before and after the player acts*. The scoring chance at a time measures the value of a state, and therefore depends on the previous efforts of the entire team, whereas the change in value directly measures *the impact of an action by a specific player.* For our specific choice of Next Goal as the reward function, we refer to **goal impact**. The total impact of a player's actions is his **Goal Impact Metric** (GIM).

The following equations show how the action impact can be computed for a transition $\mathcal{T}\{s, a, r', s', a'\}$ given Q value estimates from our DP-LSTM model. The expected future total reward before s', a' is given by $r' + \mathbb{E}_{s',a'}[Q^{team}(s', a')|s, a]$ (Our Q function computes an expected value, check Equation 3.1). The expected future total reward after s', a' is given by $r' + Q^{team}(s', a')$. Therefore:

$$impact^{team_i}(s, a, s', a') \equiv Q^{team_i}(s', a') - \mathbb{E}_{s',a'}[Q^{team_i}(s', a')|s, a] \tag{4.4}$$

$$GIM^i(D) \equiv \sum_{s,a,s',a'} n[s, a, s', a', pl' = i; D] \cdot impact^{team_i}(s, a, s', a') \tag{4.5}$$

where D indicates our dataset, $team_i$ denotes the team of player i, and $n[s, a, s', a', pl' = i; D]$ is the number of times that player i performs action a' at s' after s, a occurs. The Bellman equation (2.1) implies that $\mathbb{E}_{s',a'}[Q^{team}(s', a')|s, a] = Q^{team}(s, a) - \mathbb{E}[r'|s, a]$. The expectation can therefore be computed from estimated Q values given an expected rewards model. In our data, scoring a goal is represented as a separate action $goal$, after which no transition occurs. This means that for every transition $\mathcal{T}\{s, a, r', s', a'\}$, we have that $a \neq goal$, $r' = 0$ and thus $\mathbb{E}[r'|s, a] = 0$. So in this representation, *the impact equation* (4.4) *reduces to the difference in Q values before and after the player acts.*

4.5.2 Q Value Above Average Replacement

We compare the goal impact metric to deriving a player metric from a Q-function using an above-average-replacement framework. The fact that the same player performance ranking can be derived using two fundamentally different approaches supports the conceptual foundations of our metric. The QAAR metric, compares the expected total future reward given that player i acts next, to the expected total future reward given that a random replacement player acts next:

$$QAAR^i(D) \equiv \sum_{s,a} n[s, a, pl' = i; D]\Big(\mathbb{E}_{s',a'}[Q^{team}(s', a'|s, a, pl' = i)] - \mathbb{E}_{s',a'}[Q^{team}(s', a')|s, a]\Big) \tag{4.6}$$

where $n[s, a, pl' = i; D]$ is the number of times that player i performs an action after s, a occurs. The QAAR metric can be computed for a dataset by using the maximum likelihood estimates of transition probabilities. QAAR and GIM are natural definitions for the value-above-replacement and action-value approaches, respectively. Our main result is that they are equivalent:

Proposition 1. *For each player i recorded in our play-by-play dataset D, his Q-value-above-replacement is equal to his goal impact metric: $QAAR^i D = GIM^i(D)$.*

The complete proof is in our Appendix. This equation indicates that by summing a player's impact over an entire game season (GIM), we measure how much his general playing skill exceeds that of an average player (replacement player with average Q-value) in the same league. Thus the same method for ranking players can be derived from a Q-function using two fundamentally different approaches. The QAAR metric also offers an alternative interpretation of GIM as the expected goals added by a player over a season. The next paragraph elaborates this interpretation. We also discuss why we do not standardize the GIM metric by the number of opportunities that a player has (e.g., dividing by their total Time-On-Ice).

GIM Interpretation in terms of Expected Goals Added. We introduce the following notation.

- Let $\delta_i[s, a] = \Big(\mathbb{E}_{s',a'}[Q^{team}(s', a'|s, a, pl' = i)] - \mathbb{E}_{s',a'}[Q^{team}(s', a')|s, a]\Big)$.
- Let $n_g[s, a, pl' = i; D]$ be the number of times that player i performs an action after s, a.
- Let N be the total number of games played by each team during the (regular) season. For example in the current NHL system, $N = 82$.

Then the QAAR metric (4.6) can be written as

$$GIM^i(D) = QAAR^i(D) = \sum_g \sum_{s,a} n_g[s, a, pl' = i; D] \times \delta_i[s, a] \tag{4.7}$$

The inner sum is *the expected number of goals added in game g if we replace player i by a league-average player*. Therefore dividing the GIM metric by N:

$$GIM^i(D)/N \tag{4.8}$$

yields the *average expected number of goals added* by player i over all games in a season. Therefore the GIM metric is related by a constant factor to a measure of how many goals a player adds, compared to a league-average player.

Scaled or standardized variants of a sum metric like GIM are commonly used in sports analytics. For example, instead of dividing GIM by the constant number of games N, we could consider dividing by the player's total number of games played, or by his total Time-on-Ice, or his total number of actions, or more generally a measure of how many opportunities the player had to earn credit for his actions. The drawback of standardization by opportunities is that *stronger players tend to have more opportunities* since their coaches field them more. For example, if we change in Equation (4.8) the constant N by the number N_i of games played by player i, this scaling will tend to penalize stronger players. Therefore the opportunity-scaled variant is less likely to single out strong players than the plain GIM total.

Another important point is that *the goals-added term $\delta_i[s, a]$ can be both positive and negative*. Therefore positive and negative contributions by a player can cancel each other out; to achieve an overall high GIM score, a player must most of the time make mainly positive contributions. This is very different from, say, adding up a player's shots: the more shots a player takes, the higher the count. Moreover, for a positive $\delta_i[s, a]$ contribution, a player's action must not only help his team, but it must have a greater positive impact than the average player. In professional sports, performing better than an average player is a high bar. In sum, standardizing by opportunities has the serious drawback of penalizing strong players, and the GIM metric is not intrinsically biased towards players who take more actions, because their actual contributions may be negative. Therefore we use the straightforward count of action impact values (Equation (4.5)), which can also be read as a count of expected goals added compared to a league-average player (Equation (4.6)). The next section turns to empirical evaluation and shows some ranking examples applying GIM to rate players.

4.5.3 Rank Players with GIM

Table 4.1 lists the top-20 highest impacts players, with basic statistics. All these players are well-known NHL stars. Taylor Hall tops the ranking although he did not score the most goals. This shows how our ranking, while correlated with goals, also *reflects the value of other actions by the player.* For instance, we find that the total number of passes performed by Taylor Hall is exceptionally high at 320. Our metric can be used to *identify undervalued players*. For instance, Johnny Gaudreau and Mark Scheifele drew salaries below what their GIM rank would suggest. Later they received a $\$5M+$ contract for the 2016-17 season.

Name	GIM	Assists	Goals	Points	Team	Salary
Taylor Hall	96.40	39	26	65	EDM	$6,000,000
Joe Pavelski	94.56	40	38	78	SJS	$6,000,000
Johnny Gaudreau	94.51	48	30	78	CGY	$925,000
Anze Kopitar	94.10	49	25	74	LAK	$7,700,000
Erik Karlsson	92.41	66	16	82	OTT	$7,000,000
Patrice Bergeron	92.06	36	32	68	BOS	$8,750,000
Mark Scheifele	90.67	32	29	61	WPG	$832,500
Sidney Crosby	90.21	49	36	85	PIT	$12,000,000
Claude Giroux	89.64	45	22	67	PHI	$9,000,000
Dustin Byfuglien	89.46	34	19	53	WPG	$6,000,000
Jamie Benn	88.38	48	41	89	DAL	$5,750,000
Patrick Kane	87.81	60	46	106	CHI	$13,800,000
Mark Stone	86.42	38	23	61	OTT	$2,250,000
Blake Wheeler	85.83	52	26	78	WPG	$5,800,000
Tyler Toffoli	83.25	27	31	58	DAL	$2,600,000
Charlie Coyle	81.50	21	21	42	MIN	$1,900,000
Tyson Barrie	81.46	36	13	49	COL	$3,200,000
Jonathan Toews	80.92	30	28	58	CHI	$13,800,000
Sean Monahan	80.92	36	27	63	CGY	$925,000
Vladimir Tarasenko	80.68	34	40	74	STL	$8,000,000

Table 4.1: 2015-2016 Top-20 Player Impact Scores

4.6 Empirical Evaluation

We describe our comparison methods and evaluation methodology. Similar to clustering problems, there is **no ground truth** for the task of player evaluation. To assess a player evaluation metric, we follow previous work [80, 75] and compute its correlation with statistics that directly measure success like Goals, Assists, Points, Play Time (Section 4.6.2). There are two justifications for comparing with **success measures**. (1) These statistics are generally recognized as important measures of a player's strength, because they indicate the player's ability to contribute to game-changing events. So a comprehensive performance metric ought to be related to them. (2) The success measures are often forecasting targets for hockey stakeholders, so a good player evaluation metric should have predictive value for them. For example, teams would want to know how many points an offensive player will contribute. To evaluate the ability of the GIM metric for generalizing from past performance to future success, we report two measurements: How well the GIM metric predicts a total season success measure from a sample of matches only (Section 4.6.3), and how well the GIM metric predicts the future salary of a player in subsequent seasons (Section 4.6.4). Mapping performance to salaries is a practically important task because it provides an objective standard to guide players and teams in salary negotiations [47].

4.6.1 Comparison Player Evaluation Metrics

We compare GIM with the following player evaluation metrics to show the advantage of 1) modeling game context 2) incorporating continuous context signal 3) including history.

Our first baseline method **Plus-Minus (+/-)** is a commonly used metric that measures how the presence of a player influences the goals of his team [67]. The second baseline method **Goal-Above-Replacement (GAR)** estimates the difference of team's scoring chances when the target player plays, vs. replacing him or her with an average player [34]. **Win-Above-Replacement (WAR)**, our third baseline method, is the same as GAR but for winning chances [34]. Our fourth baseline method **Expected Goal (EG)** weights each shot by the chance of it leading to a goal. These four methods consider only very limited game context. The last baseline method **Scoring Impact (SI)** is the most similar method to GIM based on Q-values. But Q-values are learned with pre-discretized spatial regions and game time [84]. As a lesion method, we include **GIM-T1**, where we set the maximum trace length of LSTM to 1 (instead of 10) in computing GIM. This comparison assesses the importance of including enough history information.

Computing Cost. Compared to traditional metrics like +/-, learning a Q-function is computationally demanding (over 5 million gradient descent steps on our dataset). However, after the model has been trained off-line, the GIM metric can be computed quickly with a single pass over the data.

Training Settings. We describe the player evaluation as a descriptive task rather than a predictive generalization problem. Our training dataset contains all data in the play-by-play ice hockey dataset. For the season total (section 4.6.2), the round-by-round correlation experiment (section 4.6.3), and the future seasons salary experiment (section 4.6.4), we apply the 2015-2016 season ice-hockey dataset. For the cross-season correlation experiment (section 4.6.5), we apply the 2015-2019 season ice-hockey dataset.

4.6.2 Season Totals: Correlations with standard Success Measures

In the following experiment, we compute the correlation between player ranking metrics and success measures over the entire season. Table 4.2 shows the correlation coefficients of the comparison methods with 14 standard success measures: Assist, Goal, Game Wining Goal (GWG), Overtime Goal (OTG), Short-handed Goal (SHG), Power-play Goal (PPG), Shots (S), Point, Short-handed Point (SHP), Power-play Point (PPP), Face-off Win Percentage (FOW), Points Per Game (P/GP), Time On Ice (TOI) and Penalty Minute (PIM). These are all commonly used measures available from the NHL official website (www.nhl.com/stats/player). *GIM achieves the highest correlation* *in 12 out of 14 success measures.* For the remaining two (TOI and PIM), GIM is comparable to the highest. Together, the Q-based metrics GIM, GIM-1 and SI show the highest correlations with success measures. EG is only the fourth best metric, because it considers only the expected value of shots without look-ahead. The traditional sports analytics metrics correlate poorly with almost all success measures. This is evidence that AI techniques that provide fine-grained expected action-value estimates lead to better performance metrics. With the neural network model, GIM can handle continuous input without pre-discretization. This prevents the loss of game context information and explains why both GIM and GIM-T1 performs better than SI in most success measures. And the higher correlation of GIM compared to GIM-T1 also demonstrates the value of game history. In terms of absolute correlations, GIM achieves high values, except for the very rare events OTG, SHG, SHP and FOW. Another exception is Penalty Minutes (PIM), which interestingly, show positive correlation with all player evaluation metrics, although penalties are undesirable. We hypothesize that better players are more likely to receive penalties, because they play more often and more aggressively.

methods	Assist	Goal	GWG	OTG	SHG	PPG	S
+/-	0.236	0.204	0.217	0.16	0.095	0.099	0.118
GAR	0.527	0.633	0.552	0.324	0.191	0.583	0.549
WAR	0.516	0.652	0.551	0.332	0.192	0.564	0.532
EG	0.783	0.834	0.704	0.448	0.249	0.684	0.891
SI	0.869	0.745	0.631	0.411	0.27	0.591	0.898
GIM-T1	0.873	0.752	0.682	0.428	0.291	0.607	0.877
GIM	**0.875**	**0.878**	**0.751**	**0.465**	**0.345**	**0.71**	**0.912**

methods	Point	SHP	PPP	FOW	P/GP	TOI	PIM
+/-	0.237	0.159	0.089	-0.045	0.238	0.141	0.049
GAR	0.622	0.226	0.532	0.16	0.616	0.323	0.089
WAR	0.612	0.235	0.531	0.153	0.605	0.331	0.078
EG	0.854	0.287	0.729	0.28	0.702	0.722	0.354
SI	0.869	0.37	0.707	0.185	0.655	**0.955**	**0.492**
GIM-T1	0.902	0.384	0.736	0.288	0.738	0.777	0.347
GIM	**0.93**	**0.399**	**0.774**	**0.295**	**0.749**	0.835	0.405

Table 4.2: Correlation with standard success measures.

4.6.3 Round-by-Round Correlations: Predicting Future Performance From Past Performance

A sports season is commonly divided into **rounds**. In round n, a team has finished n games in a season, for example, the round 1 finishes when all the team in the NHL league has finished the first game in their game season. For a given performance metric, we measure the correlation between (i) its value computed **over the first** n **rounds**, and (ii) the value of the three main success measures, assists, goals, and points, computed **over the entire season**. This allows us to assess how quickly different metrics acquire predictive power for the final season total, so that future performance can be predicted from past performance. We also evaluate the **auto-correlation** of a metric's round-by-round total with its own season total. The auto-correlation is a measure of temporal consistency, which is a desirable feature [75], because generally the skill of a player does not change greatly throughout a season. Therefore a good performance metric should show temporal consistency.

We focused on the expected value metrics EG, SI, GIM-T1 and GIM, which had the highest correlations with success in Table 4.2. Figure 4.6 shows metrics' round-by-round correlation coefficients with assists, goals, and points. The bottom right shows the auto-correlation of a metric's round-by-round total with its own season total. *GIM is the most stable metric* as measured by auto-correlation: after half the season, the correlation between the round-by-round GIM and the final GIM is already above 0.9.

We find both GIM and GIM-T1 eventually dominate the predictive value of the other metrics, which shows the advantages of modeling sports game context without pre-discretization. And possession-based GIM also dominates GIM-T1 after the first season half, which shows the value of including play history in the game context. But how quickly and how much the GIM metrics improve depends on the specific success measure. For instance, in Figure 5.6, GIM's round-by-round correlation with Goal (top right graph) dominates by round 10, while others require a longer time.

4.6.4 Future Seasons: Predicting Players' Salary

In professional sports, a team will give a comprehensive evaluation to players before deciding their contract. The more value players provide, the larger contract they will get. Accordingly, a good performance metric should be positively related to the amount of players' **future** contract. The NHL regulates when players can renegotiate their contracts, so we focus on players receiving a new contract following the games in our dataset (2015-2016 season).

methods	2016 to 2017 Season	2017 to 2018 Season
Plus Minus	0.177	0.225
GAR	0.328	0.372
WAR	0.328	0.372
EG	0.587	0.6
SI	0.609	0.668
GIM-T1	0.596	0.69
GIM	**0.666**	**0.763**

Table 4.3: Correlation with Players' Contract

Table 4.3 shows the metrics' correlations with the amount of players' contract over all the players who obtained a new contract during the 2016-17 and 2017-18 NHL seasons. Our GIM score achieves the highest correlation in both seasons. This means that the metric can serve as an objective basis for contract negotiations. The scatter plots of Figure 4.7 illustrate GIM's correlation with amount of players' future contract. In the 2016-17 season (left), we find many underestimated play-

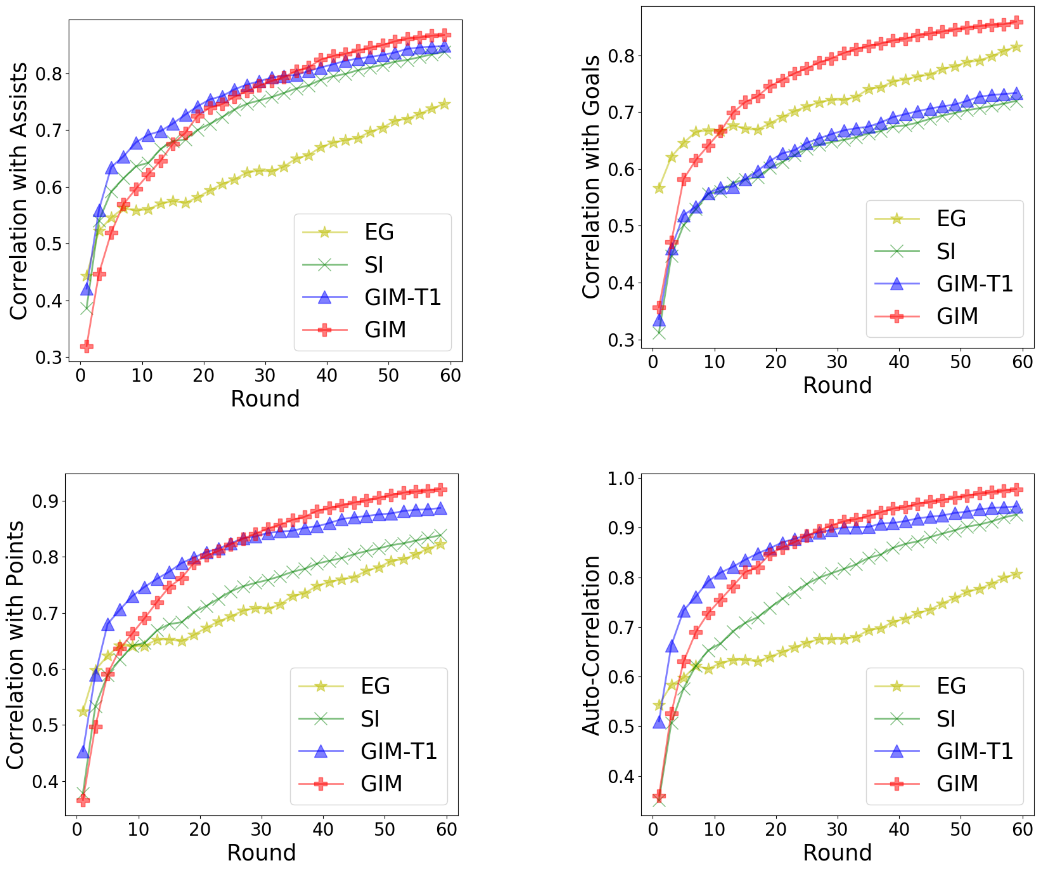


Figure 4.6: Correlations between round-by-round metrics and season totals.

ers in the right bottom part, with high GIM but low salary in their new contract. It is interesting that the percentage of players who are undervalued in their new contract decreases in the next season (from $32/258$ in 2016-17 season to $8/125$ in 2017-2018 season). This suggests that GIM provides an early signal of a player's value after one season, while it often takes teams an additional season to recognize performance enough to award a higher salary.

4.6.5 Cross-Season Correlations: Measuring the consistency of GIM between seasons.

A player performance is generally consistent between two continuous seasons, consequently, a promising player evaluation metrics should assign similar values to the same player in two season. Figure 4.8, Figure 4.9 and Figure 4.10 illustrate the scatter plots of the players' GIMs between two continuous seasons. Table 4.4 shows the corresponding correlation coefficient. The computation omits the players who quit the NHL league or have not attend the league during the any of the two continuous seasons The correlation of GIMs between two continuous season is positive, which proves the consistency of GIM.

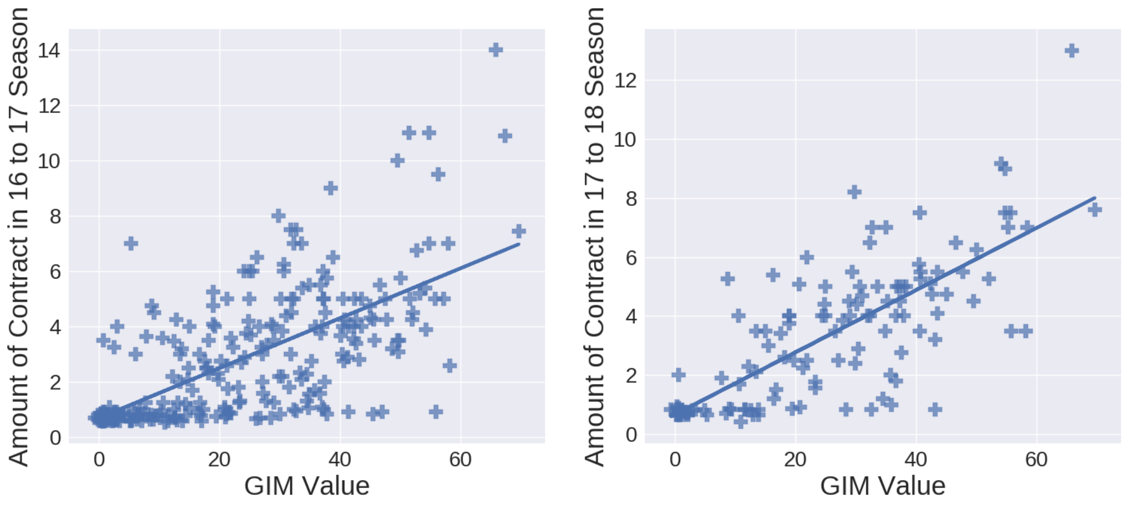


Figure 4.7: Player GIM vs. Value of new contracts in the 2016-17 (left) and 2017-18 (right) NHL season.

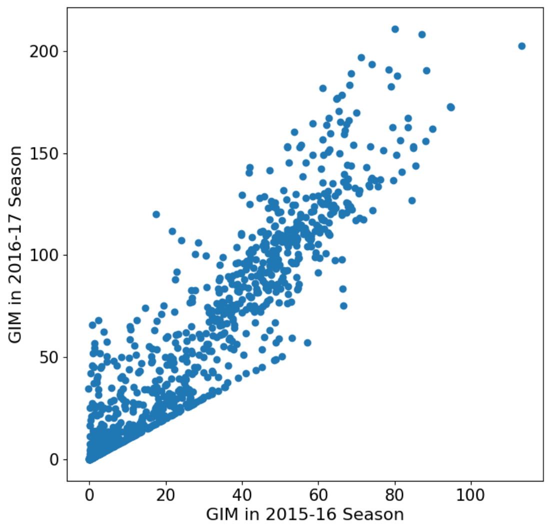


Figure 4.8: The scatter plot of players' GIMs between 15-16 and 16-17 NHL season.

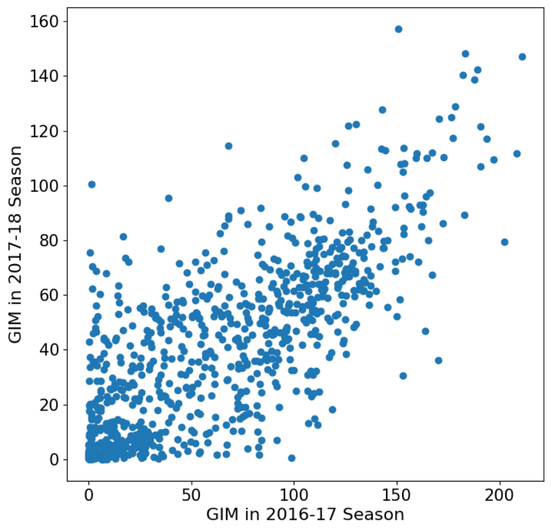


Figure 4.9: The scatter plot of players' GIMs between 16-17 and 17-18 NHL season.

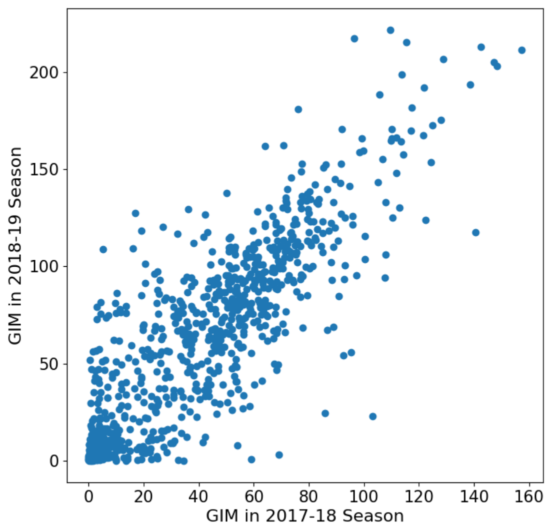


Figure 4.10: The scatter plot of players' GIMs between 17-18 and 18-19 NHL season.

NHL Season	Correlations
2015-16 v.s. 2016-17	0.9281
2016-17 v.s. 2017-18	0.7609
2017-18 v.s. 2018-19	0.8409

Table 4.4: The pearson correlation coefficient of the players' GIMs between two continuous seasons.

4.7 Summary

We applied DRL to learn complex spatio-temporal NHL dynamics. The trained neural network provides a rich source of knowledge about how a team's chance of scoring the next goal depends on the match context. Based on the learned action values, we developed an innovative context-aware performance metric GIM that provides a comprehensive evaluation of NHL players, taking into account **all** of their actions. In our experiments, GIM had the highest correlation with most standard success measures, was the most temporally consistent metric, and generalized best to players' future salary. Our approach applies to similar continuous-flow sports games with rich game contexts, like soccer and basketball.

Chapter 5

Extending the Action-Value Function for Evaluating Soccer Players

5.1 Introduction: Valuing Actions of Soccer Players

Soccer is arguably the most challenging to analyze of all the major team sports [13]. The game context of soccer is much more complicated than that of ice hockey, given that soccer has more players (22 players), larger pitch (350 feet long and 150 feet wide) and longer playing time (90 minutes), all of which lead to complex spatio-temporal distribution patterns for each team. In this chapter, we apply *Deep Reinforcement Learning* (DRL) to learn an action-value Q-function from events in a soccer game. We introduce a stacked two-tower LSTM to capture the playing dynamics for home and away teams separately. Unlike the traditional control problem in reinforcement learning aiming to learn the optimal policy, we solve the prediction problem in the passive learning (on policy) setting.

Based on the learned Q-function, we use the Goal Impact Metric (GIM, see Equation (4.5)) to evaluate player performance. GIM ranks a player by aggregating the impact of all his actions, where the **impact** of an action is the change of consecutive Q values due to this action. In empirical comparison with four comparison metrics, GIM shows the highest correlation with most standard success measurements. Generalizing from an initial sample of season matches, GIM is the best predictor of season total goals and assists.

To compute the action values for all players, we build a large dataset consisting of over 4.5M action events by pooling data from several soccer leagues. This dataset allows the model to learn general estimations for action values. However, as the game context within a specific league may differ from that of the general soccer game, the contribution of a player varies among leagues. To address the trade-off between generalizing across leagues and specializing to a specific one, we propose a *fine-tuning* approach: begin with the general model as an initialization, then train on the specific data from a certain league. Given the English Football League (EFL) Championship data, we use fine-tuning to improve the model's fitting performance as well as the evaluation results for players in this league.

5.2 Learning Q Values: Model Architecture and Training

This section introduces a neural network architecture and the weight training methods to learn a Q-function ($Q_{team}(s, a)$, defined in Equation (4.1)). Due to the complexity of soccer, the architecture is more complex than the one we used for ice hockey.

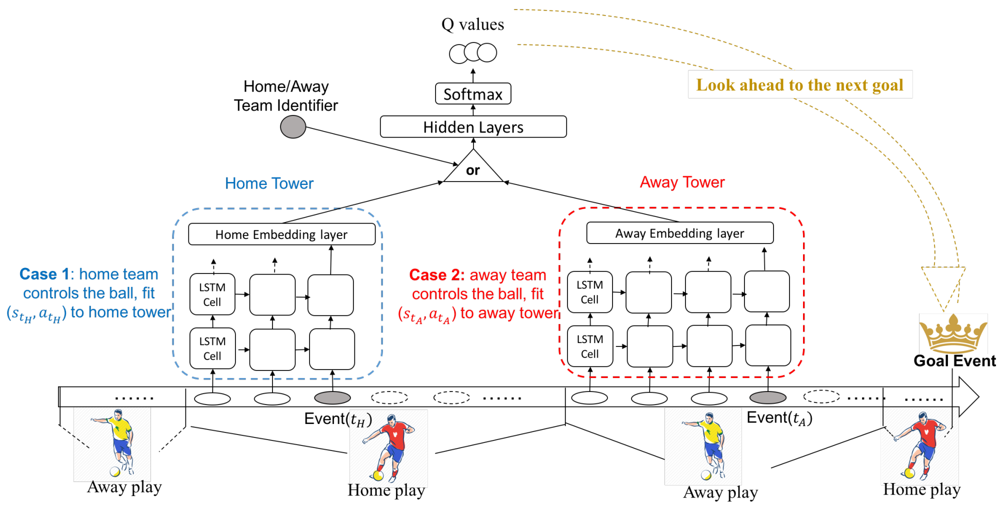
5.2.1 Model Architecture: Function Approximation with Neural Network

We discuss the model architecture for learning the Q values. Given a **discrete** state space, it is possible to use dynamic programming for computing Q-values [85]. But our soccer model contains continuous observation features derived from continuous time stamps and spatial locations. A common solution is to discretize spatio-temporal indices [37]. However, the resulting discontinuities undermine the precision of state values and influence the model performance. In this chapter, we develop a neural network approach that can directly incorporate continuous observation features.

To generate Q-values, our model applies the two-tower design to fit the data of home/away teams separately and a recurrent neural network to capture the sequential features in play history. Figure 5.1 shows our model structure. The model fits home and away data separately with a two-tower structure [90], because from domain knowledge (e.g. the home team advantage [95]) we expect the Q values to be different depending on whether a team plays at home or away. Each tower captures the play history with a stacked LSTM, which is a multi-layer LSTM, where outputs of LSTM cells in lower layers are used as the input for higher layers. Compared to the single layer LSTM, stacking adds levels of abstraction for sequences' input features. This increases the model's ability to generalize across complex game contexts. The complete play history of game contexts and actions (s_t, a_t) is summarized in the last hidden state of the top LSTM layer. Our model uses a team identifier unit to select the hidden state from the home or the away tower according to who controls the ball in the current play. The selected hidden state values are sent to hidden layers whose outputs are normalized by a softmax function and considered as our estimates of $\hat{Q}^{Home}(s, a)$, $\hat{Q}^{Away}(s, a)$, and $\hat{Q}^{Neither}(s, a)$.

5.2.2 Weight Training

We train the two-tower neural network with an on-policy Temporal Difference (TD) prediction method **Sarsa** [94, Ch.6.4] and apply a dynamic-possession LSTM to control the trace length during training. Our goal is to learn a function that estimates $Q_{team}(s, a)$ for the play dynamics observed in our dataset, with which we evaluate the performance of players instead of controlling them (which is hard in the case of professional players). The training details are discussed below.



Q values
Softmax
Hidden Layers
Home/Away Team Identifier
or
Look ahead to the next goal
Home Tower
Away Tower
Home Embedding layer
Away Embedding layer
LSTM Cell
Case 1: home team controls the ball, fit (s_{t_H}, a_{t_H}) to home tower
Case 2: away team controls the ball, fit (s_{t_A}, a_{t_A}) to away tower
Goal Event
Event(t_H)
Event(t_A)
Away play
Home play
Away play
Home play

Figure 5.1: The architecture of our Two-Tower Dynamic Play LSTM (TTDP-LSTM). The figure shows how the model processes two generic time instances, one where the home team acts—which is analyzed by the home tower—and another where the away team acts—analyzed by the away tower.

Home/Away Tower Weight Training. At training step t, our model sends the output from the home/away tower to the hidden layers if the home/away team controls the ball at t. During one training step, the hidden layers estimate the Q values for two continuous actions and states within one transition $\mathcal{T}\{s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}\}$. The estimated Q values are applied to compute the TD loss:

$$\mathcal{L}(\theta) = \sum_{team \in T} \mathbb{E}\left[(r_{team,t+1} + \hat{Q}^{team}(s_{t+1}, a_{t+1}; \theta) - \hat{Q}^{team}(s_t, a_t; \theta))^2\right] \tag{5.1}$$

Given this loss function, we optimize the weights of our neural model (Figure 5.1) by mini-batch gradient descent via backpropagation. As for each transition, an error signal is sent only to either the home or the away tower, the flow of gradients will only influence one of the two towers and thus their weights are updated independently. This independence separates home and away signals and helps the network to learn their impact (e.g., home advantage [95]).

Dynamic Possession-LSTM. Team sports like soccer show a turn-taking aspect where one team is on the offensive and the other defends; one such turn is called a **play**. A play ends when possession passes from the team at time t to the opposing team at time $t+1$ [63]. In a sports game, events within a play are highly correlated, but when a team loses control of the ball (meaning the play ends), the attacking team switches to defense. The dependence between actions from successive plays is therefore much weaker. The turn-taking aspect inspires a natural way of determining the trace length tl_t, which controls how far back in time the LSTM propagates the error signal from the current time at the input history. Instead of fixing the trace length, our model dynamically computes it and sets tl_t to the number of time steps from current time t to the beginning of the current **play** (with a maximum of 10 steps), so that LSTM can restrict the history traces to the continuous possession of one team. Using possession changes to define episodes for temporal models has been proven to be successful in many continuous-flow sports, especially basketball [21, 37].

Training Settings. For our TTDP-LSTM model in Figure 5.1, both home and away towers apply a two-layer LSTM, whose signals are sent to two hidden layers with three output nodes. The number of nodes in LSTM hidden states and hidden layers are both 256. The max trace length of LSTM is 10 [41]. During training, we minimize the loss function $\mathcal{L}(\theta)$ with Adam optimizer that applies a general learning rate of 10E-04 on the entire dataset containing over 4.5M event data and a fine-tuning learning rate of 10E-05 on the league-specific dataset.

5.3 Model Validation: Q Values

Our case studies illustrate the learned Q-function with temporal and spatial projections. To validate the model performance, we show that the learned Q values are well-calibrated, meaning that they offer a satisfactory fit to empirical scoring frequencies observed under different game contexts.

5.3.1 Illustration of Temporal and Spatial Projection.

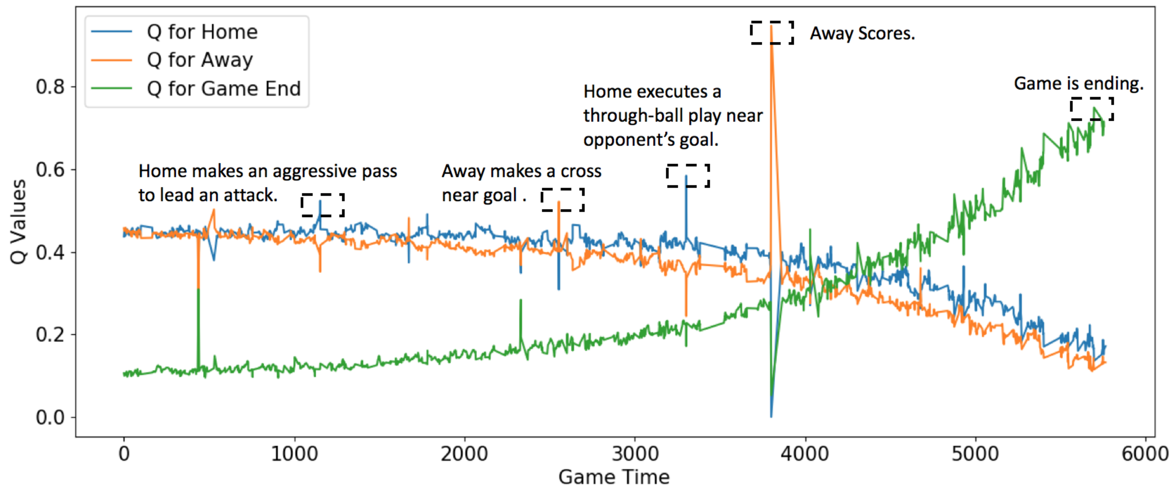


Figure 5.2: Temporal Projection of the learned Q-function. The game is between Fulham (Home) and Sheffield Wednesday (Away), which has happened on Aug.19th, 2017.

Temporal Projection. We illustrate the estimated Q values for actions and states across game times. Figure 5.2 shows a **value ticker** [21] that represents the evolution of the Q values during a randomly sampled game from our dataset. The figure plots values of the three output nodes representing $\hat{Q}^{Home}(s,a)$, $\hat{Q}^{Away}(s,a)$, and $\hat{Q}^{Neither}(s,a)$, according to which we highlight critical events to show the context-sensitivity of the Q-function. We observe that: 1) High scoring probabilities for one team decrease those of its opponent. 2) The probability that neither team scores rises significantly at the end of the match.

Spatial Projection. To study the influence of players' positions on scoring probability, we generate Q values for the entire soccer pitch. Our neural model can generalize from observed states and actions to those that have not occurred in the observed game season. Our model's generalization ability allows us to estimate a Q value for any action performed at any position.

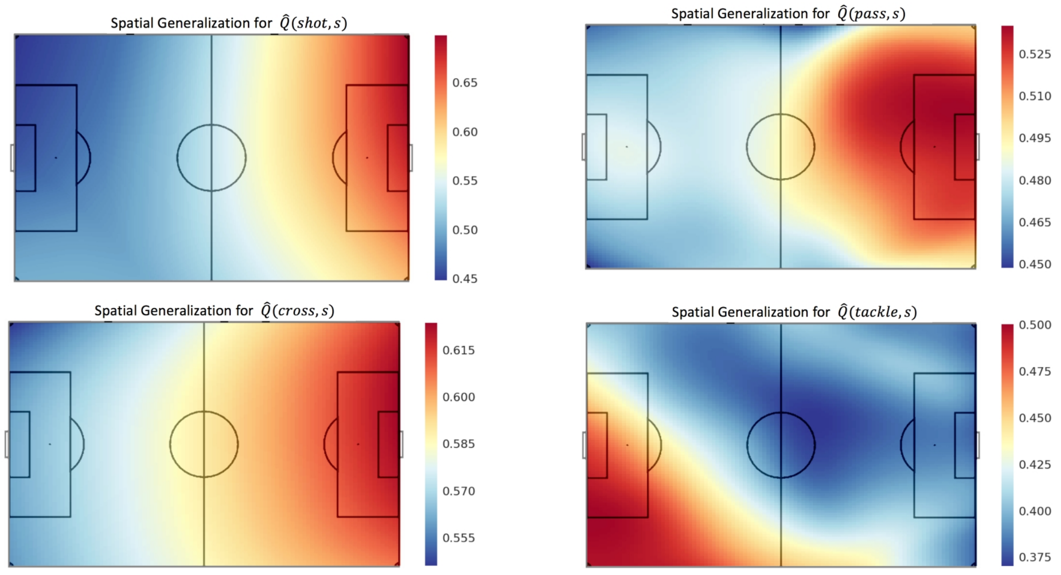


Figure 5.3: Spatial Projections for estimated Q values: $\hat{Q}^{Home}(s, shot)$, $\hat{Q}^{Home}(s, pass)$, $\hat{Q}^{Home}(s, cross)$ and $\hat{Q}^{Home}(s, tackle)$ over the entire soccer pitch. We use the adjusted coordinate described in Section 3.3.1.

Figure 5.3 shows the learned smooth Q-function surface $\hat{Q}^{Home}(s,a)$ over possible game trajectories for several actions of home team including shot, pass, cross, and tackle. We select those actions because they are frequent and have been studied in many previous works [17, 100]. Among the selected actions, we observe the followiong. (1) The Q value of a shot increases with closeness to the opponent's goal. (2) Angles from the left side of the goal appear slightly more promising than from the right. The plots for $\hat{Q}^{home}(s, pass)$ and $\hat{Q}^{home}(s, cross)$ show the same phenomena. An explanation for the first observation is that players have more chance to score when they approach their opponent's goal. The second observation reflects the data: The numbers of successful shots on the upper and the lower soccer pitch are 3,769 v.s. 3,544. The difference of these scoring frequencies verifies the asymmetry captured by the neural network model. Figure 5.4 shows two goal-scoring banana kicks (marked by a red circle) made on the upper right corner while no such shots on the lower corner are found in our data. Therefore the right/left foot asymmetry in player's shooting skills partially explains the asymmetrical scoring chance.

The asymmetrical scoring chance on a soccer pitch also explains why the defensive action tackle made near the bottom left corner is more valuable (the last plot): tackles disturb opponents' actions that might lead to successful shots on their upper corner. Figure 5.5 shows the positions of tackle. A successful tackle near the opponent's goal will create a promising scoring opportunity. However, most players perform tackles in the defensive zone near their own goal, and thus our DRL model does not capture the value of tackles in the offensive zone well (due to the lack of data). To help evaluate the model with respect to defensive tackles, Table 5.1 show the frequency of a team scoring the next goal after performing tackle on the top or bottom half a pitch, which allows calculating that p(Next Goal|top)=0.3373 and p(Next Goal|bottom)=0.3407. While the model captures the advantage of bottom pitch tackles over top pitch ones, it seems to overestimate the magnitude of this effect.

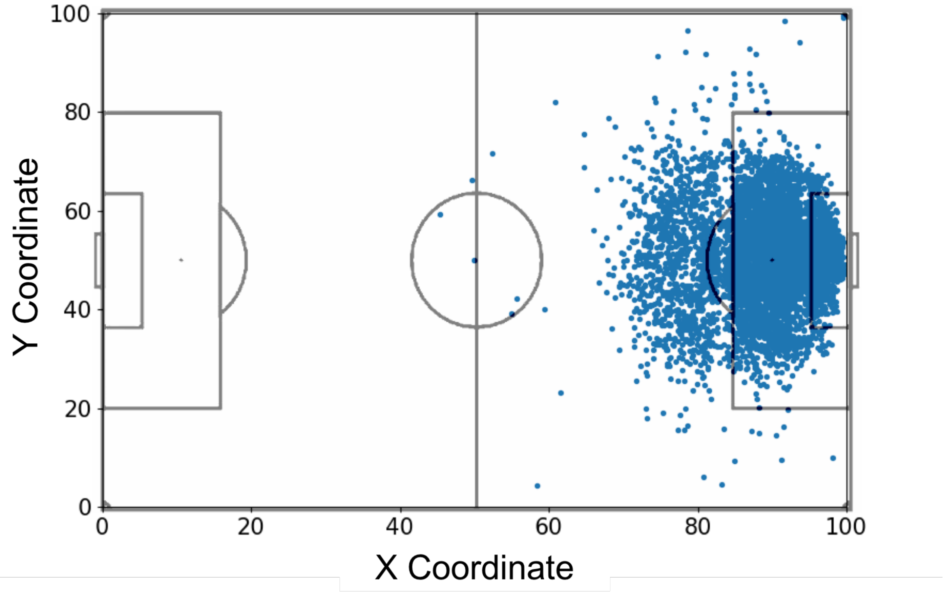


Figure 5.4: Spatial illustration of all goal-scoring shots under the adjusted coordinates (with our dataset).

Scoring v.s., Positions		Score Next Goal	
		Yes	No
Tackle	Top	17267	33923
Position	Bottom	17930	34698

Table 5.1: Calibration results for the spatial illustration of tackle values.

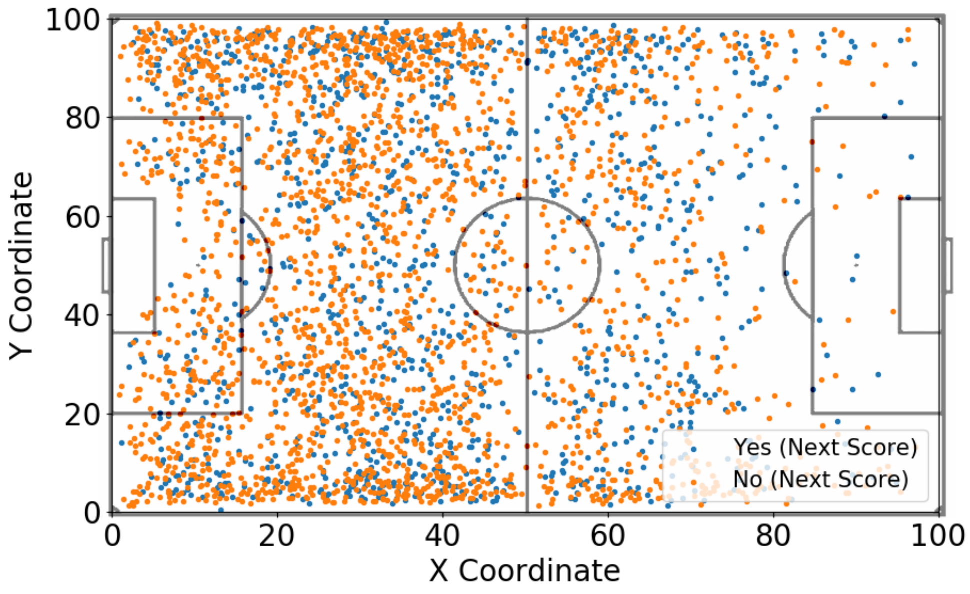


Figure 5.5: Spatial illustration of tackle under the adjusted coordinates (with our dataset) in 100 randomly sampled game. We label these tackles by whether the team that performs the tackle manages to score the next goal.

5.3.2 Calibration Quality for the learned Q-function

The calibration measurement evaluates how well our learned Q-function fits the observed next-goal scoring frequencies under different game contexts. Our approach to defining the game context is dividing the continuous state space into discrete bins representing a type of game context. To calculate empirical visiting frequencies of bins, we assign an observed state to a bin according to the values of three discrete **context features** in the last observation: Manpower (Short Handed (SH), Even Strength (ES), Power Play (PP)), Goal Differential (≤ -3, -2, -1, 0, 1, 2, $\geq$ 3) and Period (1 (first half), 2 (second half)). The total number of bins is $3 \times 7 \times 2 = 42$. This partition has two advantages. 1) The context features are well-studied and important for soccer experts [29], so the model predictions can be checked against domain knowledge. 2) The partition covers a wide range of match contexts, and each bin aggregates a large set of play histories. If our model exhibits a systematic bias, the aggregation should amplify it and become detectable.

Given the set of bins where each bin A contains a total of $|A|$ states, the empirical and estimated scoring probabilities for each bin are defined as follows:

- *Empirical Scoring Probabilities* : for each observed state s, we set $g^{team}_{obs}(s) = 1$ if the observed episode containing state s ends with a goal by team $team = Home$, $Away$ or neither ($team = Neither$). Then $Q^{team}_{obs}(A) = \frac{1}{|A|} \sum_{s \in A} g^{team}_{obs}(s)$

Man.	Goal.	P.	$\|A\|$	TT_Home	TT_Away	TT_MAE	Markov_MAE
ES	-1	1	73176	0.4374	0.4159	0.0052	0.1879
ES	-1	2	96408	0.3496	0.3025	0.0782	0.1783
ES	0	1	356597	0.4437	0.4272	0.026	0.1908
ES	0	2	160080	0.356	0.3077	0.0814	0.1792
ES	1	1	88726	0.4402	0.4128	0.0335	0.1899
ES	1	2	119901	0.3459	0.295	0.077	0.1787
PP	-1	1	876	0.4366	0.4045	0.1752	0.1937
PP	-1	2	3319	0.352	0.2911	0.0668	0.1685
PP	0	1	3183	0.4414	0.403	0.1308	0.187
PP	0	2	7183	0.3579	0.2855	0.0841	0.1804
PP	1	1	1316	0.4391	0.3949	0.115	0.1825
PP	1	2	7676	0.356	0.2862	0.1121	0.1792

Table 5.2: Calibration Results. TT_Home and TT_Away report the average scoring probability $\hat{Q}^{team}(A)$ estimated by our TTDP-LSTM model. Here we compare only Q values for pass and shot as they are frequent and well-studied actions. TT_MAE is the Mean Absolute Error (MAE) between estimated scoring probabilities from our model and empirical scoring probabilities. For comparison, we also report a Markov_MAE which applies the estimates from a discrete-state Markov model [85].

- *Estimated Scoring Probabilities*: we apply our TTDP-LSTM model to estimate a Q value for each observed sequence and average the resulting estimates to compute the estimated scoring probabilities : $\hat{Q}^{team}(A) = \frac{1}{|A|}\sum_{s \in A} \hat{Q}^{team}(s, a)$

We evaluate the fit as the difference between the average empirical scoring probability $Q^{team}_{obs}(A)$ and the average estimated scoring probability $\hat{Q}^{team}(A)$. We show the results in Table 5.2 where the context features (Man. Goal. and P.) define a bin, and $|A|$ records the frequency of bin A in our dataset. The estimated Q-function matches several well-known phenomena: 1) The chance of either team scoring another goal decreases in the second period. 2) A clear **home team advantage** [95]: Comparing two match contexts with the home and away team roles exchanged, the relative advantage of the home team is greater than that of the away team. 3) Manpower advantage by the home team means a lower scoring chance for the away team.

Our conclusions are as follows. 1) The model fit is satisfactory (The average MAE for all bins is below 0.1), except for some relatively rare game contexts (e.g. the context whose corresponding bin count is only 876, which indicates this context appears only 876 times out of 3M match states). 2) Our model significantly outperforms the Markov Model. This shows the value of a function approximation model that can utilize continuous space-time information without losing information due to discretization.

5.4 Player Ranking: Case Study

Applying the GIM metric (Equation 4.5) defined in the previous chapter, we discuss the ranking results for several players. We rank the EFL Championship players by their GIMs over the entire 2017-2018 game season. Our case study ranks only players in one league because they face the same level of competition and therefore their contributions are comparable. We chose the EFL Championship, which is just below the Premier League in the league hierarchy, because it has a large number of players in our data set and it has been much less studied than the Premier League.

Fine-Tuning. Different leagues have their own characteristics including competition level, season length, and playoff agenda. Therefore we apply a fine-tuning technique to achieve a better fit to the EFL Championship games: 1) First, train a general model to evaluate actions in European soccer with our dataset containing games from multiple European Soccer leagues. 2) Fine-tune the initial weight values from the general model, with a smaller learning rate and using only EFL Championship game data. Fine-tuning refines the general model and improves its ability to fit the behaviour of players. Compared to training the model from scratch, fine-tuning the general model significantly reduces training time and prevents over-fitting. In the following assessment, we describe GIM values computed with the fine-tuned model and present both a general ranking for all actions and action-specific rankings.

5.4.1 All-Actions Assessment

Table 5.3 lists the 10 players with highest GIM for all actions. Our ranking includes the players with the most goals and assists. We investigate the positive correlation between our metric and standard success measures further in the next section. Matej Vydra tops our 2017-2018 season ranking. He dominated the scoring board of the England Championship league and won the 2017-18 Golden Boot award[1]. In the next season (2018-2019), the Premier League team Burnley recognized the talent of Vydra and signed him on a three-year deal from team Derby after the 2017-18 game season. Another example is Tom Cairney, who has only 5 goals and 5 assists over the entire season but ranks 6th in GIM assessment. Although he does not lead any standard success measures (Goals, Assists), his impact was an indispensable factor of his team's success in winning the 2017-18 EFL playoffs. For example, he scored the only goal of the final in which Fulham beat Aston Villa by 1-0 in the Wembley stadium and earned promotion to the Premier league. As the team captain, Tom Cairney was nominated as the EFL's Championship Player of the Season award[2].

[1]https://www.skysports.com/football/news/11688/11361634/

[2]https://www.bbc.com/sport/football/43641225

name	team	GIM	Goals	Assists
Matej Vydra	Derby	18.017	21	4
Leon Clarke	Sheffield United	17.785	19	5
Lewis Grabban	Sunderland	16.045	12	0
Bobby De Cordova-Reid	Bristol	15.976	19	7
Diogo José Teixeira da Silva	Wolverhampton	15.707	17	5
Tom Cairney	Fulham	15.24	5	5
Ivan Cavaleiro	Wolverhampton	14.979	9	12
Stefan Johansen	Fulham	13.565	8	8
James Maddison	Norwich	13.23	14	8
Gary Hooper	Sheffield Wednesday	11.953	10	3

Table 5.3: 2017-2018 season top-10 Player Impact Scores for players in EFL Championship game season.

name	GIM	Goal
Matej Vydra	4.747	21
Leon Clarke	4.024	19
Lewis Grabban	3.775	12
Kouassi Ryan Sessegnon	3.657	15
Harry Wilson	3.135	7
Famara Diedhiou	3.015	13
Sean Maguire	2.5	10
Joe Garner	2.44	10
Jarrod Bowen	2.408	14
Callum Paterson	2.29	10

Table 5.4: Top-10 soccer players with largest shot impact in 2017-2018 EFL Championship game season.

name	GIM	Assist
Leon Clarke	8.05	5
Matej Vydra	5.957	4
Bobby De Cordova-Reid	5.134	7
Chris Wood	4.732	1
Gary Hooper	4.694	3
Ivan Cavaleiro	4.533	12
Diogo José Teixeira da Silva	4.283	5
Gary Madine	4.202	2
Tom Cairney	4.123	5
Conor Hourihane	4.042	2

Table 5.5: Top-10 soccer players with largest pass impact in 2017-2018 EFL Championship game season.

5.4.2 Action-Specific Assessment

An action-specific ranking evaluates only the impacts of action of interest. We compute two GIM rankings of EFL Championship players by shots and passes respectively. These are frequent actions in soccer with high impact. Table 5.4 and Table 5.5 list the top 10 players. GIM computed from shots only can be seen as an alternative to the popular expected goals (XG) metric. A shot with high impact will significantly increase the probability of scoring and thus top players in Table 5.4 also lead the goal scoring. For instance, Matej Vydra is the player with the highest scoring impact and he also dominated goal scoring during 2017-18 game season. However, the relation between pass impact and the number of assists is more complex. There is some association, because assists are often high-value passes. On the other hand, assists are an incomplete measure of passing ability because it neglects midfield and defensive zone passes. Our ranking, in contrast, provides a comprehensive evaluation to **all** the passes of a player. For example, Conor Hourihane plays as Midfielder and managed only 2 assists over the entire season. But he makes many influential passes and is ranked as a top-10 passer by our metric.

5.5 Player Ranking: Empirical Evaluation

We describe our comparison methods and evaluation methodology. Similar to clustering and recommendation problems, there is **no ground truth** for player ranking. To assess a player evaluation metric, we follow previous work [80, 63] and compute its correlation with statistics that directly measure success.

5.5.1 Comparison Player Evaluation Metrics

We compare GIM with the baseline player evaluation metrics to show the advantage of 1) modeling game context 2) incorporating continuous context signal and history 3) separately handling home and away state action signals. We compare GIM with the following baseline player evaluation metrics. i) Plus-Minus **(PM)** is a commonly studied metric that measures how much the presence of a player influences the goals of his team [67]. ii) Expected Goal **(XG)** weights each shot by its chance of leading to a goal. Players are ranked by their total expected goal shots. Both PM and RG consider only very limited game context and action types. The next three baselines assign an impact value to all actions and evaluate players according to their total action impact. iii) Valuing Actions by Estimating Probabilities **(VAEP)** [29] applies the difference of action values to compute the impact of on-the-ball actions. Instead of applying Temporal Difference learning to estimate Q values, VAEP uses a classifier[3] to estimate the probability that an action will be followed by a goal within the next k (window size) events. iv) Scoring Impact **(SI)** is based on a Markov model with *pre-discretized spatial and temporal features* (e.g. x,y coordinate and game time) [84]. Dynamic programming is applied to estimate a Q-function and impact values for the discrete state-action space. v) We examine the option of *merging* the home/away towers and fitting all the states and actions with a single-layer network. We refer to the resulting impact score as (**M-GIM**) for "merge".

We also conduct a league-specific study and evaluate our Fine-Tuning GIM (FT-GIM) for players in the EFL Championship. Training a separate model with only EFL Championship data from scratch consumes more computational resources than fine-tuning the general model. Our experiment records 4,386,894 gradient steps to learn a reliable model from initial weights while fine-tuning requires only 818,120 gradient steps.

Training Settings. We describe the player evaluation as a descriptive task rather than a predictive generalization problem. Our training dataset contains all data in our play-by-play soccer dataset.

[3]The classifier is implemented with a neural network rather than CatBoost (selected by [29]) due to the size of dataset. We discuss our VAEP implementation further in the appendix.

5.5.2 Season Totals: Correlations with Standard Success Measures

We report the correlations between player ranking metrics and commonly used success measures over the entire 2017-18 game season and highlight the comprehensiveness of our GIM metric. The examined success measures include Goals, Assists, Shots per Game (SpG), Pass Success percentage (PS%) and Key Passes per game (KeyP). We also study two penalty measures: Yellow card received (Yel) and Red card received (Red). Table 5.6 shows the correlations between the comparison methods and the success/penalty measures, for the players in all 10 leagues. In addition to the general study, Table 5.7 shows the result of a **league-specific evaluation** where we compare only the correlations for players in the EFL Championship.

Our GIM achieves very good correlations compared to the other comparison methods. Among the positive success measures, GIM has the highest correlation with 4 out of 5 success measures (Goals, Assists, SPG, and KeyP) and a competitive result for the other (PS%). Together, the Q-function based metrics GIM, M-GIM, and SI show the highest correlations with success measures. XG is only the fourth best metric, because it considers only the expected value of shots and does not correct for the team effort leading up to the shot. VAEP achieves only limited correlation with the success measures. This is because their model assigns similar expected values to all actions, which translates into all action impact values being close to 0. The traditional Plus-Minus metric correlates poorly with almost all success measures. We conclude that RL techniques that provide fine-grained expected action-value estimates lead to better performance metrics.

Comparing the different RL approaches, the neural network model allows GIM to handle continuous inputs without pre-discretization. This prevents the loss of game context information and explains why both GIM and M-GIM perform better than SI in most success measures. The higher correlation of GIM compared to M-GIM also demonstrates the value of separately modeling home/away data. For Yel and Red which reflect the number of received penalties—negative contributions by a player—only our GIM-based metrics (GIM, M-GIM) show a negative correlation with both of them. The model correctly recognizes that a penalty will significantly reduce the scoring probability, influencing the overall player GIM. In contrast, other metrics, focus on the actions that are likely to lead to goals, which tends to reward aggressive players who incur more penalties.

Methods	Goals	Assists	SpG	PS%	KeyP	Yel	Red
PM	0.284	0.318	0.199	**0.288**	0.218	0.001	-0.069
VAEP	0.093	0.290	0.121	-0.111	0.116	0.024	0.133
XG	0.422	0.173	0.328	0.164	0.278	0.534	0.034
SI	0.585	0.153	0.438	-0.140	0.052	0.114	-0.089
M-GIM	0.648	0.367	0.573	0.153	0.417	-0.110	<u>-0.145</u>
GIM	**0.844**	**0.498**	**0.596**	0.16	**0.562**	<u>-0.181</u>	-0.137

Table 5.6: Correlation with standard success measures for all the players. We bold the highest correlations and underline the lowest ones for penalties.

Methods	Goals	Assists	SpG	PS%	KeyP	Yel	Red
PM	0.262	0.223	0.122	0.155	0.112	0.033	-0.046
VAEP	0.08	0.26	0.116	-0.126	0.137	-0.015	0.215
XG	0.420	0.165	0.394	0.149	0.254	0.578	-0.021
SI	0.574	0.124	0.408	-0.144	0.054	0.084	-0.147
M-GIM	0.629	0.309	0.551	**0.171**	0.388	-0.039	-0.132
GIM	0.638	0.382	0.553	-0.053	0.468	-0.026	-0.105
FT-GIM	**0.736**	**0.585**	**0.569**	0.082	**0.592**	<u>-0.110</u>	<u>-0.171</u>

Table 5.7: Correlation with standard success measures for players in the EFL Championship. We bold the highest correlations and underline the lowest ones for penalties.

The league-specific study demonstrates the benefit of fine-tuning, for the machine learning models. Compared to the correlations for players in all 10 leagues, Championship League players' correlations generally decrease. Both traditional action-count metrics (PM, XG) and impact-based metrics (VAEP, SI, GIM, M-GIM) show the decrease, but it is more severe for our GIM metric whose correlations nearly drop 20% when the players in the Championship League are evaluated by the general model. Fine-tuning addresses this issue: the FT-GIM metric achieves a larger negative correlation with both penalty counts (Yel and Red).

5.5.3 Round-by-Round Correlations: Predicting Future Performance From Past Performance

This experiment assesses the player performance metrics through round-by-round correlations. A sports season can be divided into **rounds**. In round n, a team or player has finished n games in a season. For a given performance metric, we measure the correlation between (i) its value computed *over the first n rounds*, and (ii) the value of the two main success measures, assists, and goals, computed *over the entire season*. This allows us to assess how quickly different metrics acquire predictive power for the final season total, so that future performance can be predicted from past performance. A good performance metric should be consistent with a player's overall performance in the early season, which provides the player and his team with evidence for trading or training.

Figure 5.6 [4] shows the round-by-round correlations for the players in all 10 leagues. The predictive power of GIM grows more quickly than with any other baseline: its correlation with both assists (left) and goals (right) dominates others before the first half of the season. M-GIM achieves the second highest correlations, for assists even higher than GIM in the first 5 rounds. However, its predictive power substantially drops after the first 10 rounds. The remaining two metrics XG and SI show only weak correlations with assists and goals.

The question for our next experiment is: *does fine-tuning help predict a player's final total performance from the past performance?* This experiment focuses on players in the EFL Championship. Figure 5.7 shows round-by-round correlations of the performance metrics with EFL Cham-

[4] In the Figure 5.6 and 5.7, we omit the players whose team play less than 40 game in the 2017-18 season.

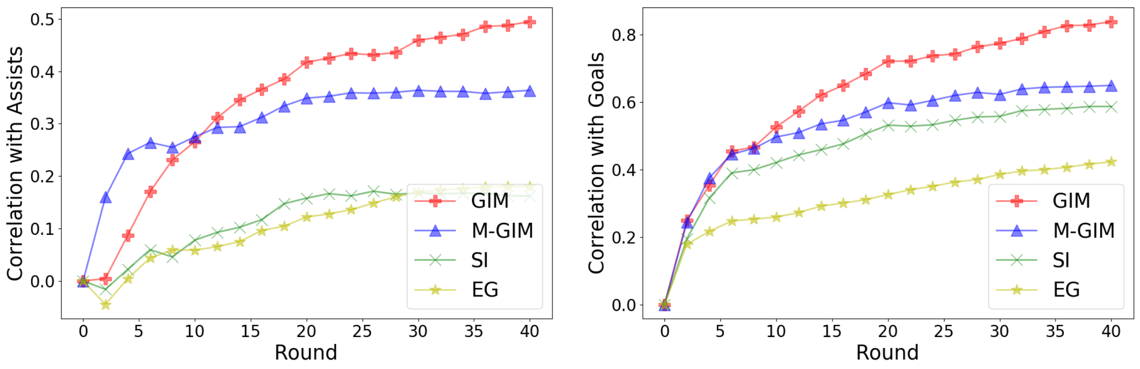


Figure 5.6: Correlations between round-by-round metrics and season totals for all players.

pionship players' total assists and goals. We make the following observations. 1) Compared to the all-player setting of Figure 5.6, the metrics' correlations decline when restricted to EFL Championship players. This decline is more apparent for our GIM metric. The reason is that the neural network trained on the general player population does not fit the behaviour of players in the EFL Championship as well. 2) Fine-tuning significantly improves the correlations of GIM, especially for its correlation with assists, where the correlation of FT-GIM exceeds that of other metrics after the first 10 rounds.

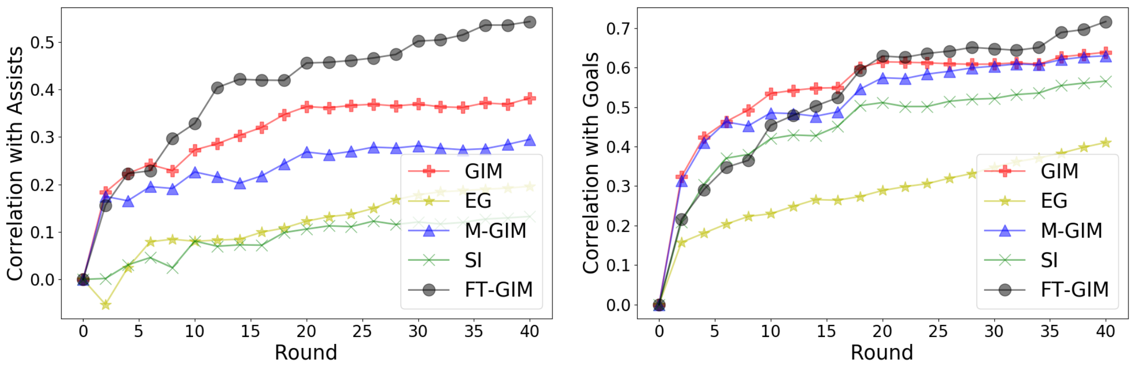


Figure 5.7: Correlations between round-by-round metrics and season totals for the players in EFL Champion.

5.6 Summary

This chapter introduced the approach of applying Deep Reinforcement Learning (DRL) to learn complex spatio-temporal dynamics for professional soccer analytics. We designed a neural network architecture that to our knowledge is the most complex deployed in sports analytics to date: A stacked two-tower LSTM architecture, with one tower each for home and away teams. The network was trained with on-ball action logs from several European leagues, comprising a total of over 4.5M action events. The trained neural network provides a rich source of knowledge about how a team's chance of scoring the next goal depends on the match context.

Based on the learned action values, we computed a new context-aware performance metric GIM for soccer players, taking into account **all** of their actions. In our experiments, GIM computed over the entire season showed the highest correlation with most standard success measures. Generalizing from a sample of season matches, GIM was the best predictor of season total goals and assists. To improve the evaluation results for players in a specific league, we applied a fine-tuning approach to achieve an effective balance between generalizing across leagues and specializing to a specific league.

Deep RL methods have enjoyed spectacular success in board games. Our results show that the analysis of physical team sports is another highly promising application area.

Chapter 6

Understanding the Action Values with Interpretable Mimic Learning

6.1 Introduction: Mimic a Deep Reinforcement Learner

Deep Reinforcement Learning has mastered human-level control policies in a wide variety of tasks [71]. Despite excellent performance, the learned knowledge remains implicit in neural networks and hard to explain: there is a trade-off between model performance and interpretability [60]. One of the frameworks for addressing this trade-off is mimic learning [9], which seeks a sweet spot between accuracy and efficiency, by training an interpretable mimic model to match the predictions of a highly accurate model. Many works [24, 14, 27] have developed types of mimic learning to distill knowledge from deep models to a mimic model with tree representation.

Mimic Learning for Sports Q-functions Following the mimic learning framework, We build a traditional regression tree as a mimic model to interpret the action-value function. We first learn general regression trees for all players, for both Q and impact functions. To understand the Q functions, we compute the feature importance and use partial dependence plots to analyze the influence of different features with the mimic trees. To highlight the strengths and weaknesses of an individual player compared to a general player, we construct player-specific mimic trees for Q values and impact. Based on a player-specific tree, we define an interpretable measure for which players are most exceptional overall.

To strengthen the generalization ability of the mimic model, we add a linear model to each leaf node, which defines a novel Linear Model U-Tree (LMUT). U-tree [69, 99] is a classic online reinforcement learning method which represents a Q function using a tree structure. We examine the performance of LMUT under professional sports environments. This experiment evaluates four node-splitting methods for LMUT and interprets the learned tree model by computing the feature importance as well as rule extractions. Our empirical results show that the time remaining and the action blocked are the two most important features for Ice Hockey and Soccer. The extracted rules also highlight relevant interactions among different features, which significantly facilitates understanding of the strategy for different sports.

Mimic Learning for General DRL This chapter introduces a novel mimic learning framework for Reinforcement Learning in a control setting, where the mimic learning can execute actions. We examine two different approaches to *generating data for RL mimic learning*. Within the first *Experience Training* setting, which allows applying traditional batch learning methods to train a mimic model, we record all state-action pairs during the training process of DRL and complement them with Q values as soft supervision labels. Storing and reading the training experience of a DRL model consumes much time and space, and the training experience may not even be available to a mimic learner. Therefore our second *Active Play* setting generates streaming data through interacting with the environment using the mature DRL model. The active play setting requires an on-line algorithm to dynamically update the model as more learning data is generated. We introduce a novel online learning algorithm for LMUT, which applies Stochastic Gradient Descent to update the linear models, given some memory of recent input data stored on each leaf node.

We conducted an empirical evaluation in three virtual (game) environments with five baseline methods. Two natural evaluation metrics for an RL mimic learner are: 1) fidelity [27]: how well the mimic model matches the predictions of the neural net, as in supervised learning, and 2) *play performance*: how well the average return achieved by a controller based on the mimic model matches the return achieved by the neural net. Play performance is the most relevant metric for reinforcement learning. Perfect fidelity implies a perfect match in play performance. However, our experiments show that approximate fidelity does not imply a good match in play performance. This is because RL mimic learning must strike a balance between coverage: matching the neural net across a large section of the state space, and optimality: matching the neural net on the states that are most important for performance. In our experiments, LMUT learning achieves a good balance: the best match to play performance among the mimic methods, and competitive fidelity to the neural net predictions. The transparent tree structure of LMUT makes the DRL neural net interpretable. To analyze the mimicked knowledge, we calculate the importance of input features and extract rules for typical examples of agent behavior. For image inputs, the super-pixels in input images are highlighted to illustrate the key regions.

6.2 Background and Related Work

We introduce the previous works that are most related to our work.

6.2.1 Mimic Learning

Recent works on mimic learning [9, 24, 27] have demonstrated that models like shallow feed-forward neural networks or decision trees can mimic the function of a deep neural net. In the *oracle* *framework*, soft output labels are collected by passing inputs to a large, complex and accurate deep neural network [49] Then we train a mimic model with the soft output as supervisor. The results indicate that training a mimic model with soft output achieves substantial improvement in accuracy and efficiency, over training the same model type directly with hard targets from the dataset.

6.2.2 U-Tree Learning

A tree structure is transparent and interpretable, allowing rule extraction and measuring feature influence [24]. U-tree [69] learning was developed as an online reinforcement learning algorithm for a tree structure representation. A U-tree takes a set of observed feature/action values as input and maps it to a state value (or Q-value). [99] introduces the continuous U-tree (CUT) for continuous state features. CUT learning generates a tree-based discretization of the input signal and estimates state transition probabilities in every leaf node [99]. Dynamic programming is applied to solve the resulting Markov Decision Process (MDP). Although CUTs have been successfully applied in test environments like Corridor and Hexagonal Soccer, constructing a CUT from raw data is rather slow and consumes significant computing time and space.

6.3 Mimic Learning for Sports Environment

In this section, we introduce the method of learning a Classification And Regression Tree (CART) to mimic the action-value Q function learned by the DRL model (discussed in Chapter 4 and Chapter 5). To expand the generalization ability, we propose a Linear Model U-Tree (LMUT) which adds a linear model at each leaf node of a regression tree. Empirical evaluations are conducted to evaluate the fidelity and the interpretability of both mimic models.

6.3.1 Mimic Learning with a Regression Tree

We apply Mimic Learning [9] and train a transparent regression tree to mimic the black-box neural network. As it is shown in Figure 6.1, our framework aims at mimicking Q functions (Equation 4.1) and impact (Equation 4.4). We first train the *general tree model* with the deep model's input/output for all players and then use it to initialize the *player-specific model* for an individual player. The transparent tree structure provides much information for understanding the Q functions and impact.

We focus on two mimicking targets: *Q functions* and *Impact*. For Q functions, we fit the mimic tree with the NHL play data and their associated soft outputs (Q values) from our DRL model (neural network). The last 10 observations (determined experimentally) from the sequence are extracted, and CART learning is applied to fit the soft outputs. This is a multi-output regression task, as our DRL model outputs a Q vector containing three Q values ($\hat{Q}_t = \langle \hat{Q}_t^{home}, \hat{Q}_t^{away}, \hat{Q}_t^{end} \rangle$) for an observation features vectors (s_t) and an action (a_t). A straightforward approach for the multi-target regression problem is training a separate regression model for each Q value. But separate trees for each Q function are somewhat difficult to interpret. An alternative approach to reduce the total

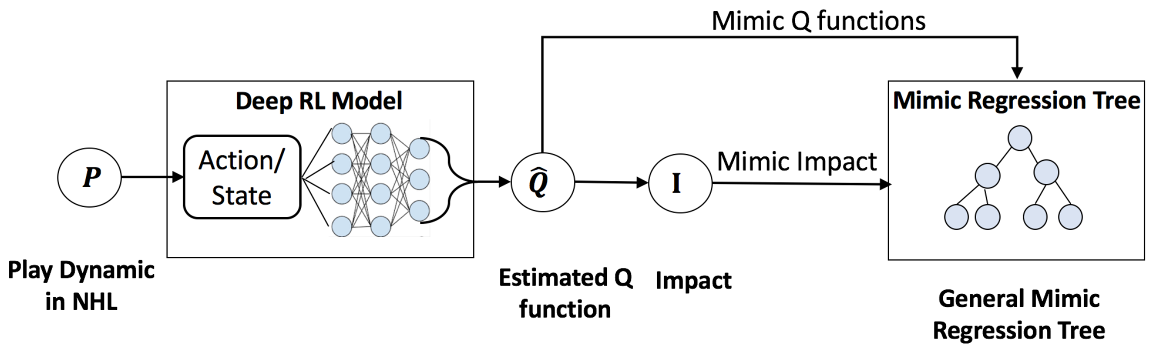


Figure 6.1: Interpretable Mimic Learning Framework

tree size is training a Multi-variate Regression Tree (MRTs) [28], which fits all three Q values simultaneously in a regression tree. An MRT can also model the dependencies between the different Q variables [91]. For Impacts, we have only one output ($impact_t$) for each sequence (s_t) and current action (a_t) at time step t.

Fidelity: Evaluating Regression Performance

We examine the mimic performance of regression tree for the Q functions and impact. A common problem of regression trees is over-fitting. We use the Mean Sample Leaf (MSL) to control the minimum number of samples at each leaf node. We apply ten-fold cross validation to measure the performance of our mimic regression tree by Mean Square Error (MSE) and variance. As is shown in Table 6.1, the tree achieves satisfactory performance when MSL equals 20 (the minimum MSE for Q functions, small MSE and variance for impact).

model	Q_home	Q_away	Q_end	Impact
RT-MSL1	3.35E-04 (1.43E-09)	3.21E-04(1.26E-09)	1.74E-04(2.18E-09)	1.33E-03(5.43E-09)
RT-MSL5	2.59E-04(1.07E-09)	2.51E-04(0.89E-09)	1.35E-04(1.87E-10)	9.84E-04(2.72E-09)
RT-MSL10	2.38E-04(1.02E-09)	2.30E-04(0.89E-09)	1.25E-04(2.30E-10)	8.66E-04(2.17E-09)
RT-MSL20	**2.31E-04(0.92E-09)**	**2.22E-04(0.82E-09)**	**1.23E-04(2.05E-10)**	7.92E-04(1.45E-09)
RT-MSL30	2.35E-04(0.98E-09)	2.27E-04(0.85E-09)	1.27E-04(2.32E-10)	7.67E-04(1.16E-09)
RT-MSL40	2.39E-04(0.96E-09)	2.30E-04(0.85E-09)	1.29E-04(2.19E-10)	**7.58E-04(1.10E-09)**

Table 6.1: Performance of General Mimic Regression Tree (RT) with different Minimum Samples in each Leaf node (MSL). We apply ten-fold cross validation and report the regression result with format: Mean Square Error (Variance)

Interpreting Q functions and Impact with Mimic Regression Tree

We now show how to interpret Q functions and Impact using the general Mimic tree, by deriving feature importance and a partial dependence plot.

Feature Importance for Q functions. In CART regression tree learning, variance reduction is the criterion for evaluating the quality of a split. Therefore we compute the importance of a target feature by summing the variance reductions at each split using the target feature [24]. We list the top 10 important features in the mimic tree for Q values in Table 6.2. The frequency of a feature is the number of times the tree splits on the feature. The notation $f(t-n)$ indicates that a feature f occurs n time steps before the current time. For Q values, remaining game time is the most influential feature with significantly larger importance value than others. This is because less time means fewer chances of any goals. Manpower is the second important feature because a team will have a less scoring chance if it is short-handed, which well matched the domain knowledge of sports games.

Feature Name	Frequency	Importance
Game Time Remain (t)	12,524	0.817431
Manpower (t-1)	93	0.070196
Team Identifier (t-1)	57	0.020504
Manpower (t)	346	0.017306
Shot (t)	31	0.011159
Score Differential (t)	3,229	0.009568
X Coordinate (t)	11,797	0.006968
X Coordinate (t-1)	3,406	0.006963
Manpower (t-2)	82	0.005045
Home/Away Team (t)	135	0.003755

Table 6.2: Top 10 features for Q values.

Feature	Influence
X distance (t)	0.6632
Outcome (t)	0.2275
Y distance (t)	0.0469
Game Time Remain (t)	0.0242
Duration (t)	0.0062
X Coordinate (t-1)	0.0059
Game Time Remain (t-1)	0.0035
Interrupted (t)	0.0035
X velocity (t)	0.0030
Outcome (t-1)	0.0019

Table 6.3: Feature importance for the impact of shot.

Feature	Influence
X Velocity (t)	0.1355
Distance to Goal(t)	0.1264
Game Time Remain (t-1)	0.1082
Game Time Remain (t)	0.0816
Outcome (t)	0.0773
Outcome (t-1)	0.0760
Distance to Goal (t-1)	0.0411
Angle (t)	0.0373
Angle (t-1)	0.0298
X Velocity (t-1)	0.0174

Table 6.4: Feature importance for the impact of pass.

Feature Importance for Impacts. The impact values are computed with the Q-function (Equation (4.4). We rank the state and action features by their importance values. Tables 6.3 and 6.4 show the top 10 important features for shot and pass. Figure 6.2 and Figure 6.3 illustrate the structure of the CART trees by plotting its top three layers. The trees for both shot and pass impacts place at the root action outcome (a binary feature marking success or failure of an action), which intuitively is one of the most important action features. We also find that the shot impact significantly increases as a player approaches the goal, which is consistent with our finding in the spatial projection for Q values. For passing, its impact increases with game velocity. An explanation is that a quick pass prevents potential interruptions from opponents. When the game is ending, we observe that while the average passing impact decreases, the big difference is an increase in pass impact variance. We illustrate this in the plot of figure 6.4. It shows that as the game comes to an end, the variance of pass impact values is higher (marked by the solid rectangle and the dashed rectangle). Our decision tree (Figure 6.3) accurately locates the time when this phenomenon starts to occur (39.45 marked by the dashed line) and shows that the average passing impact will become smaller (around -0.02) after this time. Our hypothesis for the increase in variance is that towards the end of the game, players tend to take riskier passes, with potentially higher impact on the game, both good and bad. The reduction in average impact is a smaller effect compared to the increase in variance. It is generally consistent with the temporal projection of Q-values (Figure 5.2). A direct and probably sufficient explanation is that the expected next-goal scoring probability (Q value) for home/away team becomes smaller as the game ends and it flattens the corresponding impact values on average. Another important observation is that in addition to features from current time t, the historical features (e.g. X Coordinate (t-1)) are also considered as important for predicting the impact of the current action.

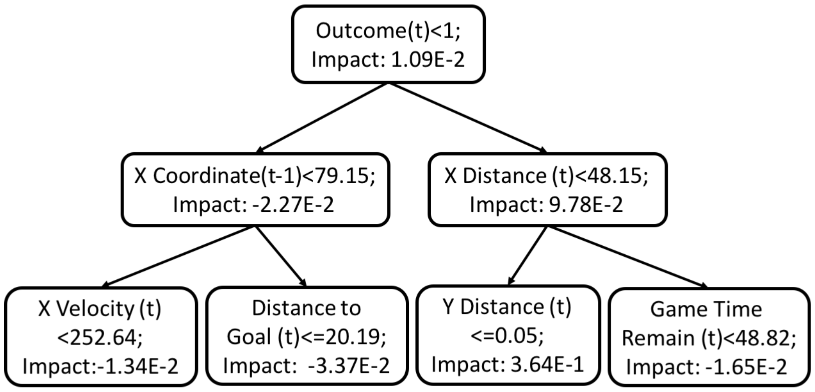


Figure 6.2: Regression tree for the impact of shot.

Partial Dependence Plots. A partial dependence plot is a common visualization to determine qualitatively what a model has learned and thus provides interpretability [60, 24]. The plot approximates the prediction function for a single target feature, by marginalizing over the values of all other features. We select X Coordinate (of puck), Time Remaining and X Velocity (of puck), three continuous features with high importance for both the Q and the impact mimic tree. As it is shown in Figure 6.5, Time Remaining has significant influence on Q values but very limited effect on impact. This is consistent with our findings for feature importance. For X Coordinate, as a team is likely to

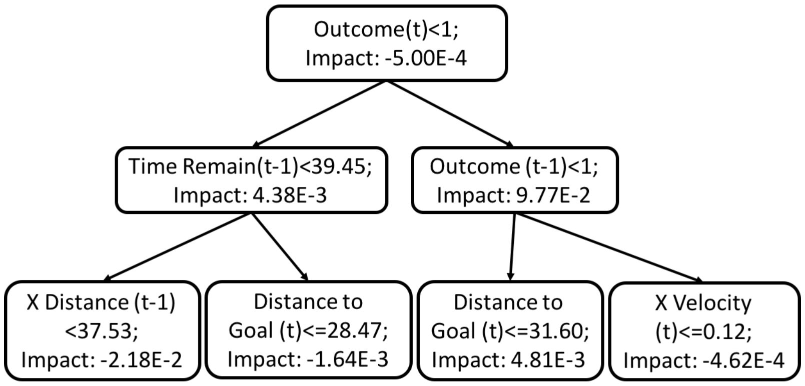


Figure 6.3: Regression tree for the impact of pass.

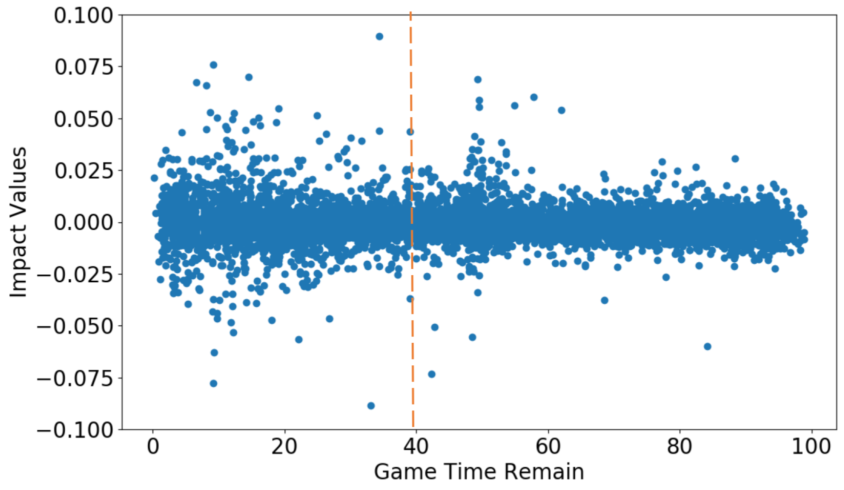


Figure 6.4: The impact values for passes v.s. Game Time **Remain**. We randomly sample 10 games and plot the impact values for all passes.

score the next goal in the offensive zone, both Q values and impact increase significantly when the puck is approaching its opponent's goal (larger X Coordinate). And compared to the position of the puck, velocity in X-axis has limited influence on Q values but it does affect the impact. This shows that the impact function uses speed on the ice as an important criterion for valuing a player. We also observe the phenomenon of home advantage [95] as the Q value (scoring probability) of the home team is slightly higher than that of the away team.

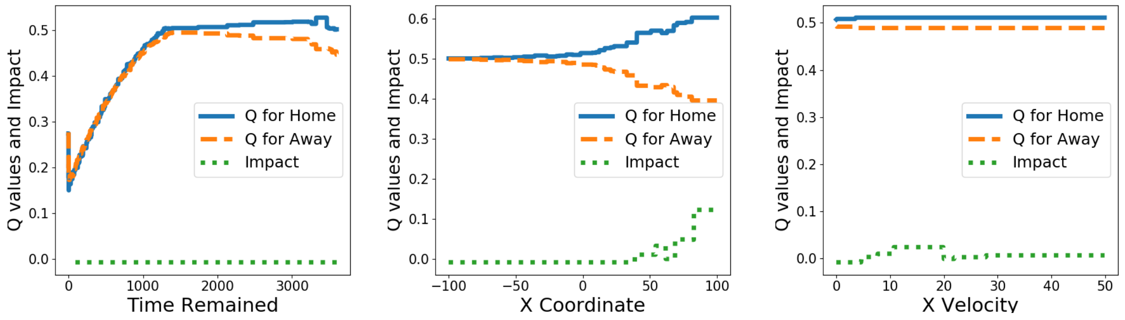


Figure 6.5: Partial Dependence Plot for Time Remaining (left), X Coordinate (middle) and X Velocity (right)

Highlighting Exceptional Players

Our approach to quantifying which players are exceptional is based on a partition the continuous state space into a discrete set of m disjoint regions. Given a Q or impact function, exceptional players can be found by region-wise comparison of a player's excepted impact to that of a random player's. For a specific player, this comparison highlights match settings in which the player is especially strong or weak. The formal details are as follows.

Let n_D be the number of actions by player P, of which n_ℓ fall into discrete state region $\ell = 1, \ldots, m$. For a function ψ, let $\hat{\psi}_\ell$ be the value of ψ estimated from *all* data points that fall into region ℓ, and let $\hat{\psi}_\ell^P$ be the value of ψ estimated from the n_ℓ data points for region ℓ and player P. Then the weighted squared ψ-difference is given by:

$$\sum_\ell n_\ell / n_D (\hat{\psi}_\ell - \hat{\psi}_\ell^P)^2. \tag{6.1}$$

Regression trees provide an effective way to discretize a Q-function for a continuous state space [99]: Each leaf forms a partition cell in state space (constructed by the splits with various features along the path from root to the leaf). The regression trees described in Section 6.3.1 could be used, but they represent *general* discretizations learned for all the players over a game season, which means that they may miss distinctions that are important for a specific player. For example, if an individual player is especially effective in the neutral zone, but the average player's performance is not special in the neutral zone, the generic tree will not split on "neutral zone" and therefore will not be able to capture the individual's special performance. Therefore we learn for each player, a *player-specific* regression tree.

The General Tree is learned with all the inputs and their corresponding Q or Impact values (soft labels). The Player Tree is initialized with the General Tree and then fitted with the n_D datapoints of a specific player P and their corresponding Q values ($\psi^P_{\hat{Q}}(\psi_{\hat{Q}}, s^P_t, a^P_t) \rightarrow range(\hat{Q}^P_t)$) or Impact values ($\psi^P_I(\psi_I, s^P_t, a^P_t) \rightarrow range(Impact^P_t)$). It inherits the tree structure of the general model RT-MSL20 in table 6.1, uses the target player data to prune the general tree, then expands the tree with further splits. Initializing with the general tree assumes players share relevant features and prevents over-fitting to a player's specific data. A Player Tree defines a discrete set of state regions, so we can apply Equation 6.1 with the Q or impact functions. Table 6.5 shows the weighted squared differences for the top 5 players in the GIM metric.

Player	Q_home	Q_away	Q_end	Impact
Taylor Hall	1.80E-04	2.49E-04	2.28E-04	6.66E-05
Joe Pavelski	**4.64E-04**	**2.90E-04**	**3.04E-04**	1.09E-04
Johnny Gaudreau	2.12E-04	1.96E-04	1.43E-04	6.77E-05
Anze Kopitar	2.58E-04	2.00E-04	2.43E-04	8.28E-05
Erik Karlsson	2.97E-04	1.89E-04	1.86E-04	**2.00E-04**

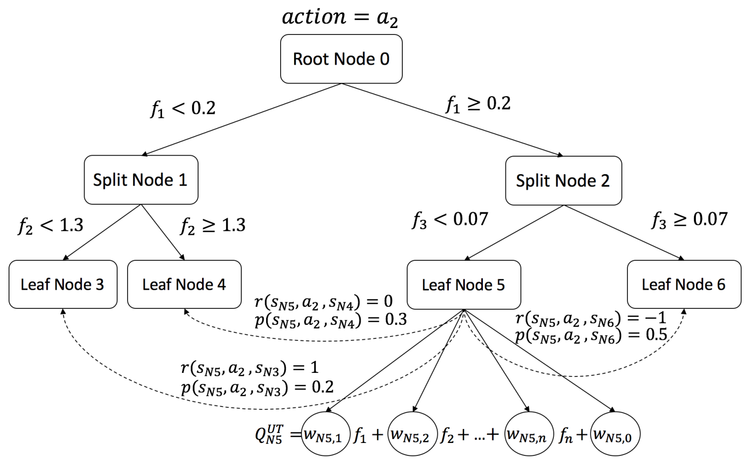
Table 6.5: Exceptional Players Based on Tree Discretization

We find that 1) Joe Pavelski, who scored the most in the 2015-2016 game season, has the largest Q values difference and 2) Erik Karlsson, who had the most points (goal+assists), has the largest Impact difference. They are the two players who differ the most from the average players by Q-value and Impact.

6.3.2 Mimic Learning with a Linear Model U-Tree

A neural network with continuous activation functions computes a continuous function. A regression tree can approximate a continuous function arbitrarily closely, given enough leaves. Continuous U-Trees (CUTs) are essentially regression trees for value functions, and therefore a natural choice for a tree structure representation of a DRL Q function. However, their ability to generalize is limited, and CUT learning converges slowly. We introduce a *novel extension of CUT*, Linear Model U-Tree (LMUT), that allows CUT leaf nodes to contain a linear model, rather than simple constants. Being strictly more expressive than a regression tree, a linear model tree can also approximate a continuous function arbitrarily closely, with typically many fewer leaves [23]. Smaller trees are more interpretable, and therefore more suitable for mimic learning.

As shown in Figure 6.6 and Table 6.6, each leaf node of a LMUT defines a partition cell of the input space, which can be interpreted as a discrete state s for the decision process. Within each partition cell, LMUT also records the reward r and the transition probabilities p of performing action a on the current state s, as shown in the Leaf Node 5 of Figure 6.6. So LMUT learning builds a Markov Decision Process (MDP) from the interaction data between environment and deep model. Compared to a linear Q-function approximator [94], a LMUT defines an ensemble of linear Q-functions, one for each partition cell. Since each Q-value prediction Q_N^{UT} comes from a single linear model, the prediction can be explained by the feature weights of the model.



$action = a_2$
Root Node 0
$f_1 < 0.2$
$f_1 \geq 0.2$
Split Node 1
Split Node 2
$f_2 < 1.3$
$f_2 \geq 1.3$
$f_3 < 0.07$
$f_3 \geq 0.07$
Leaf Node 3
Leaf Node 4
Leaf Node 5
Leaf Node 6
$r(s_{N5}, a_2, s_{N4}) = 0$
$p(s_{N5}, a_2, s_{N4}) = 0.3$
$r(s_{N5}, a_2, s_{N6}) = -1$
$p(s_{N5}, a_2, s_{N6}) = 0.5$
$r(s_{N5}, a_2, s_{N3}) = 1$
$p(s_{N5}, a_2, s_{N3}) = 0.2$
$Q_{N5}^{UT} = w_{N5,1} f_1 + w_{N5,2} f_2 + \ldots + w_{N5,n} f_n + w_{N5,0}$

Figure 6.6: An example of Linear Model U-Tree (LMUT).

Node Name	Partition Cell
Leaf Node 3	$f_1 < 0.2$, $f_2 < 1.3$
Leaf Node 4	$f_1 < 0.2$, $f_2 \geq 1.3$
Leaf Node 5	$f_1 \geq 0.2$, $f_2 < 0.07$
Leaf Node 6	$f_1 \geq 0.2$, $f_2 \geq 0.07$

Table 6.6: Partition Cell

The Training Process of LMUT

We now discuss how to train an LMUT. Similar to [99], we separate the training into two phases: 1) Data Gathering Phase and 2) Node Splitting Phase.

Data Gathering Phase. Data Gathering Phase assigns transitions to leaf nodes and prepares them for fitting linear models and splitting nodes. Given an input transition $\mathcal{T}$, we pass it through feature splits down to a leaf node. As an option, an LMUT can dynamically build an MDP, in which case it updates transition probabilities, rewards and average Q values on the leaf nodes. The complete Data Gathering Phase process is detailed in part I (the first for loop) of Algorithm 1.

Node Splitting Phase. After node updating, LMUT scans the leaf nodes and updates their linear model with Stochastic Gradient Descent (SGD). If SGD achieves insufficient improvement on node N, LMUT determines a new split and adds the resulting leaves to the current partition cell. For computational efficiency, our node splitting phase considers only a single split for each leaf given a single minibatch of new transitions. Part II of Algorithm 1 shows the detail of the node splitting phase. LMUT applies a minibatch stagewise fitting approach to learn linear models in the leaves of the U-tree. Like other stagewise approaches [58], this approach provides smoothed weight estimates where nearby leaves tend to have similar weights. We use Stochastic Gradient Descent to implement the weight updates.

Stochastic Gradient Descent (SGD) Weight Updates. SGD is a straightforward well-established online weight learning method for a single linear regression model. The weights and bias of linear

Algorithm 1: Linear Model U-Tree Learning

```
Input: Transitions $\mathcal{T}_1, \ldots, \mathcal{T}_B$; A LMUT with leaf nodes $N_1, \ldots, N_L$, each with weight
       vector $\mathbf{w_1}, \ldots, \mathbf{w_L}$

for $t = 1$ to $B$ do
    Find the partition cell on leaf node $N$ by $I_t, a_t$ in $\mathcal{T}_t$
    Add $\mathcal{T}_t = \langle I_t, a_t, r_t, I_{t+1}, \hat{Q}_t(I_t, a_t)\rangle$ to transition set on $N$
    if $FlagMDP$            /* Update the Markov Decision Process */
    then
        Map observation $(I_t, I_{t+1})$ to state $(s_t, s_{t+1})$ within partition cell of N
        Update Transitions Probability $P(s_t, a_t, s_{t+1}) = \frac{count(s_t,a_t,s_{t+1})+1}{\sum_i count(s_t,a_t,s_i)+1}$
        Update Reward $R(s_t, a_t, s_{t+1}) = \frac{R(s_t,a_t,s_{t+1}) * count(s_t,a_t,s_{t+1}) + r_t}{count(s_t,a_t,s_{t+1})+1}$
        Compute $Q^{UT}_{avg}(s_t, a_t) = \frac{\hat{Q}_t(s_t,a_t) * count(s_t,a_t) + \hat{Q}_t(I_t,a_t)}{count(s_t,a_t)+1}$
        Increment $count(s_t, a_t)$ and $count(s_t, a_t, s_{t+1})$ by 1
    end
end

for $i = 1$ to $L$ do
    $w_i, err_i$ := WeightUpdate($\mathcal{T}_{\mathbf{N_i}}, \mathbf{w_i}$)     /* Update the weights by SGD */
    if $err \leq MinImprovement$ then
        for distinction D in GetDistinction($N_i$) do
            Split Node $N_i$ to FringeNodes by distinction D
            Compute distribution of Q function $\sigma_{N_i}(Q)$ on Node $N_i$
            for each node $F$ in FringeNodes do
                Compute distribution $\sigma_F(Q)$
                $p$ = SplittingCriterion($\sigma_{N_i}(Q)$, $\sigma_F(Q)$)
                if $p \geq minSplit$ then
                    $BestD = D$
                    $minSplit = p$
                end
                Remove all the fringe nodes
            end
        end
        if $BestD$ then
            Split Node $N_i$ by $BestD$ to define ChildNodes $N_{i,1}, \ldots, N_{i,C}$
            Assign Transitions set $\mathcal{T}_{\mathbf{N_i}}$ to ChildNodes
            for $c = 1$ to $C$ do
                $w_{i,c} := w_i$
                $w_{i,c}, err_{i,c}$ := WeightUpdate($\mathcal{T}_{\mathbf{N_{i,c}}}, \mathbf{w_{i,c}}$)
            end
        end
    end
end
```

regression on leaf node N are updated by applying SGD over all Transitions assigned to N. For a transition $\mathcal{T}_t = \langle I_t, a_t, r_t, I_{t+1}, \hat{Q}(I_t, a_t)\rangle$, we take I_t as input and $\hat{Q}_t \equiv \hat{Q}(I_t, a_t)$ as label. We build a separate LMUT for each action, so the linear model on N is function of the J state features: $Q^{UT}(I_t|w_N, a_t) = \sum_{j=1}^{J} I_{tj} w_{Nj} + w_{N0}$. We update the weights w_N on leaf node N by applying SGD with loss function $\mathcal{L}(w_N) = \sum_t 1/2(\hat{Q}_t - Q^{UT}(I_t|w_N, a_t))^2$. The updates are computed with a single pass over each minibatch.

Algorithm 2: SGD Weight Update at a leaf node

```
**Input:** Transitions $\mathcal{T}_1, \dots, \mathcal{T}_m$, node N = leaf node with weight vector $w_0$
**Output:** updated weight vector $w$, training error $err$
**Hyperparameters:** number of iterations $E$; step size $\alpha$
$w := w_0$ ;
for $e$ = 1 to $E$ do
    for t=1 to $m$ do
        $w := w + \alpha \nabla_w \mathcal{L}(w)$ ;
    end
end
Compute training error $err = 1/m \sum_{t=1}^{m} (\hat{Q}_t - Q^{UT}(I_t|w, a_t))^2$
```

Splitting Criterion. We investigate several fast heuristic methods for selecting promising splits f_i (feature value) for a given input feature I_i (i denotes the index of feature, or the i^{th} dimension of inputs I). These heuristics are crucial for both computational feasibility and fidelity. The key idea behind our methods is to sort all the data points by their I_i-value, then choose a split (or breakpoint) f_i that maximizes the difference in the Q-distributions of the split groups created by f_i (We refer to the group of data points with $I_i \leq f_i$ and $I_i > f_i$ as the split groups). Our proposed heuristics combine sorting with variance reduction and t-test, or use segmented regression with efficient iterative estimation as a subroutine to achieve fast performance on large datasets. We apply heuristic with Gaussian Mixture as our baseline.

- *Sorting with Variance Reduction:* Maximizing the difference in the Q-distributions of the datapoint groups after a split can be estimated by variance reduction on Q. Simply sorting first on I_i allows us to incrementally estimate the variance reduction for every x_i-value quickly with a single pass through the dataset, as shown by the following equations:

$$\begin{aligned}\sigma^2 &= \frac{1}{N_1} \cdot \sum_{n=1}^{N_1} (Q_n - \mu)^2 \\ &= (\sum_{n=1}^{N_1} \frac{{Q_n}^2}{N_1}) - (\sum_{n=1}^{N_1} \frac{Q_n}{N_1})^2 \end{aligned} \tag{6.2}$$

where N_1 represents the data points in one split group after splitting on an I_i-value, μ is the Q mean of the split group. Both terms in Equation 6.2 are calculated incrementally in a single pass for all I_i-values.

- *Sorting with T-test:* This method also sorts all the data points by their I_i-value. Then, it uses the test-statistic of two-sample Welch's t-test [59] to evaluate breakpoints.

  $$\text{t-score} = \frac{\mu_1 - \mu_2}{\sqrt{\frac{{\sigma_1}^2}{N_1} + \frac{{\sigma_2}^2}{N2}}}$$

  The t-test measures the y-difference between the two split groups sperated by an x_i-value. We select the x_i-value that produces the largest t-score as breakpoint c_i. As with variance reduction, the t-score can also be computed incrementally in linear time.

- *Iterative Segmented Regression:* Segmented regression performs a piecewise linear regression of y on I_i with a breakpoint f_i between two line segments [101].

  $$\hat{y} = \begin{cases} \alpha \cdot x_i & \text{for } I_i \leq f_i \\ (\alpha + \beta) \cdot x_i - \beta \cdot f_i & \text{for } I_i > f_i \end{cases}$$

  We first use segmented regression as a subroutine with an efficient iterative approach [72] to find a breakpoint candidate on each feature I_i. The following algorithm elaborates on the iterative approach for a feature I_i at iterative step s:

  -
    $$U^s = \begin{cases} I_i - f_i^s & \text{for } I_i > f_i^s \\ 0 & \text{otherwise} \end{cases}$$

    $$V^s = \begin{cases} -1 & \text{for } I_i > f_i^s \\ 0 & \text{otherwise} \end{cases}$$

  - fit the model
    $$\hat{y} = \alpha \cdot I_i + \beta \cdot U^s + \gamma \cdot V^s$$

  - update the breakpoint f_i
    $$f_i^{s+1} = \frac{\gamma}{\beta} + f_i^s$$

  - repeat the process until the breakpoint f_i is converged or the maximum iterative step is reached.

  Then, for each breakpoint candidate, we calculate the y-variances of two groups separated by the breakpoint candidate. We select the breakpoint candidate that maximizes the difference in the y-variances of the two split groups as the breakpoint f_i.

- *Gaussian Mixture:* This method uses the expectation maximization algorithm to calculate a two-component bivariate Gaussian mixture model [31] for (I_i, Q) data pairs.

$$p(I_i, Q) = \sum_{k=1}^{2} \pi_k \cdot \mathcal{N}(I_i, Q|\mu_k, \Sigma_k)$$

  Then, the breakpoint c_i that best separates the two Gaussian clusters on each predictor variable x_i can be computed in closed form by quadratic discriminant analysis.

Fidelity: Evaluating Regression Performance

A mimic model with strong fidelity [27] should achieve a small prediction error, that is, the root mean squared difference (RMSE) between the prediction of the tree and the prediction of the DRL model must be small.

As Table 6.7 shows, Iterative Segmented Regression and Sorting with Variance Reduction achieve greater fidelity on test set than other methods. We recommend Iterative Segmented Regression as a good default method, and Sorting with Variance Reduction as a close second. The null model calculates the mean value of the response variable and uses that as its prediction. The extended version reports high correlations between the outputs of the neural and mimic models: for iterative segmented regression, they are almost always above 0.9 and in many cases above 0.99.

	Ice Hockey		Soccer	
Split methods	Shots	Passes	Shots	Passes
Gaussian Mixture	0.05483	0.04276	0.00698	0.01000
Iterative Segmented Regression	0.01441	**0.00964**	**0.00508**	**0.00997**
Sorting + Variance Reduction	**0.01219**	0.01012	0.00646	0.01092
Sorting + T-test	0.05709	0.06695	0.01223	0.01796
Null Model	0.13924	0.10808	0.13648	0.06151

Table 6.7: Fidelity to Deep Model: RMSE on Test Set

Feature Importance

A basic question for understanding a neural net is which input features most influence its predictions. Given a model tree, we can compute the feature importance as the sum of variance reductions over all splits that use the feature [63]. Table 6.8 and Table 6.9 show the feature importance of the top 10 most relevant features for the action values of shots in ice hockey and soccer respectively, with feature frequency defined as how many times the tree splits on the feature. Time remaining is important for both ice hockey and soccer because the probability of either team scoring another goal decreases quickly when not much time is left. Moreover, time remaining has a stronger influence on ice hockey than soccer because there are generally more goals in ice hockey. At the beginning of a game, the probability of a team scoring in ice hockey is higher than that in soccer. As time goes towards the end, the probability of scoring in ice hockey decreases more quickly than that in soccer. Unsurprisingly, puck/ball to goal distance and action outcomes (i.e. shots being blocked or not) are also among the most relevant features for shots in ice hockey and soccer.

Ice Hockey	FeatureImportance	FeatureFrequency
time remaining (t)	0.0594	248
y coordinate of puck (t)	0.03418	228
x coordinate of puck (t)	0.02646	153
action blocked (t)	0.02016	12
manpower situation (t)	0.01203	14
home (t)	0.00629	1
angle (t)	0.00164	32
time remaining ($t-1$)	0.00072	9
action: reception ($t-1$)	0.00061	5
score differential ($t-1$)	0.00026	23

Table 6.8: Top 10 Features for the Shots for Ice Hockey Players.

Soccer	FeatureImportance	FeatureFrequency
action blocked (t)	0.01524	1
time remaining (t)	0.00711	36
distance to goal (t)	0.00144	31
action: through ball ($t-1$)	0.00079	1
event duration (t)	0.00068	8
time remaining ($t-1$)	0.00059	12
y velocity of ball (t)	0.00036	5
x coordinate of ball (t)	0.00015	28
manpower situation (t)	0.00011	1
action: cross ($t-1$)	0.00011	2

Table 6.9: Top 10 Features for the Shots for Soccer Players.

Rule Extraction

We can extract rules that can be easily interpreted by humans from a model tree. The rules highlight relevant interactions among input features. They also expand on the feature importance by showing how the important features influence the predictions of the neural network.

For shots in ice hockey, Figure 6.8 is a part of a tree to demonstrate how rules can be extracted. First, how good a shot is for the home team is related to which team is taking possession of the puck. In other words, whether the shot is performed by the home team or the away team. By looking at the average Q-values of the corresponding child nodes, we see that it is better for the home team if they take a shot than if the away team takes a shot. If the shot is by the home team, its Q-values are related to the time remaining in the game: with little time left (less than 335 seconds), there is less of

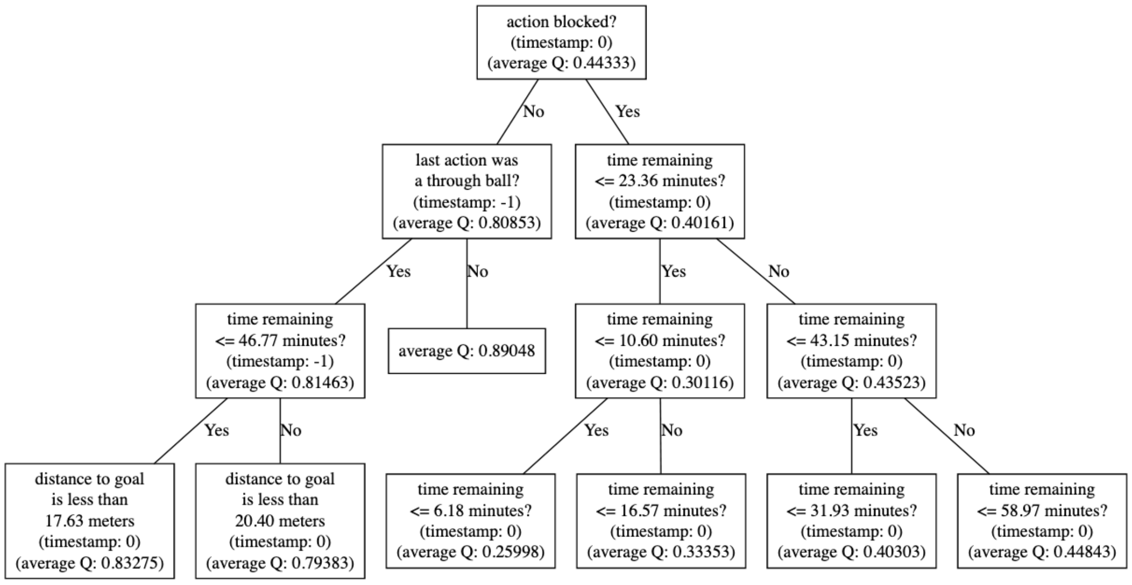

action blocked? (timestamp: 0) (average Q: 0.44333)
No
Yes
last action was a through ball? (timestamp: -1) (average Q: 0.80853)
time remaining <= 23.36 minutes? (timestamp: 0) (average Q: 0.40161)
Yes
No
Yes
No
time remaining <= 46.77 minutes? (timestamp: -1) (average Q: 0.81463)
average Q: 0.89048
time remaining <= 10.60 minutes? (timestamp: 0) (average Q: 0.30116)
time remaining <= 43.15 minutes? (timestamp: 0) (average Q: 0.43523)
Yes
No
Yes
No
Yes
No
distance to goal is less than 17.63 meters (timestamp: 0) (average Q: 0.83275)
distance to goal is less than 20.40 meters (timestamp: 0) (average Q: 0.79383)
time remaining <= 6.18 minutes? (timestamp: 0) (average Q: 0.25998)
time remaining <= 16.57 minutes? (timestamp: 0) (average Q: 0.33353)
time remaining <= 31.93 minutes? (timestamp: 0) (average Q: 0.40303)
time remaining <= 58.97 minutes? (timestamp: 0) (average Q: 0.44843)

Figure 6.7: Model Tree Example With 4 Layers for Action Values of Shots in Soccer

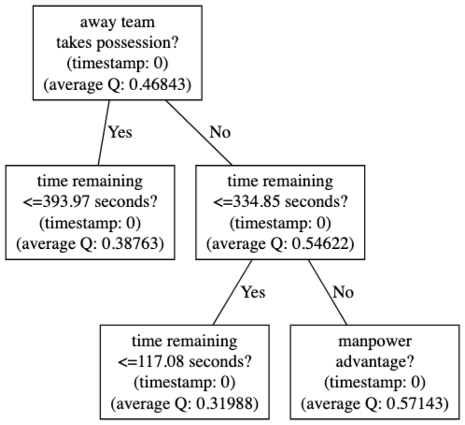

away team takes possession? (timestamp: 0) (average Q: 0.46843)
Yes
No
time remaining <=393.97 seconds? (timestamp: 0) (average Q: 0.38763)
time remaining <=334.85 seconds? (timestamp: 0) (average Q: 0.54622)
Yes
No
time remaining <=117.08 seconds? (timestamp: 0) (average Q: 0.31988)
manpower advantage? (timestamp: 0) (average Q: 0.57143)

Figure 6.8: Rule Example 1 for Action Values of Shots in Ice Hockey. The model tree for ice hockey produces a prediction for the Q-probability that the home team scores the next goal after a shot.

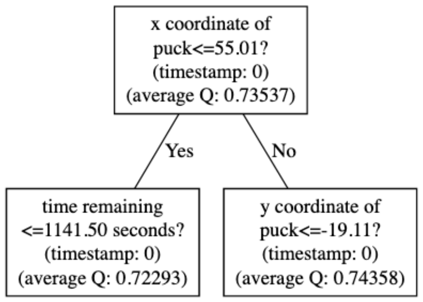

x coordinate of puck<=55.01? (timestamp: 0) (average Q: 0.73537)
Yes
No
time remaining <=1141.50 seconds? (timestamp: 0) (average Q: 0.72293)
y coordinate of puck<=-19.11? (timestamp: 0) (average Q: 0.74358)

Figure 6.9: Rule Example 2 for Action Values of Shots in Ice Hockey

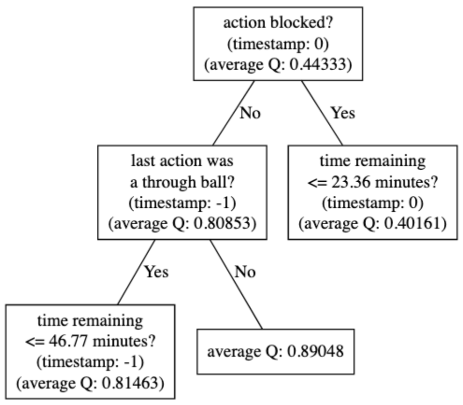


Figure 6.10: Rule Example for Action Values of Shots in Soccer

a chance of any team scoring. However, given sufficient time, the next feature the tree considers is whether the home team has a manpower advantage. Figure 6.9 shows another part of the same tree. It supports the rule that the action values of shots in ice hockey are better when the puck is closer to the net (recall the defensive zone has negative x coordinates and offensive zone has positive x coordinates). If the puck is sufficiently close, then the tree next considers the y-coordinate of the puck location.

For shots in soccer, Figure 6.10 is the top part of Figure 6.7. As in ice hockey, it shows that action values of shots are better when shots are not blocked. Furthermore, it gives an insight that through-ball passes are not the best thing to do to assist a goal. The tree suggests that if a shot is taken right after a through-ball pass, the shot is usually less promising to a goal.

6.4 Mimic Learning for Active Deep Reinforcement Learning

This section introduces two active mimic learning frameworks for DRL models and evaluate the LMUT under three virtual game environments.

6.4.1 Mimic Learning Framework for DRL

Unlike supervised learning, a DRL model is not trained with static input/output data pairs; instead it interacts with the environment by selecting actions to perform and adjusting its policy to maximize the expectation of cumulative reward. We now present two settings to mimic the Q functions in DRL models.

Experience Training

Experience training generates data for batch training, following [9, 24]. To construct a mimic dataset, we record all the observation signals I and actions a during the DRL process. A signal I is a vector of continuous features that represents a state (one-hot representation for discrete features). Then, by inputting them to a mature DRL model, we obtain their corresponding soft output Q and use the entire input/output pairs $\{(\langle I_1, a_1\rangle, \hat{Q}_1(I_1, a_1)), (\langle I_2, a_2\rangle, \hat{Q}_2(I_2, a_2))$ $, ..., (\langle I_T, a_T\rangle, \hat{Q}_T(I_T, a_T))\}$ as the **experience training dataset**. .

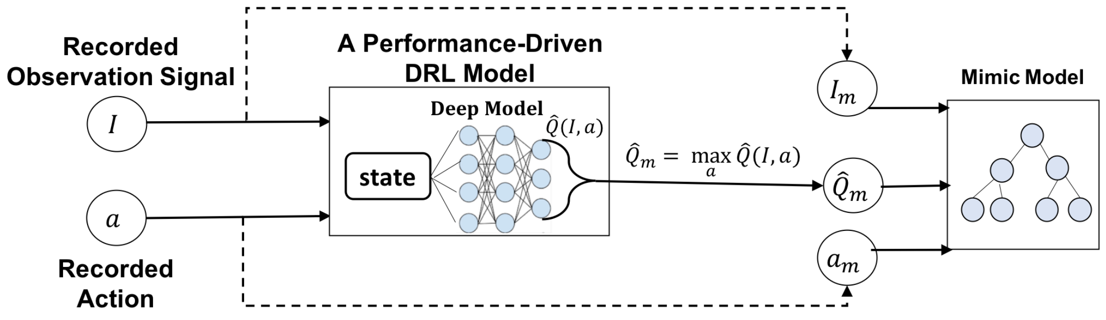

Recorded Observation Signal
I
a
Recorded Action
A Performance-Driven DRL Model
Deep Model
state
$\hat{Q}(I,a)$
$\hat{Q}_m = \max_a \hat{Q}(I,a)$
I_m
$\hat{Q}_m$
a_m
Mimic Model

Figure 6.11: Experience Training Setting

Active Play

Active play generates mimic data by applying a mature DRL model to interact with the environment. Similar to [97], our active learner ℓ has three components: (q, f, I). The first component q is a querying function $q(I)$ that gives the current observed signal I, selects an action a. The querying function controls ℓ's interaction with the environment so it must consider the balance between exploration and exploitation. Our querying function is the ϵ-greedy scheme [71] (ϵ decaying from 1 to 0). The second component f is the deep model that produces Q values: $f : (I, a) \rightarrow range(\hat{Q})$.

As shown in Figure 6.12, the mimic training data is generated in the following steps: *Step 1:* Given a starting observation signal I_t on time step t, we select an action $a_t = q(I_t)$, and obtain a soft output Q value $\hat{Q}_t = f(I_t, a_t)$. *Step 2:* After performing a_t, the environment provides a reward r_t and the next state observation I_{t+1} . We record a labelled **transition** $\mathcal{T}_t = \{I_t, a_t, r_t, I_{t+1}, \hat{Q}_t(I_t, a_t)\}$ where the soft label $\hat{Q}_t(I_t, a_t)$ comes from the well trained DRL model. A transition is the basic observation unit for U-tree learning. *Step 3:* We set I_{t+1} as the next starting observation signal, repeat above steps until we have training data for the active learner ℓ to finish sufficient updates over the mimic model m. This process produces an infinite data stream (transitions $\mathcal{T}$) in sequential order. We use mini-batch online learning, where the learner returns a mimic model after some fixed batch size B of queries.

Compared to Experience Training, Active Play does not require recording data during the training process of DRL models. This is important for the following reasons. (1) Many mimic learners have access only to the trained deep models. (2) Training a DRL model often generates a large amount of data, which requires much memory and is computationally challenging to process. (3) The Experience Training data includes frequent visits to sub-optimal states, which makes it difficult for the mimic learner to obtain an optimal return, as our evaluation illustrates.

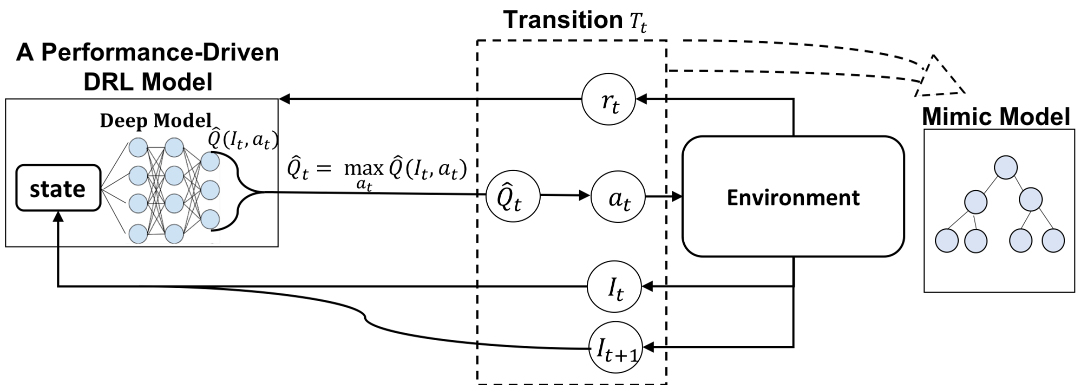


Figure 6.12: Active Play Setting.

6.4.2 Empirical Evaluation

In this section, we evaluate the previously introduced mimic learning model, including CART and LMUT, under the classic reinforcement learning environment (virtual games). Our experiments study the mimic performance of our mimic models by comparing them with other baseline methods. Empirical evaluation measures the mimic match in regression and game playing, under experience training and active play learning.

Evaluation Environment

The evaluation environments include **Mountain Car**, **Cart Pole** and **Flappy Bird**. Our environments are simulated by OpenAI Gym toolkit [16]. Mountain Car and Cart Pole are two benchmark tasks for reinforcement learning [79]. Mountain Car is about accelerating a car to the top of the hill and Cart Pole is about balancing a pole in the upright position. Mountain Car and Cart Pole have a discrete action space and a continuous feature space. Flappy Bird is a mobile game that controls a bird to fly between pipes. Flappy Bird has two discrete actions, and its observation consists of four consecutive images [71]. We follow the Deep Q-Learning (DQN) method to play this game. During the image preprocessing, the input images are first rescaled to 80*80, transferred to gray image and then binary images. With 6,400 features, the state space of Flappy Bird is substantially more complex than that for Cart Pole and Mountain Car.

Baseline Methods

Batch Methods. We fit the input/output training pairs $(\langle I, a\rangle, \hat{Q}(I, a))$ using batch tree learners. A **CART** regression tree [64] (Section 6.3.1) predicts for each leaf node, the mean Q-value over the samples assigned to the leaf. M5 [77] is a tree training algorithm with more generalization ability. It first constructs a piecewise constant tree and then prunes to build a linear regression model for the instances in each leaf node. The WEKA toolkit [40] provides an implementation of M5. We include M5 with Regression-Tree option (**M5-RT**) and M5 tree with Model-Tree option (**M5-MT**) in our baselines. M5-MT builds a linear function on each leaf node, while M5-RT has only a constant value.

On-line Learning Methods. The recent **Fast Incremental Model Tree (FIMT)** [48] method is applied. Similar to M5-MT, it builds a linear model tree, but can perform explicit change detection and informed adaption for evolving data stream. We experiment with a basic version of FIMT and an advanced version with **Adaptive Filters** on leaf nodes (named **FIMT-AF**).

Fidelity: Regression Performance

We evaluate how well our LMUT approximates the soft output ($\hat{Q}$ values) from Q function in a Deep Q-Network (DQN). We report the standard regression metrics **Mean Absolute Error (MAE)**, and **Root Mean Square Error (RMSE)**.

Under the *Experience Training* setting, we compare the performance of CART, M5-RT, M5-MT, FIMT and FIMT-AF with our LMUT. The dataset sizes are 150K transitions for Mountain Car, 70K transitions for Car Pole, and 20K transitions for Flappy Bird. Because of the high dimensionality of the Flappy Bird state space, storing 20K transitions requires 32GB main memory. Given an experience training dataset, we apply 10 fold cross evaluation to train and test our model.

For the *Active Play* setting, batch training algorithms like CART and M5 are not applicable, so we experiment only with online methods, including FIMT, FIMT-AF and LMUT. We first train the mimic models with 30k consecutive transitions from evaluation environments, and evaluate them with another 10k transitions. The result for the three evaluation environments are shown in Table 6.10, Table 6.11 and Table 6.12.

Compared to the other two online learning methods (FIMT and FIMT-AF), LMUT achieves a better fit to the neural net predictions with a much smaller model tree, especially in the active play online setting. This is because both FIMT and FIMT-AF update their model tree continuously after each datum, whereas LMUT fits minibatches of data at each leaf. Neither FIMT nor FIMT-AF terminate on high-dimensional data.[1] So we omit the result of applying FIMT and FIMT-AF in the Flappy Bird environment. Comparing to batch methods, the CART tree model has significantly more leaves than our LMUT, but not better fit to the DQN than M5-RT, M5-MT and LMUT, which suggests overfitting. In the Mountain Car and Flappy Bird environments, model tree batch learning (M5-RT and M5-MT) performs better than LMUT, while LMUT achieves comparable fidelity, and leads in the Cart Pole environment. In conclusion, (1) our LMUT learning algorithm outperforms the state-of-the-art online model tree learner FIMT. (2) Although LMUT is an online learning method, it showed competitive performance to batch methods even in the batch setting.

Learning Curves. We apply consecutive testing [48] to analyze the performance of LMUT learning in more detail. We compute the correlation and testing error of LMUT as more transitions for learning are provided (From 0 to 30k) under the active play setting. To adjust the error scale across different game environments, we use **Relative Absolute Error (RAE)** and **Relative Square Er-**

[1]For example, in the Flappy Bird environment, FIMT takes 29 minutes and 10.8GB main memory to process 10 transitions on a machine using i7-6700HQ CPU.

Method		Evaluation Metrics		
		MAE	RMSE	Leaves
Expe-rience Train-ing	CART	0.284	0.548	1772.4
	M5-RT	0.265	0.366	779.5
	M5-MT	**0.183**	**0.236**	240.3
	FIMT	3.766	5.182	4012.2
	FIMT-AF	2.760	3.978	3916.9
	LMUT	0.467	0.944	620.7
Active Play	FIMT	3.735	5.002	1020.8
	FIMT-AF	2.312	3.704	712.4
	LMUT	0.475	1.015	453.0

Table 6.10: Result of Mountain Car

Method		Evaluation Metrics		
		MAE	RMSE	Leaves
Expe-rience Train-ing	CART	15.973	34.441	55531.4
	M5-RT	25.744	48.763	614.9
	M5-MT	19.062	37.231	155.1
	FIMT	43.454	65.990	6626.1
	FIMT-AF	31.777	50.645	4537.6
	LMUT	**13.825**	**27.404**	658.2
Active Play	FIMT	32.744	62.862	2195.0
	FIMT-AF	28.981	51.592	1488.9
	LMUT	14.230	43.841	416.2

Table 6.11: Result of Cart Pole

Method		Evaluation Metrics		
		MAE	RMSE	Leaves
Expe-rience Train-ing	CART	0.018	0.036	700.3
	M5-RT	0.027	0.041	226.1
	M5-MT	**0.016**	**0.030**	412.6
	LMUT	0.019	0.043	578.5
Active Play	LMUT	0.024	0.050	229.0

Table 6.12: Result of Flappy Bird

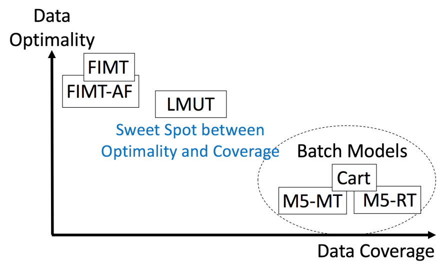


Figure 6.13: Coverage v.s. Optimality

ror (RSE). We repeat the experiment 10 times and plot the shallow graph in Figure 6.14. In the Mountain Car environment, LMUT converges quickly with its performance increasing smoothly in 5k transitions. But for complex environments like Cart Pole and Flappy Bird, the evaluation metrics fluctuate during the learning process but approach the optimum within 30k transitions.

Matching Game Playing Performance

We now evaluate how well a model mimics Q functions in DQN by directly playing the games with them and computing the average reward per episode. (The games in OpenAI Gym toolkit are divided into episodes that start when a game begins and terminate when: (1) the player reaches the goal, (2) fails for a fixed number of times or (3) the game time passes a preset threshold). Specifically, given an input signal I_t, we obtain Q values from the mimic model and select an action $a_t = \max_a Q(I_t, a)$. By executing a_t in the current game environment, we receive a reward r_t and next observation signal I_{t+1}. This process is repeated until a game episode terminates. This experiment uses **Average Reward Per Episodes (ARPE)**, a common evaluation metric that has been applied by both DRL models [71] and OpenAI Gym tookit [16], to evaluate mimic models. In the *Experience Training* setting, the play performance of CART, M5-RT, M5-MT, FIMT, FIMT-AF and our LMUT are evaluated and compared by partial 10-fold cross evaluation, where we select 9 sections of data to train the mimic models and test them by directly playing another 100 games. For the *Active play*, only the online methods FIMT and FIMT-AF are compared, without the Flappy

Bird environment. Here we train the mimic models with 30k transitions, and test them in another 100 games.

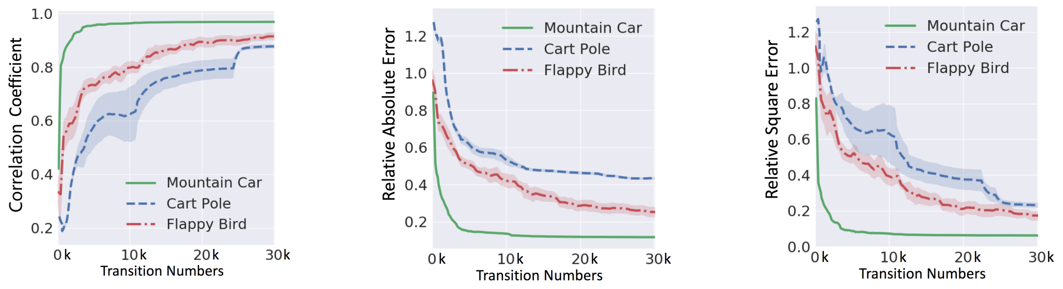


Figure 6.14: Consecutive Testing of LMUT

Table 6.13 shows the results for game playing performance. We first experiment with learning a Continuous U-Tree (CUT) *directly using reinforcement learning* [99] instead of mimic learning. CUT converges slowly with limited performance, especially in the high-dimensional Flappy Bird environment. This shows the difficulty of directly constructing a tree model from the environment.

We find that *among all mimic methods, LMUT achieves the Game Play Performance APER closest to the DQN.* Although the batch learning models have strong fidelity in regression, they do not perform as well in game playing as the DQN. Game playing observation shows that the batch learning models (CART, M5-RT, M5-MT) are likely to choose sub-optimal actions in some key scenarios (e.g., when a pole tilts to one side with high velocity in Cart Pole.). This is because the neural net controller selects many sub-optimal actions at the beginning of training, so the early training experience contains many sub-optimal state-action pairs. The batch models fit the entire training experience equally, while our LMUT fits more closely the most recently generated transitions from a mature controller. More recent transitions tend to correspond to optimal actions. The FIMT algorithms keep adapting to the most recent input only, and fail to build adequate linear models on their leaf nodes. Compared to them, LMUT achieves a sweet spot between optimality and coverage (Figure 6.13).

Interpretability

We discuss how to interpret a DRL model through analyzing the knowledge stored in the transparent tree structure of LMUT: computing feature influence, analyzing the extracted rules and highlighting the super-pixels.

Feature Importance. For Mountain Car and Cart Pole, we report the feature influences in table 6.14. The most important feature for Mountain Car and Cart Pole are Velocity and Pole Angle respectively, which matches the common understanding of the domains. For Flappy Bird the observations are 80*80 pixel images, so LMUT uses pixels as splitting features. Figure 6.15 illustrates the pixels with feature influences $Inf_f > 0.008$ (the mean of all feature influences). Because locating

Model		Game Environment		
		Mountain Car	Cart Pole	Flappy Bird
Deep Model	*DQN*	*-126.43*	*175.52*	*123.42*
Basic Model	CUT	-200.00	20.93	78.51
Experience Training	CART	-157.19	100.52	79.13
	M5-RT	-200.00	65.59	42.14
	M5-MT	-178.72	49.99	78.26
	FIMT	-190.41	42.88	N/A
	FIMT-AF	-197.22	37.25	N/A
	LMUT	-154.57	145.80	97.62
Active Play	FIMT	-189.29	40.54	N/A
	FIMT-AF	-196.86	29.05	N/A
	LMUT	**-149.91**	**147.91**	**103.32**

Table 6.13: Game Playing Performance

the bird is very important. the most influential pixels are located on the top left where the bird is likely to be.

	Feature	Influence
Mountain Car	Velocity	376.86
	Position	171.28
Cart Pole	Pole Angle	30541.54
	Cart Velocity	8087.68
	Cart Position	7171.71
	Pole Velocity At Tip	2953.73

Table 6.14: Feature Influence

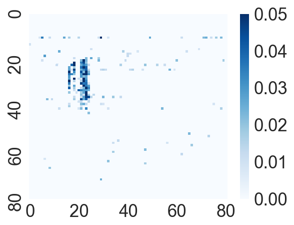


Figure 6.15: Super pixels in Flappy Bird

Rule Extraction. Rule extraction is a common method to distill knowledge from tree models [27, 14]. We extract and analyze rules for the Mountain Car and Cart Pole environment. Figure 6.16 (top) shows three typical examples of extracted rules in *Mountain Car* environment. The rules are presented in the form of partition cells (splitting intervals in LMUT). Each cell contains the range of velocity, position and a Q vector ($\mathbf{Q} = \langle Q_{move\_left}, Q_{no\_push}, Q_{move\_right} \rangle$) representing the average Q-value in the cell. The top left example is a state where the cart is moving toward the left hill with very small velocity. The extracted rule suggests pushing right (Q_{move_right} has the largest value -29.4): the cart is almost stopped on the left, and by pushing right, it can increase its momentum. The top middle example illustrates a state where the car is approaching the top of the left hill with larger left side velocity (compared to the first example). In this case, however, the cart should be pushed left ($Q_{move\_left} = -25.2$ is the largest), in order to store more Gravitational Potential Energy and prepare for the final rush to the target. The rush will lead to the state shown in the top right image, where the cart should be pushed right to reach the target. Notice that the fewer steps are required to reach the target in a given state, the larger its Q-value.

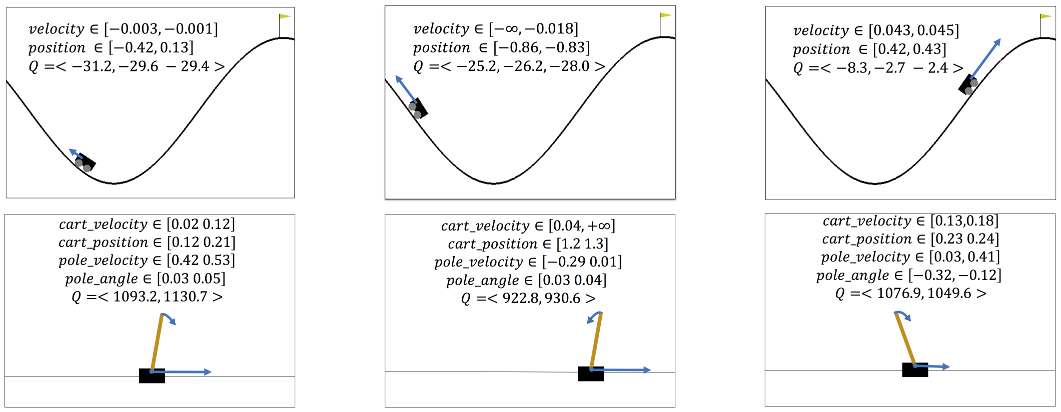

$velocity \in [-0.003, -0.001]$
$position \in [-0.42, 0.13]$
$Q =< -31.2, -29.6 \ - 29.4 >$
$velocity \in [-\infty, -0.018]$
$position \in [-0.86, -0.83]$
$Q =< -25.2, -26.2, -28.0 >$
$velocity \in [0.043, 0.045]$
$position \in [0.42, 0.43]$
$Q =< -8.3, -2.7 \ - 2.4 >$
$cart\_velocity \in [0.02\ 0.12]$
$cart\_position \in [0.12\ 0.21]$
$pole\_velocity \in [0.42\ 0.53]$
$pole\_angle \in [0.03\ 0.05]$
$Q =< 1093.2, 1130.7 >$
$cart\_velocity \in [0.04, +\infty]$
$cart\_position \in [1.2\ 1.3]$
$pole\_velocity \in [-0.29\ 0.01]$
$pole\_angle \in [0.03\ 0.04]$
$Q =< 922.8, 930.6 >$
$cart\_velocity \in [0.13, 0.18]$
$cart\_position \in [0.23\ 0.24]$
$pole\_velocity \in [0.03, 0.41]$
$pole\_angle \in [-0.32, -0.12]$
$Q =< 1076.9, 1049.6 >$

Figure 6.16: Examples of Rule Extraction for Mountain Car and Cart Pole.

Figure 6.16 (bottom) shows three examples of extracted rules in the *Cart Pole* environment. Each cell contains the scope of cart position, cart velocity, pole angle, pole velocity and a Q vector ($\mathbf{Q} = \langle Q_{push\_left}, Q_{push\_right} \rangle$). The key for Cart Pole is using inertia and acceleration to balance the pole. The bottom left example illustrates the tree rule that the cart should be pushed right (i.e., ($Q_{push\_right} > Q_{push\_left}$), if the pole tilts to the right with a velocity less than 0.5. A similar scenario is the second example, where the pole is also tilting to the right but has velocity towards the left. The Q-values correctly indicate that we should push right to maintain this trend; even if the cart is close to the right-side border, which makes its Q values smaller than in the first example. The third example describes a case where a pole tilts to the left with velocity towards the right. The model correctly selects a left push to achieve a left acceleration.

Super-pixel Explanation. In video games, DRL models take the raw pixels from four consecutive images as input. To mimic the deep models, our LMUT also learns on four continuous images and performs splits directly on raw pixels. Deep models for image input can be explained by super-pixels [78]. We highlight the pixels that have feature influence $Inf_f > 0.008$ (the mean of all feature influences) along the splitting path from root to the target partition cell. Figure 6.17 provides two examples of input images with their highlighted pixels at the beginning of game (top) and in the middle of game (bottom). Most splits are made on the first image which reflects the importance of the most recent input. The first image is often used to locate the pipes (obstacles) and the bird, while the remaining three images provide further information about the bird's location and velocity.

6.5 Summary

In this chapter, we introduced the approach of interpreting action-value function with mimic learning. We first examined the performance of learning a traditional regression tree as a mimic model for the action values. The evaluation included computing the feature importance and the partial dependency plots which illustrated the influence of game features on action values. We then proposed

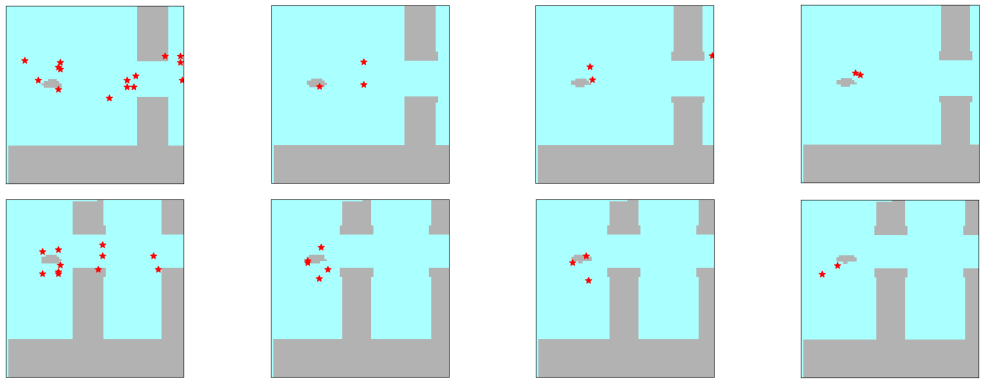


Figure 6.17: Flappy Bird input images with Super-pixels (marked with red stars). The input order of four consecutive images is left to right.

a novel Linear Model U-tree that represented an interpretable model with the expressive power to approximate a Q-value function learned by a deep neural net. A novel on-line LMUT mimic learning algorithm based on stochastic gradient descent is introduced to update the model tree. We examined different splitting methods, and validated the performance of LMUT on the sports data. This chapter also introduced the mimic learning framework for general Reinforcement Learning Environments. The empirical evaluation compared LMUT with five baseline methods on three virtual Reinforcement Learning environments. The LMUT model achieved the clearly best match to the neural network in terms of *its performance on the RL task*. We illustrated the ability of LMUT to extract the knowledge implicit in the neural network model, by (1) computing the influence of features, (2) analyzing the extracted rules, and (3) highlighting the super-pixels.

Chapter 7

Embedding Player Information with Representation Learning

7.1 Introduction

Team sports define a complex interaction structure with a rich action space and a heterogeneous state space [38]. As more and larger event stream datasets for professional sports matches become available, advanced machine learning algorithms have been applied to model the complex dynamics of sports games. These models facilitate many applications with high real-world impact, such as predicting match outcomes [18], assessing the performance of players or teams [63, 29, 84], and recommending tactics to coaches [2]. However, previous models often pool observations of different players without capturing the specific roles and behaviors of each athlete. Neglecting the special characteristics of individual players significantly compromises the model performance for the application tasks [33].

A promising approach to incorporating player information into sports statistics is deep agent representation. However, learning agent representation in a sports game is challenging and existing agent embedding methods are inadequate to model professional players for several reasons: 1) Previous methods commonly learn a representation of agents' *policy* [36, 22, 4, 56, 8, 25]. The policy embedding methods model only how a player acts in a *given* match context, and thus fail to capture the statistical information about which match contexts players are likely to act in, as well as the immediate outcomes (or rewards). 2) Although these policy embeddings can facilitate the construction of an artificial agent in *control* problems, the goal of agent representation in sports analytics is *predicting* game-relevant events using player identity information [33, 2, 87]. 3) Previous agent representation models are often validated in virtual games instead of physical sports. They assume a limited number of agents (usually less than 10) and *sufficient* observations for each of them. In contrast, professional sports leagues often have upwards of one-thousand players, and many bench (backup) players play only a few (less than 20) games in a season. The large number of agents and unbalanced observations make the embedding models unstable and overfit to players with high participation during training, which undermines both the convergence and the performance of underlying policy representations [4].

In this chapter, we propose a novel player representation framework that learns *player representations via player generation.* Conditioning on the current game context, the generation model predicts the distribution of the currently acting player. During this process, we learn a distribution of latent variables as the context-specific prior, allowing a neural encoder to derive an approximate posterior as a *contextualized representation* for the observed player. We train the encoder by maximizing an Evidence Lower Bound (ELBo, Equation (7.9)) that moves the posterior for each player toward the prior mode. This *shrinkage effect* [57] is ideal for player representations, because: 1) It allows information to be transferred between the observations of different players and draws closer the representations of players who share many statistical similarities under a game context. 2) A shrinkage estimator prevents the encoder from overfitting to players with high participation, allowing our representation to generalize to the diversity and the sparsity of player distributions based on the context information.

Following our representation framework, we design a Variational Recurrent Ladder Agent Encoder (VaRLAE) to learn player representations. VaRLAE utilizes a ladder structure [89] where, in each layer, latent variables condition on a context variable and the representations from upper layers, maintaining a causal dependency of latent variables that follows a Markov Game Model [61]. To incorporate play history into player representations, VaRLAE applies a recurrent network to sequential sports data. The ladder hierarchy of latent variables embeds both context and player information, which effectively improves the generative performance and prevents posterior-collapse [89, 43].

We demonstrate the generative performance of our VaRLAE on a massive National Hockey League (NHL) dataset containing over 4.5M events. VaRLAE achieves a leading identification accuracy (>12% for the players with sparse participation) over other deterministic encoders and policy representation models. To study how much the learned player representations improve downstream applications, we assess two major tasks in sports analytics: Predicting expected goals and final match score differences. Empirical results show the improvement in predictive accuracy after incorporating the embeddings generated by VaRLAE as input.

7.2 Related Work

We describe the previous works that are most related to our approach.

Agent Representation. Previous works [39, 55, 56] represented an agent by imitating its policy in the learning-from-demonstration [7] setting. Some recent works [4, 36, 81, 11] extended the behavior modelling to a multi-agent system. Multi-agent representations include modeling interactions between agents and their opponents' behavior for reasoning about a specific goal, such as winning a poker or video game. These interactive policy representations scale poorly to a large number of agents [4, 3]. To identify the agent representations, [36] introduced a triple loss that encourages an exponential distance between deterministic embeddings for different agents. Policy representations have been successful for the design of artificial agents in control problems, such as games or robot control. However, real human players in professional sports cannot be controlled as artificial agents can. For the general sports analytic tasks, the goal of representing players is *predicting* game-relevant events using player identity information [33, 2, 87]. This goal often requires modeling as many aspects of a players' behavior as possible, so we also represent the distribution of states and rewards. A recent work [33] proposed the method of learning a deterministic embedding for the current line-up and showed their embedding can improve the predictive performance in a score-line prediction task. However, their encoder focused only on the current game context without modeling the game history, which limited their embedding performance. To our knowledge, this is the only published work about player embedding in professional sports.

Variational Encoders. Variational encoders apply a set of latent variables $\mathbf{z}$ to embed the observations $\boldsymbol{o}$. The latent variables form a disentangled representation where a change in one dimension corresponds to a change in one factor of variation [12]. Such representations are often interpretable [53, 44] and have been applied to embed information in multiple domains (e.g. robot skills [42] and task environment [106]). The learned representations can significantly facilitate generating many kinds of complicated data, such as images [54, 102] and action trajectories [104, 92]. To learn these representations, the Variational Auto-Encoder (VAE) maximizes an Evidence Lower Bound (ELBo): $\log p(\boldsymbol{o}) \geq -\mathcal{D}_{\text{KL}}(q(\mathbf{z}|\boldsymbol{o})\|p(\mathbf{z})) + \mathbb{E}_{q(\mathbf{z}|\boldsymbol{o})}\Big[\log p(\boldsymbol{o}|\mathbf{z})\Big]$. The traditional VAE design uses a Gaussian encoder and a neural decoder to model the approximate posterior $q(\mathbf{z}|\boldsymbol{o})$ and the likelihood function $p(\boldsymbol{o}|\mathbf{z})$ respectively. The Conditional VAE (CVAE) [102] is an extension that conditions the generation process on some external environment variables. To model sequential data, the Variational Recurrent Neural Network (VRNN) [26] combines VAE with LSTM recurrence. To model dependencies among latent variables, [89] introduces a Ladder VAE (LVAE) that maintains a top-down latent dependency and effectively prevents posterior collapse.

7.3 Player Representation Framework

We introduce the contextual variables for ice hockey and the player representation framework.

7.3.1 Contextual Variables for Ice Hockey Players

We model the ice-hockey games with a Markov Game Model (a more detailed definition is presented in Section 3.1): $G = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \gamma)$. At each time step t, an agent performs an action $a_t \in \mathcal{A}$ at a game state $s_t \in \mathcal{S}$ after receiving an observation $\boldsymbol{o}_t \in \Omega$. This process generates the reward $r_t \sim R(s_t, a_t)$ and the next state $s_{t+1} \sim \mathcal{P}_{a_t}(s_t, s_{t+1})$. For each game, we consider event data of the form $[(\boldsymbol{o}_0, pl_0, a_0, r_0), (o_1, pl_1, a_1, r_1), \ldots, (\boldsymbol{o}_t, pl_t, a_t, r_t), \ldots]$: at time t, after observing environment $\boldsymbol{o}_t$, player pl_t takes a turn (possesses the puck) and chooses an action a_t, which produces a reward r_t denoting whether a goal is scored. The discounting factor γ is set to 1.

The play dynamics for the acting player pl_t can be captured by the following contextual variables: 1) The *game state* s_t describes the game environment where the action is performed. To alleviate the partial observability of observations, a game state includes the game history: $s_t \equiv (\boldsymbol{o}_t, r_{t-1}, a_{t-1}, pl_{t-1}, o_{t-1}, \ldots, o_0)$. We utilize the RNN hidden states $\mathbf{h}_{t-1}$ to capture the game history [41], so $s_t \equiv (\boldsymbol{o}_t, \mathbf{h}_{t-1})$. 2) The *action* a_t records the action of the on-puck player. 3) The *reward* r_t denotes whether a goal is scored after performing a_t. As in general RL, the sequence state-action-reward can be interpreted *causally* in sports: the player makes observations summarized in a state signal s_t, which influences his action a_t; together with the environment, the state and action influence whether the player scores a goal. The corresponding causal graph is $s_t \to a_t \to r_t$.

7.3.2 Player Representation via Player Generation

We introduce our framework of learning player representations through modeling a player generation distribution: $p(pl_t|s_t, a_t, r_t)$. This distribution describes where and what of a player's characteristics: what game states they tend to act in, what their actions are, and what immediate outcomes they achieve. Inspired by previous work [42, 106, 103], we utilize latent variables $\mathbf{z}_t$ as a representation of game context, which can be decoded to the distribution of current acting players:

$$p(pl_t|s_t, a_t, r_t) = \int \mathrm{d}\mathbf{z}_t p(pl_t|\mathbf{z}_t) p(\mathbf{z}_t|s_t, a_t, r_t) \tag{7.1}$$

where, *before* observing the acting player pl_t, $p(\mathbf{z}_t|s_t, a_t, r_t)$ models a **context-aware** prior capturing which players are likely to perform a_t under s_t and receive r_t. The motivation for a contextualized representation is the behavior of sophisticated agents, like professional players, is highly sensitive to context and it is difficult to learn a fixed representation that can adequately describe a player's tendencies under every game context. *After* observing the target player pl_t, we learn an approximate posterior $q(\mathbf{z}_t|s_t, a_t, r_t, pl_t)$ as a **contextualized player representation**, so the complete generative process becomes:

$$p(pl_t|s_t, a_t, r_t) \approx \int \mathrm{d}\mathbf{z}_t p(pl_t|\mathbf{z}_t) q(\mathbf{z}_t|s_t, a_t, r_t, pl_t) \tag{7.2}$$

During training, our ELBo object (Equation 7.9) induces a **shrinkage effect** through minimizing the Kullback–Leibler (KL) Divergence between the posterior representation for each individual player $q(\mathbf{z}_t|s_t, a_t, r_t, pl_t)$ and a context-specific prior $p(\mathbf{z}_t|s_t, a_t, r_t)$. A shrinkage estimator is ideal for learning player representation for team sports because 1) it has strong statistical properties that allow information to be transferred between the observations of different players, or of the same player in different game contexts. The shrinkage effect becomes stronger for players who share many statistical similarities under a game context, which draws their representations closer. This naturally formalizes our intuition that *statistically similar players are assigned similar representations under similar game contexts.* 2) The Shrinkage term also works as a regularizer that prevents the player representations from overfitting to some frequently-present players. Compare to a deterministic auto-encoder, the stochastic shrinkage estimator can generalize to more agents with sparse observations.

In contrast to a policy representation [36, 103, 22] that captures how a player acts in a *given* state, our player representation also models when and where they act, and how successful their actions tend to be. Since our context includes actions, states, and rewards, our learned embeddings reflect how players differ in all three of these dimensions. We refer to this property as the **context-completeness.**

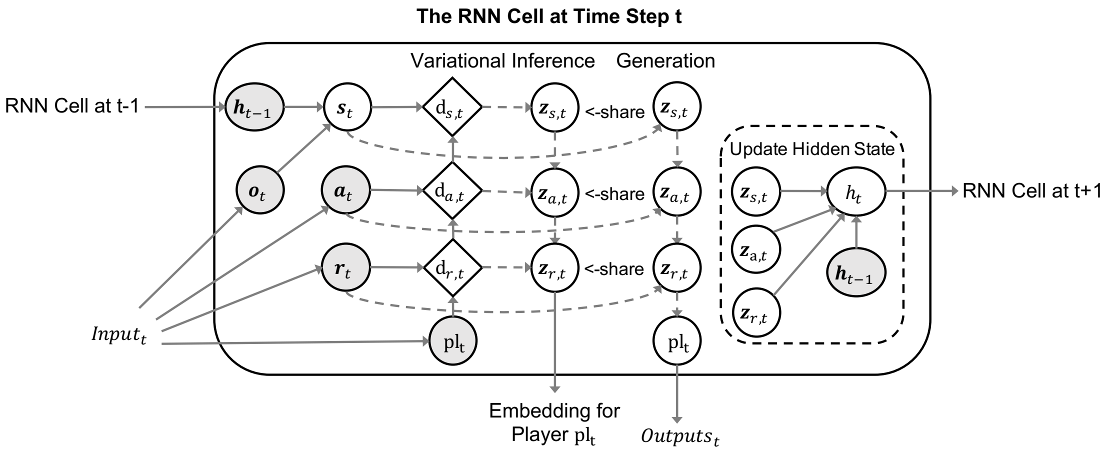


Figure 7.1: Our VaRLAE model includes a conditional Ladder-VAE at every RNN cell. VaRLAE applies a top-down dependency structure ordered as the sports causal relationship (Section 7.3.1). The thick/dash lines denote logical-functions /stochastic-dependence. The shaded nodes are given.

7.4 A Variational Agent Encoder for Learning Agent Representations

This section introduces a VaRLAE (Figure 7.1) model for learning player representations. We describe the generation and inference computations following our representation framework (Section 7.3.2).

Generation. To incorporate match history into the game state s_t, the generation model combines CVAE with a recurrent model following [26]. Previous works [43, 105], however, showed that a high-capacity decoder often causes the problem of *posterior collapse*: the generator computes the distribution of the currently acting player from context variables without relying on latent variables. To make latent variables embed meaningful player information, we introduce a separate latent variable $\mathbf{z}_{c,t}$ for each component of our context, organized in a ladder VAE [89] (Figure 7.1). To utilize the context information (e.g. game state s_t at the top layer), the generator is forced to apply the hierarchical latent variables. This ladder structure models the dependency of these contextualized latent variables ($\mathbf{z}_{s,t} \rightarrow \mathbf{z}_{a,t} \rightarrow \mathbf{z}_{r,t}$) following the causal relationship defined in sports game (Section 7.3.1), which improves its generative performance [89].

Latent Priors. The priors are computed as a function of the game context, rather than a context-independent Gaussian distribution. At each time step, we compute the conditional priors (of player representations) following a Gaussian distribution at each layer:

$$p(\mathbf{z}_{c,t}|c_t, \mathbf{z}_{(+),t}) = \mathcal{N}(\boldsymbol{\mu}^p_{c,t}, \boldsymbol{\sigma}^p_{c,t}) \text{ where } [\boldsymbol{\mu}^p_{c,t}, \boldsymbol{\sigma}^p_{c,t}] = \psi^{prior,c}[\psi^c(c_t, \psi^{\mathbf{z}}(\mathbf{z}_{(+),t}))] \tag{7.3}$$

The context variable $c_t \in \{s_t, a_t, r_t\}$ and $\mathbf{z}_{(+),t}$ denotes latent variables from the upper layer. We omit $\mathbf{z}_{(+),t}$ at the top layer. The neural functions $\psi^{(\cdot)}$ are implemented by a Multi-Layer Perceptron (MLP) with batch normalization at each layer. We use linear and soft-plus activation function to compute the Gaussian parameters μ and σ respectively.

Given the latent variables $\mathbf{z}_{r,t}$ sampled from the context-specific Gaussian prior, our model generates the label of the acting (or on-puck) player as follows:

$$p(pl_t|\mathbf{z}_{r,t}) = Categorical(\theta_{1,t}, \ldots, \theta_{N,t}), \text{ where } \theta_{i,t} := \boldsymbol{\phi}\{\psi^{dec}[\psi^{\mathbf{z}}(\mathbf{z}_{r,t})]\} \tag{7.4}$$

The neural function ψ^z extracts features from latent variable $\mathbf{z}_t$. These features are then sent to another decoder function ψ^{dec}. Given the decoder outputs, we apply a softmax function $\boldsymbol{\phi}$ to generate categorical parameters $\boldsymbol{\theta}_t$. $\theta_{i,t}$ represents the probability of player i acting at time t.

Inference. We apply variational inference to derive an objective function for estimating the parameters of our hierarchical model. Our VaRLAE defines a top-down dependency structure and utilizes the hierarchical priors and approximate posteriors on latent variables to derive an approximate log-likelihood function for the observed data [89, 54].

After observing player pl_t, we first implement a deterministic upward pass to compute the approximate likelihood contributions. Conditioning on reward variable r_t, the bottom layer computes:

$$[\hat{\boldsymbol{\mu}}^q_{r,t}, \hat{\boldsymbol{\sigma}}^q_{r,t}] = \psi^{q,r}(d_{r,t}) \text{ where } d_{r,t} = [pl_t, \psi^{r,t}(r_t)] \tag{7.5}$$

The higher layers take information from lower layers and compute:

$$[\hat{\boldsymbol{\mu}}^q_{c,t}, \hat{\boldsymbol{\sigma}}^q_{c,t}] = \psi^{q,c}(d_{c,t}) \text{ where } d_{c,t} = [d_{(-),t}, \psi^{c,t}(c_t)] \tag{7.6}$$

Here we have $c_t \in \{a_t, s_t\}$ and $d_{(-),t}$ denotes deterministic outputs from a lower layer. Similar to the generator, neural functions $\psi^{(\cdot)}$ are implemented by MLP with batch normalization.

We then implement a stochastic downward pass to recursively compute the approximate posterior. At the top layers, the Gaussian posterior applies the estimated parameters from a deterministic function:

$$q(\mathbf{z}_{s,t}|s_t, pl_t) = \mathcal{N}(\boldsymbol{\mu}^q_{s,t}, \boldsymbol{\sigma}^q_{s,t}) \text{ where } [\boldsymbol{\mu}^q_{s,t}, \boldsymbol{\sigma}^q_{s,t}] = [\hat{\boldsymbol{\mu}}^q_{s,t}, \hat{\boldsymbol{\sigma}}^q_{s,t}] \tag{7.7}$$

At the lower layers, the inference model applies a precision-weighted combination of $(\hat{\boldsymbol{\mu}}^q_{c,t}, \hat{\boldsymbol{\sigma}}^q_{c,t})$ carrying bottom-up information and $(\boldsymbol{\mu}^p_{c,t}, \boldsymbol{\sigma}^p_{c,t})$ from the generative distribution carrying top-down prior information. The approximate posteriors are computed by:

$$q(\mathbf{z}_{c,t}|c_t, \mathbf{z}_{(+),t}, pl_t) = \mathcal{N}(\boldsymbol{\mu}^q_{c,t}, \boldsymbol{\sigma}^q_{c,t}) \text{ where}$$
$$\boldsymbol{\mu}^q_{c,t} = \frac{\hat{\boldsymbol{\mu}}^q_{c,t}(\hat{\boldsymbol{\sigma}}^q_{c,t})^{-2} + \boldsymbol{\mu}^p_{c,t}(\boldsymbol{\sigma}^p_{c,t})^{-2}}{(\hat{\boldsymbol{\sigma}}^q_{c,t})^{-2} + (\boldsymbol{\sigma}^p_{c,t})^{-2}} \text{ and } \boldsymbol{\sigma}^q_{c,t} = \frac{1}{(\hat{\boldsymbol{\sigma}}^q_{c,t})^{-2} + (\boldsymbol{\sigma}^p_{c,t})^{-2}} \tag{7.8}$$

Here we have $c_t \in \{a_t, r_t\}$ and $\mathbf{z}_{(+),t}$ denotes latent variables from the upper layer. This parameterization has a probabilistic motivation by viewing $\hat{\boldsymbol{\mu}}^q_{c,t}$ and $\hat{\boldsymbol{\sigma}}^q_{c,t}$ as the approximate Gaussian likelihood that is combined with a Gaussian prior $\boldsymbol{\mu}^p_{c,t}$ and $\boldsymbol{\sigma}^p_{c,t}$ from the generative distribution. Together these form the approximate posterior distribution $q(\mathbf{z}|\mathbf{z}, c)$ using the same top-down dependency structure for both inference and generation.

Based on [26], the timestep-wise variational lower bound for our model is :

$$\sum_{t=1}^{T}\Big\{ \sum_{\substack{c_t\in\{s_t,\\ a_t,r_t\}}} \Big[-\beta\mathcal{D}_{\text{KL}}[q(\mathbf{z}_{c,t}|c_t, \mathbf{z}_{(+),t}, pl_t)\|p(\mathbf{z}_{c,t}|c_t, \mathbf{z}_{(+),t})]\Big] + \mathbb{E}_{\mathbf{z}_{r,t}\sim q(\mathbf{z}_{r,t}|\cdot)}\Big[\log p(pl_t|\mathbf{z}_{r,t}) - \lambda^{\zeta}\mathcal{L}^{\zeta}(\mathbf{z}_{r,t}, s_t, a_t, r_t)\Big]\Big\} \tag{7.9}$$

where β controls the scale of KL regularization. To mitigate local optima caused by posterior collapse ($\mathcal{D}_{\text{KL}}(\cdot)$ drops to 0) at the initial stage of training [43], we apply a warm-up from deterministic to variational encoder by scaling β from 0 to 1 [89]. The bottom layer latent variables $\mathbf{z}_{r,t}$ absorb context and player information from upper layer and forms a representation: $q(\mathbf{z}_t|s_t, a_t, r_t, pl_t) = q(\mathbf{z}_{r,t}|\cdot)$. This real-valued vector can replace the one-hot player representation and facilitates downstream applications such as predicting expected goals or score differences (see Section 7.5.3). We also add an application loss $\mathcal{L}^{\zeta}$ with a parameter λ^{ζ} to control its scale. This loss combines the gradient of the application models with the embedding inference. It allows co-training the embedding model and the application model which significantly accelerates training and dynamically incorporates player information into different downstream applications.

7.5 Empirical Evaluation

We evaluate the generative performance of the embedding models for player identification and study the usefulness of embeddings for application tasks by incorporating them into task models [74, 1]. Our application tasks include 1) estimating expected goals and 2) predicting the final score differences, which are among the most challenging tasks in sports analytics [68, 33].

7.5.1 Experiment Settings

We introduce our ice-hockey dataset and comparison methods following an ablation design.

Dataset: We utilize an ice hockey dataset constructed by Sportlogiq (see section 3.3.1). The data provides information about **game events** and **player actions** for the entire 2018-2019 National Hockey League (NHL) season, which contains 4,534,017 events, covering 31 teams, 1,196 games and 1,003 players. The dataset consists of events around the puck. Each event includes the identity and action of the player possessing the puck, with time stamps and features of the game context. The dataset records which unique player possesses the puck. We refer to the acting player as the on-puck player. We randomly divide the dataset containing 1,196 games into a training set (80%), a validation set (10%), and a testing set (10%) and implement 5 independent runs. The resulting means and variances are reported.

Comparison models: We employ an ablation design that removes different components from our full VaRLAE system. We first remove the hierarchical dependency of latent variables and train a Conditional Variational Recurrent Neural Network (**CVRNN**) [26]. CVRNN concatenates the context variables (s_t, a_t and r_t) and applies a single layer of latent variables to embed players with variational inference. We then replace the variational encoder with a Conditional Auto-Encoder at each RNN cell (**CAERNN**) that learns a deterministic player representation. The third model is a Conditional Variational Auto-Encoder (**CVAE**) [102] that discards the play history and conditions only on the current observations. Replacing the variational model in CVAE with a Deterministic Encoder yields (**DE**) player embedding [33]. DE is a regressor that directly maps the current observations to the acting player. We also compare our player representation framework to traditional policy embedding. The implementation follows a state-of-the-art Multi-Agent Behavior Encoder (**MA-BE**) [36]. In application tasks (Section 7.5.3), we also compare the options of 1) applying one-hot player identities (**Pids**) directly 2) adding no player information (**N/A**) to application models.

7.5.2 Generative Performance of Embedding Models: On-Puck Player Identification

This experiment studies the generative performance of embedding models: predict which player is acting given the current match context. We compare our VaRLAE model to 6 baselines: 1) identifying player with embedding models: DE, CVAE, MA-BE, CAERNN and CVRNN 2) applying a RNN to model the game history and predict the acting player without a player embedding. The large player space (over 1k players) undermines the performance of encoders that do not utilize the recent play history. To make a fair comparison, we also examine the option of *constraining* the predictions to a group of recently acting players: the current on-puck player (the correct answer) and the players that have possessed the puck in the previous 10 (the trace length of RNNs) steps during testing. To study the identification performance for the players with *sparse participation*, we select the players (a total of 51 players) with fewer than 100 observations (out of a total of 4M events) and report the results.

Table 7.1 shows the results for acting player identification. VaRLAE achieves leading performance on both the prediction accuracy and the log-likelihood. *This lead is most apparent (*$>10\%$*) for the players with sparse participation*, which demonstrates VaRLAE is more robust to the unbalanced player participation. For the variational encoders, they generally perform better than the deterministic encoders. It is because the shrinkage regularizer prevents overfitting to the distribution of popular players in the training dataset. Game history is another crucial aspect for player identification, allowing the RNN models outperform the models with only observation at the cur-

Prediction	Standard		Constraining		Sparse participation	
Method	Accuracy	LL	Accuracy	LL	Accuracy	LL
DE	9.40% ± 3.06E-5	-17.42 ± 2.23E-1	26.14% ± 6.45E-5	-1.60 ± 1.10E-5	2.33% ± 6.95E-3	-22.90 ± 0.02
CVAE	11.94% ± 2.80E-5	-4.90 ± 2.84E-5	28.33% ± 2.96E-4	-1.63 ± 7.77E-7	4.87% ± 8.93E-3	-5.02 ± 0.01
MA-BE	19.74% ± 2.47E-4	-3.08 ± 1.75E-3	51.75% ± 7.58E-5	-1.59 ± 7.92E-6	5.11% ± 2.92E-4	-7.61 ± 1.33
RNN	36.49% ± 2.21E-6	-3.10 ± 2.80E-4	54.21% ± 2.80E-6	-1.55 ± 2.43E-4	6.67% ± 5.03E-4	-6.85 ± 2.15
CAERNN	43.64% ± 1.27E-5	-2.11 ± 1.55E-3	67.43% ± 5.21E-6	-1.38 ± 7.64E-4	11.65% ± 3.06E-3	-3.96 ± 1.20
CVRNN	46.61% ± 9.08E-5	-2.12 ± 2.27E-3	71.76% ± 4.02E-6	**-1.33** ± 2.35E-6	24.30% ± 1.92E-3	-9.67 ± 2.36
VaRLAE	**50.01**% ± 2.56E-6	**-1.76** ± 1.29E-3	**78.54**% ± 3.62E-6	**-1.33** ± 5.16E-4	**36.65**% ± 2.13E-4	**-2.99** ± 0.63

Table 7.1: Results for acting players identification. We report both Accuracy and Log-Likelihood (LL).

rent time step. We also find constraining the candidate players to the group of recent on-puck players significantly improves the identification performance. This phenomenon is most apparent for MA-BE (improves $> 30\%$), which shows that policy embeddings do not distinguish individual players sufficiently.

Embedding Visualization and Case Study: We visualize the generated player embeddings as follows. 1) Randomly select 5 games from the testing set. 2) Compute the contextualized embeddings for the acting player at each event. 3) Visualize the high-dimensional embedding with the unsupervised T-distributed Stochastic Neighbor Embedding (T-SNE) [66]. Figure 7.2 illustrates the scatter plots. The embeddings computed by our VaRLAE (top plots) show a *shrinkage effect*.

Player Positions. VaRLAE embeddings are similar for players in the same position (left column), as they are more likely to perform similarly. Although the position information is hidden during training, our model manages to infer the position information from players' behaviors and assigns them closer embeddings. For a case study on defense men (middle column), we select 5 players because 1) they are well-known players, and 2) in our 5 selected games, they perform a similar number of actions. The plot shows that while defensive players tend to be mapped to similar embeddings, our encoder also learns which contexts distinguish the players from each other.

Action Locations. Action locations are part of the state variable s_t conditioning player embeddings (right column). VaRLAE embeddings are similar for players when they act in the same zone. The similarity is weaker for the CAERNN embeddings (bottom plots). Without shrinkage, CAERNN overfits; it over-emphasizes outliers in the training data, which prevents generalizing to player behavior in the testing data. Figure 7.3 illustrates the influence of *action types* on embeddings.

7.5.3 Predictive Performance On Application Tasks

We validate our VaRLAE model by feeding the generated embeddings to other application models and studying their influence on the model performance in practical application tasks.

Expected Goals Estimation: We validate the player embeddings for the practical task of estimating the Expected Goal (XG). An XG metric is a shot quality model that weights each *shot* by its chance of leading to a goal [68]. We incorporate $\mathbf{z}_t$ from the learned player representations into XG prediction. At time t, we input $s_t, shot_t, \mathbf{z}_t$ to the application model ζ (implemented as a RNN), which is trained to output $p(goal_{t+1}|shot_t, s_t, shot_t, \mathbf{z}_t)$: the probability that a goal is scored after

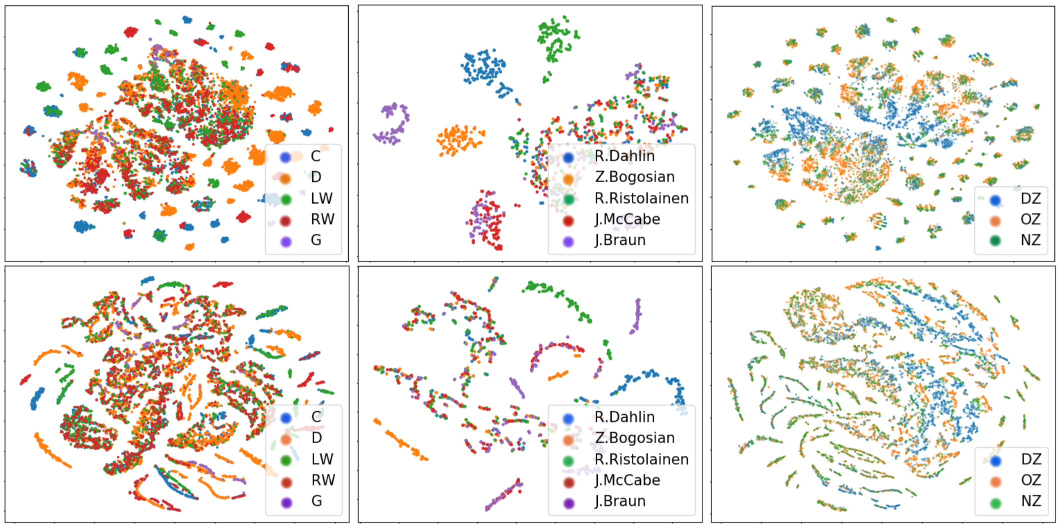


Figure 7.2: Embedding visualization. Each data point corresponds to a player embedding conditioning on the game context at the current time step t. The player embeddings are labelled 1) player positions (on the *left* column, including Center (C), Defense (D), Left-Wing (LW) and Right-Wing (RW) and Goalie (G)) 2) 5 selected defence men (on the *middle* column) and 3) player locations (on the *right* column, including Defence Zone (DZ), Neutral Zone (NZ) and Offensive Zone (OZ)). The embeddings are computed by VaRLAE (*top plots*) and CAERNN (*bottom plots*) respectively.

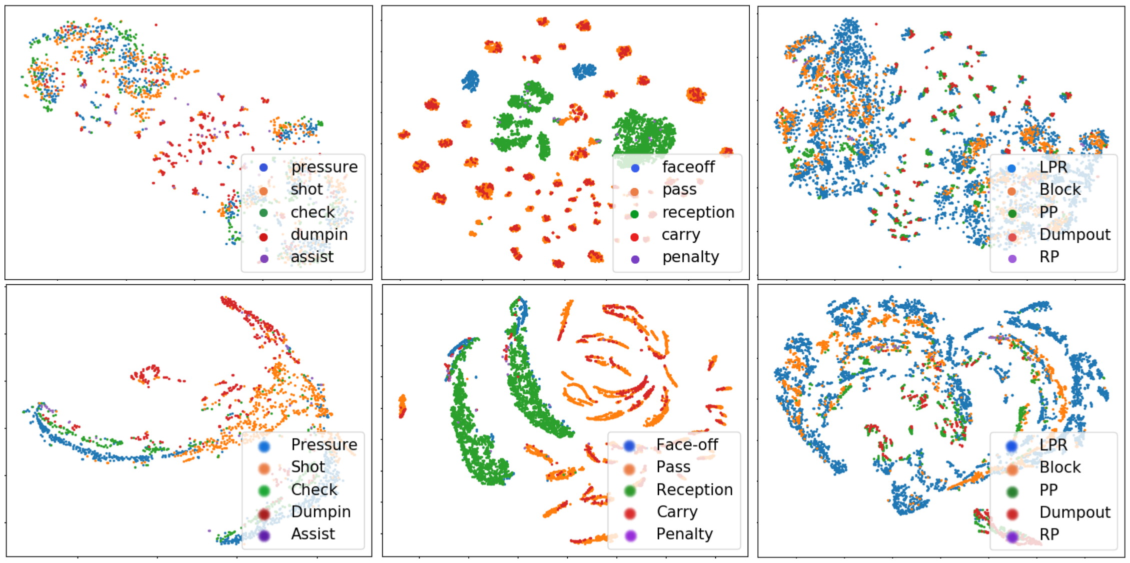


Figure 7.3: Embedding visualization. Each data point corresponds to a player embedding conditioning on the game context at the current time step t. The player embeddings are labelled by the action types. The embeddings are computed by VaRLAE (*top plots*) and CAERNN (*bottom plots*).

the player pl_t makes the shot. Our dataset provides ground-truth labels for whether a given shot led to a goal. Since only a few shot attempts lead to a goal (<3.9%), the training data is highly imbalanced. We, therefore, employ a resampling method [35], so successful and failed shots occur equally often during training.

Player Embedding	Performance			
Method	Precision	Recall	F1-score	AUC
N/A	0.12 ± 1.75E-4	0.79 ± 9.46E-4	0.21 ± 4.26E-4	0.86 ± 3.56E-4
Pids	0.09 ± 1.62E-4	0.62 ± 2.52E-3	0.15 ± 4.20E-4	0.70 ± 1.25E-3
DE	0.30 ± 1.26E-4	0.92 ± 4.21E-4	0.45 ± 1.87E-4	0.96 ± 1.86E-5
CVAE	0.33 ± 5.33E-5	0.95 ± 8.72E-5	0.49 ± 8.29E-5	0.96 ± 1.13E-6
MA-BE	0.35 ± 1.44E-4	0.91 ± 2.30E-4	0.50 ± 1.56E-4	**0.97** ± 1.46E-6
CAERNN	0.29 ± 1.05E-4	0.96 ± 1.56E-4	0.44 ± 1.65E-4	0.95 ± 1.27E-5
CVRNN	**0.40** ± 4.75E-4	0.84 ± 1.81E-4	0.54 ± 2.97E-4	0.96 ± 4.07E-6
VaRLAE	0.37 ± 2.01E-4	**0.98** ± 1.32E-4	**0.54** ± 8.14E-5	0.96 ± 2.23E-6

Table 7.2: Expected goal results applying different player embeddings.

Using the most likely label as the predicted class, Table 7.2 shows the accuracy results on the testing set. Without including any player information (N/A), predictions have large recall but very limited precision, because the model prefers labeling many shots as goals. Adding the pids to the input does not improve precision, which shows that player information is difficult to utilize from a sparse one-hot representation. Applying the embeddings from a player encoder (e.g. DE and CVAE) substantially improves both precision and recall. Among the recurrent models, VaRLAE achieves the highest recall and f1-score with promising precision and AUC, demonstrating the effectiveness of our contextualized representation framework and hierarchical model structure.

Score Difference Prediction: Dynamic Score Difference Prediction (DSDP) is a recently introduced task [33]: predict the final score difference $SD(T)$ under a game context (s_t, a_t) where t runs from 0 to T (game ends). In preliminary experiments, we observed that traditional supervised learning methods suffer a large training variance (especially early in the game when many outcomes are equally likely). To exploit the temporal dependencies between score differences at successive times, we apply reinforcement learning; specifically the temporal difference method Sarsa prediction [94]. Sarsa learns a Q-function for a generic home/away team to estimate the expected cumulative goal scoring: $Q^{team}(s_t, a_t) = \mathbb{E}(\sum_{\tau=t}^{T} g_{team,\tau})$ where $team = Home/Away$ and g_{team}=1 if the team scores at t and 0 otherwise. Given Q-functions, the Predicted Score Difference (PSD) at t is given by $PSD(t) = Q^{Home}(\cdot) - Q^{Away}(\cdot) + SD(t)$. Our *application model* ζ is a DRQNN [41] that computes the Q-functions. The inputs are state s_t, action a_t and the embedding $\mathbf{z}_t$ for the acting player pl_t. For each testing game m and time t, the absolute error is given by $|PSD(t_m) - SD(T_m)|$. For each game time t, Figure 7.4 plots the mean and the variance of the absolute error over all testing games $m = 1, \ldots, M$. The plot shows a larger difference between real and predicted SDs at the beginning of a game, but the mean and variance of the difference become smaller towards the game end. Among the evaluated embedding methods, our *VaRLAE (the black line) manages to generate the player representations that lead to the most accurate predictions.* We also find that the accuracy advantage is strongest towards the early game, especially compared to the N/A and pids. This indicates that an informative player representation significantly alleviates the difficulty of predicting multiple outcomes early in the game. Averaging over game times t defines the game prediction error for each tested game m. Table 7.3 reports the mean and standard deviation for the game prediction error. The VaRLAE embeddings yield the lowest Mean Absolute Error (MAE).

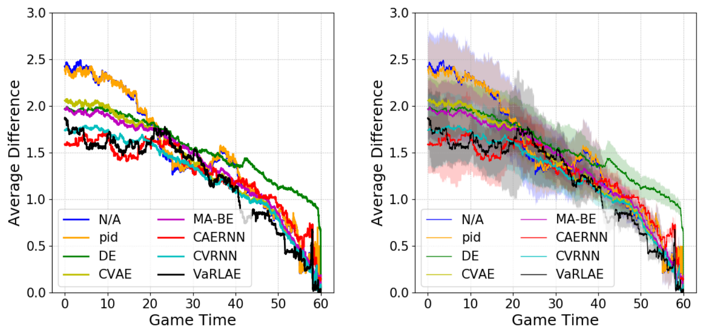
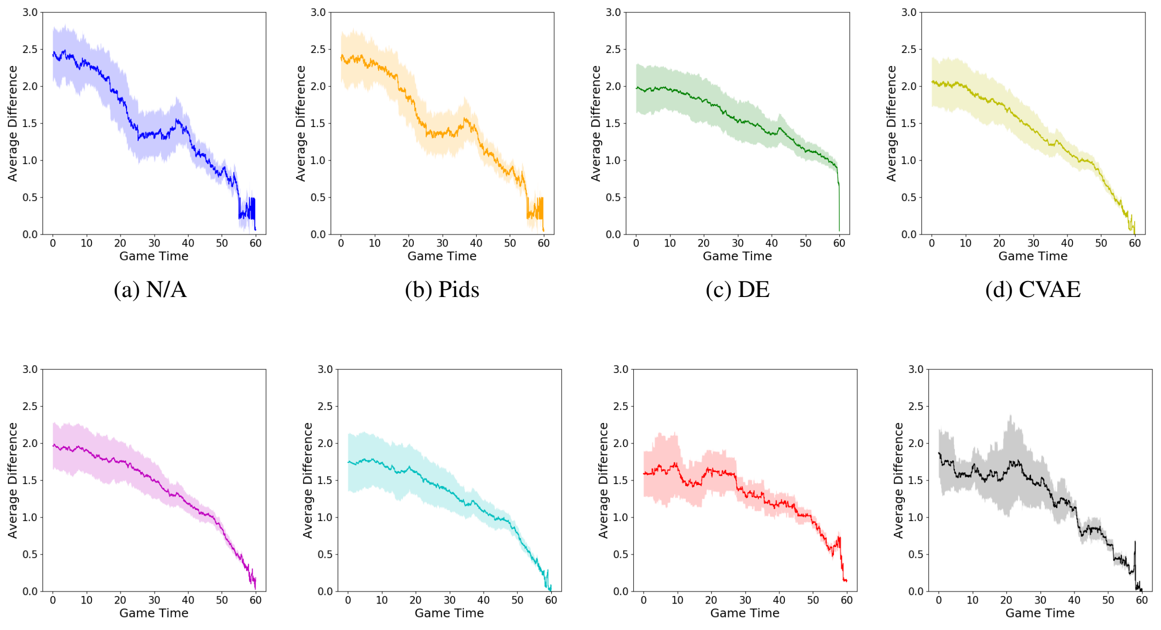


Figure 7.4: Temporal illustrations of the absolute error between predicted score differences and final score differences. The plots report mean (left) and mean±variance of the differences (right). Figure 7.5 shows the separated plots for each method.

Method	MAE
N/A	1.55 ± 0.35
Pid	1.56 ± 0.32
DE	1.48 ± 0.24
CVAE	1.49 ± 0.29
MA-BE	1.45 ± 0.25
CAERNN	1.31 ± 0.23
CVRNN	1.32 ± 0.27
VaRLAE	**1.28** ± 0.29

Table 7.3: The test set game prediction error between predicted and final score differences for the entire game.



(a) N/A (b) Pids (c) DE (d) CVAE

(e) MA-BE (f) CVRNN (g) CAERNN (h) VaRLAE

Figure 7.5: Temporal illustrations of the absolute error between predicted score differences and final score differences. We report mean±variance of the error at each time step for all compared methods.

7.5.4 Posterior Collapse

Figure 7.6 shows the Kullback–Leibler Divergence (KLD) between the posterior and the context-specific prior (for the variational encoders) during training. Among the studied methods, CVAE quickly reduces KLD to a small value (around 0.0005) after training on only a few games, but its performance is less unstable without modeling the game history. CVRNN converges slower and the KLD gradually drops to a very small number (around 3E-05) after training, which indicates the prior can replace the posterior and the decoder can generate the distribution of acting player without the player representation. It is consistent without intuition that a high capacity decoder like RNN can lead to posterior collapse [105]. Our VaRLAE significantly alleviates this problem by applying a hierarchy of latent variables and a deterministic warm-up during training (Section 7.4). The KLD reduces smoothly until it converges a value around 0.03.

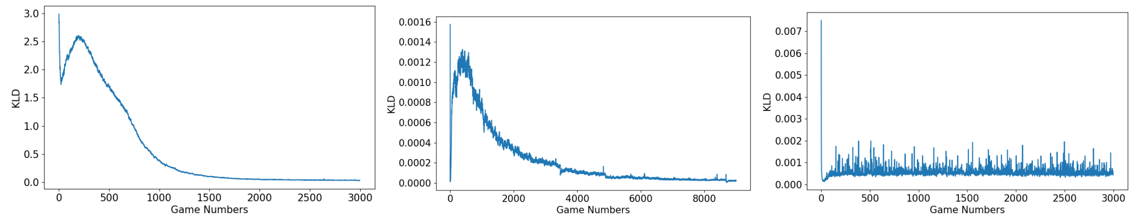


Figure 7.6: The KLD between the posteriors and the priors during training for VaRLAE , CVRNN and CVAE (from left to right).

7.6 Summary

Capturing what team sports players have in common and how they differ is one of the main concerns of sports analytics. In this chapter, we introduced a *player representation via player generation* framework that learns deep contextualized representations for ice hockey players. We described a VaRLAE model tailored towards sports data, based on a Markov Game model representation. The ELBo loss (Equation (7.9)) includes a shrinkage effect such that similar players are mapped to similar representations in similar match contexts. We validated the player representation on two downstream applications that are important in sports analytics: predicting expected goals and final match score differences. While our evaluation focuses on ice hockey, our approach is general and can be applied to other team sports.

Chapter 8

Conclusion, Discussion, and Future Directions

8.1 Summary of the Dissertation

In this dissertation, we mainly focused on learning action values for evaluating players in professional sports data. We first provided an overview of the background about previous player evaluation metrics and then introduced the available datasets for professional sports. Based on the Markov Game model, the Q-function evaluated the player performance by assigning values to their actions. We also introduce a mimic learning approach to interpreting the Q-function and explored the method of modeling the player information through representation learning. The performance of the above methods is validated in two sports environments: Ice-Hockey and Soccer. Experiment results show that the Q-function can assign the values that accurately reflect a player's contribution. Our empirical evaluation also examines the interpretation performance of the mimic learning model as well as the embedding performance of our player representation design. We provide a brief summary of our dissertation in the following three aspects:

- This dissertation described a Markov Game Model to represent the professional sports environment, based on which we defined an action-value Q function with Deep Reinforcement Learning (DRL). The Q-function represented the probability to score the next goal. The spatial and temporal illustration showed that our Q-function assigned accurate values to all the player actions and effectively reflected the match context of evaluated action. We also defined an action impact (the difference between two consecutive Q-values) and measured the player performance by summing up his or her impact over a game season (named by Goal Impact Metric (GIM)). Our experiment studied two popular team sports: Ice Hockey and soccer. The empirical results demonstrated the strong correlation between GIM and other standard success measures (e.g., goal, assist, and point) and the consistency of GIM across a game season.

- We introduced a mimic learning framework for explaining the learned Q-function. This dissertation mainly experimented with two mimic models: a traditional Classification And Regression Tree (CART) and a novel Linear Model U-Tree (LMUT). Compared to CART, LMUT was built for solving reinforcement learning problems and defined a linear model at each leaf node, which significantly improved its generalization ability. Our experiment results showed the leading fidelity of LMUT in both the virtual game environments and professional sports environments. We also described our approach to extracting rules and computing the feature importance from the transparent tree mimic model, which demonstrated the interpretability achieved by our mimic learning framework.
- This dissertation also proposed a player representation framework for incorporating the player information into the action values. To learn a representation that can capture the diversity and sparsity of player distribution in a professional sports game, we referred to the deep representation learning and the Variational Auto-Encoder design. We introduced our Variational recurrent Ladder Agent Encoder (VaRLAE). VaRLAE first learned a conditional prior describing the game context and then derived a contextualized player representation as a posterior after observing the acting player. We studied the performance of VaRLAE under two piratical tasks: estimating the expected goal and predicting the final score difference. Experiment results showed incorporating the embeddings from VaRLAE into application models can improve their performance.

8.2 Discussion

In this section, we discuss the sparsity of goals, model convergence, contextualized representation, shrinkage effect, and limitations of our method.

8.2.1 The Sparsity of Goals

A common method to evaluate players' contribution is computing their influence on goal scoring (increase the scoring probability of their team and prevent their opponent scoring). The scoring events, however, are usually rather sparse during many sports games, such as the Ice Hockey and Soccer studied in this work. This issue is similar to the sparse reward problem in Reinforcement Learning (RL). To resolve this issue, many previous works on sport analytic suggested including other measures like assist, pass, and penalty into the evaluation. It is also similar to the reward shaping technique in RL, which add some handcraft or indirect reward signal to accelerate the model convergence, but this solution often suffers from the mismatch between optimization (include all the shaped reward, e.g., assist and pass) and evaluation (consider only the real reward: scoring). The duality between player evaluation and RL inspires us to investigate a more promising Temporal Difference (TD) solution. TD learns a Q function to spread the reward (scoring) signals to previous events and assigns expected rewards to all players' actions, which significantly alleviate the problem of sparse goal.

8.2.2 Model Convergence

We discuss the convergence of our DRL models (e.g., DP-LSTM in Section 4.4, TTDP-LSTM in Section 5.2). Our DRL models are trained by the on-policy Temporal Difference (TD) method Sarsa. Previous work has proved the convergence of on-policy TD with linear function approximators [98]. However, we apply a non-linear neural network function approximator. It is well-known that on-policy TD with a non-linear function approximator often exhibits unstable convergence in the traditional RL setting, where the action-value Q function is defined as the expected cumulative rewards with unlimited look ahead: $Q(s_t, a_t) = \mathbb{E}[\sum_{\gamma=t}^{\infty} \alpha^{\gamma-t} \cdot r(s_\gamma, a_\gamma)]$. (Here $\alpha \in (0,1)$ is the discount factor and r is the reward function). To alleviate the instability of TD methods, in this dissertation, we constrain the look-ahead to the next goal (rather than the end of game) and remove the discount factor, so $Q(s_t, a_t) = \mathbb{E}[r(s_T, t_T)]$ which is the expected scoring probability of the next goal. (This is valid because $r(s_t, t_t) = 0$ except at goal occurrences T.)

8.2.3 Learning Contextualized Representations.

The behavior of sophisticated agents, like professional players, is highly sensitive to context. It is difficult to learn a fixed representation that can adequately describe a player's tendencies under every game context. Therefore, our VaRLAE first learns a representation for game context (with the context-specific prior) and then asymptotically adjusts the posterior representation for individual players to each context during training. The dynamically contextualized player representation significantly improves the robustness and comprehensiveness of embeddings, with which application models achieve better performance in the downstream tasks.

8.2.4 Shrinkage Effect

In a hierarchical model, shrinkage moves the posterior distribution for each player toward the prior mode. Shrinkage estimators have strong statistical properties because they allow information to be transferred between the observations of different individuals. The shrinkage effect becomes stronger for players who share many statistical similarities under a game context, which draws their representations closer. This naturally formalizes our intuition that *statistically similar players are assigned similar representations under similar game context.* Our neural hierarchical model encourages such a shrinkage effect by minimizing the Kullback–Leibler Divergence (KLD defined in our loss function Equation (7.9)) between posterior for each individual player and a context-specific prior.

8.2.5 Limitations

We show some limitations of our approach and discuss some potential solutions.

Partial observability for the players on pitch. At each time step, the play-by-play dataset records only positions and actions of the player controlling the ball. The information of other players (including both his teammates and opponents), however, also has influence on scoring probabilities, especially for the complex team sport like soccer. To alleviate this issue, our DRL models (DP-LSTM and, TTDP-LSTM) applies a recurrent model to fit the play history and includes the information of previous on-the-ball players, but the model performance still suffers from partial observability—the locations of the off-ball players are not known. A direction for future work is to build a multi-agent reinforcement learning framework for fully observable tracking data. Tracking data, however, are typically proprietary and not easily available to researchers.

The problem of big input data. Our dataset has over 4M events including spatial and temporal features of players. Fitting the entire data requires substantial computational resources. The scalability challenges increase when we include the play history. Therefore is is difficult to utilize standard machine learning packages (such as decision tree, random forest or gradient boosting) that typically assume the entire data can be fit into a single working memory batch. In this dissertation, our DRL models are based on mini batches to address the working memory constraints. A principled alternative is developing on-line learning methods for DRL. For mimic learning, my dissertation included an on-line method for linear model trees (Chapter 6). In future work, we will explore on-line learning methods for learning a Q-function, and evaluate their performance on big sports data.

8.3 Future Directions

In this section, we introduce some future directions that enable better modeling and understanding of the action values function for professional sports games.

8.3.1 Model Game History with Attention Mechanism

In this dissertation, our Q-function modelled play history with LSTM. As a parametric machine learning model, LSTM summarizes the entire history information with a single hidden state and it is often hard to understand which event (in the game history) is the most influential for the current prediction. A promising method to overcome this problem is the attention mechanism, which dynamically learns the importance of previous events with an attention matrix [10]. Such attention models have been commonly applied for the tasks related to Natural Language Processing (NLP) (e.g. Neural machine translation [65] and Information extraction [62]). Although both the NLP and the sports analytic tasks apply sequence data, *a fundamental difference* between them is sports events always flow in *physical time* from start (time step $t = 0$) to end (time step $t = T$) while human language order words according tot the rules of a language, not physics. Indeed, some NLP models applied bi-directional LSTMs to extract information from source sentences. As a consequence, an attention matrix can relate future events to the past outcome. To solve the mismatch between NLP and sports analytics, in future work, we will constrain the attention matrix and only allow it to assign weights only to previous events in the game history.

8.3.2 Information Bottleneck Method for Mimic Learning

Our mimic learning framework facilitated transferring the knowledge from a black-box neural model to another transparent mimic model (e.g. a tree model). A successful mimic model should achieve both the promising approximation performance (for fidelity) and the minimum model complexity (for interpretability) (e.g. a decision tree with a limited depth). In other words, the mimic model of mimic learning should act like an *interpretable minimum sufficient statistic* of the deep model for generating the same predictions. This principle is naturally consistent with the object of Information Bottleneck (IB) method [96] which is about encouraging the bottleneck representations $\mathbf{\Phi}$ to compress the input signal space X by preserving as much of the relevant information about another variable Y as possible. The IB object is given by:

$$\max_{p(\phi|x)} \Big[I(\mathbf{\Phi}; Y) - \lambda I(\mathbf{\Phi}; X) \Big] \tag{8.1}$$

Where I denotes the mutual information and λ is a Lagrange multiplier. Under the mimic learning framework, $\mathbf{\Phi}$, X, and Y denote the mimic models, the space of input data, and soft outputs from deep models respectively. $I(\mathbf{\Phi}; X)$ controls how much the mimic models compress the input data and $I(\mathbf{\Phi}; Y)$ measure how well the mimic models preserve the knowledge from deep models. By varying the parameter λ, one can explore the trade-off between the preserved meaningful information and compression at various resolutions. The IB principle is appealing, since it defines what we mean by a good representation, in terms of the fundamental trade-off between having a concise representation and one with good predictive power [5]. Such an IB principle solves mimic learning as a model compression problem that provides a theoretical foundation for our mimic learning target. In future work, we will apply the IB method to develop an object function for the interpretable mimic learning.

8.3.3 Multi-Agent Embedding for Player Representations

To incorporate the player information into sports statistics, our VaRLAE learned a representation for the current acting player without model his or her interaction with other teammates or opponents. A direction of future work is modeling interactions among players. Given an advanced dataset with full observations of on-court players (the game tracking dataset), the model can learn representations for different lineups rather than individuals. A promising method for learning representations for multiple players is multi-agent embedding. However, previous work commonly learned the embedding for agents in the game environment [4, 36, 81, 11] (e.g., poker, go, and virtual games) rather than the players in the real professional games. Compared to the game environment, a professional league has more (usually over 1K) players and the players in different teams will compete in many matches during a game season. For our future work, we will extend the multi-agent embedding techniques to model the complex interactions among a large number of players in a sports league.

Bibliography

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1638–1649, 2018.

[2] Jim Albert, Mark E Glickman, Tim B Swartz, and Ruud H Koning. *Handbook of Statistical Methods and Analyses in Sports*. CRC Press, 2017.

[3] Stefano V. Albrecht and Subramanian Ramamoorthy. Exploiting causality for selective belief filtering in dynamic bayesian networks. *J. Artif. Intell. Res.*, 55:1135–1178, 2016.

[4] Stefano V. Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artif. Intell.*, 258:66–95, 2018.

[5] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[6] Ajmol Ali. Measuring soccer skill performance: a review. *Scandinavian journal of medicine & science in sports*, 21(2):170–183, 2011.

[7] Brenna D. Argall, Sonia Chernova, Manuela M. Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics Auton. Syst.*, 57(5):469–483, 2009.

[8] Isac Arnekvist, Danica Kragic, and Johannes A. Stork. VPE: variational policy embedding for transfer reinforcement learning. In *International Conference on Robotics and Automation, ICRA*, pages 36–42, 2019.

[9] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[11] Sander C. J. Bakkes, Pieter H. M. Spronck, and Giel van Lankveld. Player behavioural modelling for video games. *Entertain. Comput.*, 3(3):71–79, 2012.

[12] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.

[13] Luke Bornn, Dan Cervone, and Javier Fernandez. Soccer analytics: Unravelling the complexity of “the beautiful game”. *Significance*, 15(3):26–29, 2018.

[14] Olcay Boz. Extracting decision trees from trained neural networks. In *Proceedings SIGKDD*, pages 456–461. ACM, 2002.

[15] Lotte Bransen and Jan Van Haaren. Measuring football players’ on-the-ball contributions from passes during games. In *MLSA-ECML Workshop*, pages 3–15. Springer, 2018.

[16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[17] Joel Brooks, Matthew Kerr, and John Guttag. Developing a data-driven player ranking in soccer using predictive model weights. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 49–55. ACM, 2016.

[18] Rory P Bunker and Fadi Thabtah. A machine learning framework for sport result prediction. *Applied computing and informatics*, 15(1):27–33, 2019.

[19] Samuel Buttrey, Alan Washburn, and Wilson Price. Estimating nhl scoring rates. *Journal of Quantitative Analysis in Sports*, 7(3), 2011.

[20] Dan Cervone, Alexander D’Amour, Luke Bornn, and Kirk Goldsberry. Pointwise: Predicting points and valuing decisions in real time with NBA optical tracking data. In *8th Annual MIT Sloan Sports Analytics Conference, February*, volume 28, 2014.

[21] Daniel Cervone, Alex D’Amour, Luke Bornn, and Kirk Goldsberry. A multiresolution stochastic process model for predicting basketball possession outcomes. *Journal of the American Statistical Association*, 111(514):585–599, 2016.

[22] Yash Chandak, Georgios Theocharous, James Kostas, Scott M. Jordan, and Philip S. Thomas. Learning action representations for reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pages 941–950, 2019.

[23] Probal Chaudhuri, Min-Ching Huang, Wei-Yin Loh, and Ruji Yao. Piecewise-polynomial regression trees. *Statistica Sinica*, pages 143–167, 1994.

[24] Zhengping Che et al. Interpretable deep models for ICU outcome prediction. In *AMIA Annual Symposium Proceedings*, volume 2016, page 371. AMIA, 2016.

[25] Yu Chen, Yingfeng Chen, Yu Yang, Ying Li, Jianwei Yin, and Changjie Fan. Learning action-transferable policy with action embedding. *CoRR*, abs/1909.02291, 2019.

[26] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2980–2988, 2015.

[27] Darren Dancey, Zuhair A Bandar, and David McLean. Logistic model tree extraction from artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(4):794–802, 2007.

[28] Glenn De'Ath. Multivariate regression trees: a new technique for modeling species–environment relationships. *Ecology*, 83(4):1105–1117, 2002.

[29] Tom Decroos, 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, KDD 2019, Anchorage, AK, USA, August 4-8, 2019.*, pages 1851–1861, 2019.

[30] Tom Decroos, Vladimir Dzyuba, Jan Van Haaren, and Jesse Davis. Predicting soccer highlights from spatio-temporal match event streams. In *AAAI 2017*, pages 1302–1308, 2017.

[31] Alin Dobra and Johannes Gehrke. Secret: a scalable linear regression tree algorithm. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 481–487, 2002.

[32] Javier Fernández, FC Barcelona, Luke Bornn, and Dan Cervone. Decomposing the immeasurable sport: A deep learning expected possession value framework for soccer. In *MIT Sloan Sports Analytics Conference*, 2019.

[33] Sujoy Ganguly and Nathan Frank. The problem with win probability. In *Proceedings of the 12th MIT Sloan Sports Analytics Conference. Boston*, 2018.

[34] Tobias Gerstenberg, Tomer Ullman, Max Kleiman-Weiner, David Lagnado, and Josh Tenenbaum. Wins above replacement: Responsibility attributions as counterfactual replacements. In *Proceedings of the Cognitive Science Society*, volume 36, 2014.

[35] Phillip I Good. *Resampling methods*. Springer, 2006.

[36] Aditya Grover, Maruan Al-Shedivat, Jayesh K. Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 1797–1806, 2018.

[37] Joachim Gudmundsson and Michael Horton. Spatio-Temporal Analysis of Team Sports. *ACM Comput. Surv.*, 50(2):22:1–22:34, April 2017.

[38] Joachim Gudmundsson and Michael Horton. Spatio-temporal analysis of team sports. *ACM Comput. Surv.*, 50(2):22:1–22:34, 2017.

[39] Florent Guenter, Micha Hersch, Sylvain Calinon, and Aude Billard. Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*, 21(13):1521–1544, 2007.

[40] Mark Hall, Eibe Frank, et al. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[41] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR, abs/1507.06527*, 2015.

[42] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin A. Riedmiller. Learning an embedding space for transferable robot skills. In *6th International Conference on Learning Representations, ICLR*, 2018.

[43] Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. In *7th International Conference on Learning Representations, ICLR*, 2019.

[44] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR*, 2017.

[45] Ronald A Howard. Dynamic programming and markov processes. 1960.

[46] Moustafa Ibrahim, Srikanth Muralidharan, Zhiwei Deng, Arash Vahdat, and Greg Mori. A hierarchical deep temporal model for group activity recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[47] Todd L Idson and Leo H Kahane. Team effects on compensation: an application to salary determination in the National Hockey League. *Economic Inquiry*, 38(2):345–357, 2000.

[48] Elena Ikonomovska, João Gama, and Sašo Džeroski. Learning model trees from evolving data streams. *Data mining and knowledge discovery*, 23(1):128–168, 2011.

[49] Ulf Johansson, Cecilia Sönströd, and Rikard König. Accurate and interpretable regression trees using oracle coaching. In *Computational Intelligence and Data Mining (CIDM), 2014 IEEE Symposium on*, pages 194–201. IEEE, 2014.

[50] Edward H Kaplan, Kevin Mongeon, and John T Ryan. A Markov model for hockey: Manpower differential and win probability added. *INFOR: Information Systems and Operational Research*, 52(2):39–50, 2014.

[51] Thomas Kempton, Nicholas Kennedy, and Aaron J Coutts. The expected value of possession in professional rugby league match-play. *Journal of sports sciences*, 34(7):645–650, 2016.

[52] Tarak Kharrat, Javier López Pena, and Ian McHale. Plus-minus player ratings for soccer. *arXiv preprint arXiv:1706.04943*, 2017.

[53] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 2654–2663, 2018.

[54] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[55] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems*, pages 849–856, 2008.

[56] Petar Kormushev, Sylvain Calinon, and Darwin G. Caldwell. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148, 2013.

[57] John Kruschke. *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press, 2014.

[58] Niels Landwehr, Mark Hall, and Eibe Frank. Logistic model trees. *Machine learning*, 59(1-2):161–205, 2005.

[59] Austin FS Lee. Optimal sample sizes determined by two–sample welch's t test. *Communications in Statistics-Simulation and Computation*, 21(3):689–696, 1992.

[60] Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[61] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning*, volume 157, pages 157–163, 1994.

[62] Guiliang Liu, Xu Li, Jiakang Wang, Mingming Sun, and Ping Li. Extracting knowledge from web text with monte carlo tree search. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 2585–2591, 2020.

[63] Guiliang Liu and Oliver Schulte. Deep reinforcement learning in ice hockey for context-aware player evaluation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3442–3448. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[64] Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

[65] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421, 2015.

[66] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[67] Brian Macdonald. A regression-based adjusted plus-minus statistic for nhl players. *Journal of Quantitative Analysis in Sports*, 7(3):29, 2011.

[68] Brian Macdonald. An expected goals model for evaluating nhl teams and players. In *Proceedings of the 2012 MIT Sloan Sports Analytics Conference, http://www. sloansportsconference. com*, 2012.

[69] Andrew Kachites McCallum et al. Learning to use selective attention and short-term memory in sequential tasks. In *Proceedings SAB*, volume 4, pages 315–325, 1996.

[70] Ian G McHale, Philip A Scarf, and David E Folker. On the development of a soccer player performance rating system for the english premier league. *Interfaces*, 42(4):339–351, 2012.

[71] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[72] Vito MR Muggeo. Estimating regression models with unknown break-points. *Statistics in medicine*, 22(19):3055–3071, 2003.

[73] Dapo Omidiran. A new look at adjusted plus/minus for basketball analysis. In *MIT Sloan Sports Analytics Conference [online]*, 2011.

[74] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237, 2018.

[75] Stephen Pettigrew. Assessing the offensive productivity of nhl players using in-game win probabilities. In *9th Annual MIT Sloan Sports Analytics Conference*, 2015.

[76] Martin L. Puterman and Jonathan Patrick. Dynamic programming. In *Encyclopedia of Machine Learning and Data Mining*, pages 377–388. Springer Publishing Company, Incorporated, 2017.

[77] Ross J. Quinlan. Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, Singapore, 1992. World Scientific.

[78] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings SIGKDD*, pages 1135–1144. ACM, 2016.

[79] Martin Riedmiller. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *ECML*, pages 317–328. Springer, 2005.

[80] Kurt Routley and Oliver Schulte. A markov game model for valuing player actions in ice hockey. In *Uncertainty in Artificial Intelligence (UAI)*, pages 782–791, 2015.

[81] Jonathan Rubin and Ian D. Watson. Computer poker: A review. *Artif. Intell.*, 175(5-6):958–987, 2011.

[82] Michael Schuckers and James Curro. Total Hockey Rating (THoR): A comprehensive statistical rating of National Hockey League forwards and defensemen based upon all on-ice events. In *MIT sloan Sports Analytics Conference*, 2013.

[83] Michael Schuckers and James Curro. Total hockey rating (thor): A comprehensive statistical rating of national hockey league forwards and defensemen based upon all on-ice events. In *7th Annual MIT Sloan Sports Analytics Conference*, 2013.

[84] Oliver Schulte, Mahmoud Khademi, Sajjad Gholami, Zeyu Zhao, Mehrsan Javan, and Philippe Desaulniers. A markov game model for valuing actions, locations, and team performance in ice hockey. *Data Mining and Knowledge Discovery*, pages 1–23, 2017.

[85] Oliver Schulte, Zeyu Zhao, Mehrsan Javan, and Philippe Desaulniers. Apples-to-apples: Clustering and ranking nhl players using location information and scoring impact. In *Proceedings MIT Sloan Sports Analytics Conference*, 2017.

[86] Steven R Schultze and Christian-Mathias Wellbrock. A weighted plus/minus metric for individual soccer player performance. *Journal of Sports Analytics*, 4(2):121–131, 2018.

[87] Robert P. Schumaker, Osama K. Solieman, and Hsinchun Chen. Research in sports statistics. In *Sports Data Mining*, volume 26 of *Integrated Series in Information Systems*, pages 29–44. Springer US, 2010.

[88] Joseph Sill. Improved nba adjusted +/- using regularization and out-of-sample testing. In *MIT sloan Sports Analytics Conference*, 2010.

[89] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, pages 3738–3746, 2016.

[90] Yuhang Song, Main Xu, Songyang Zhang, and Liangyu Huo. Generalization tower network: A novel deep neural network architecture for multi-task learning. *arXiv preprint arXiv:1710.10036*, 2017.

[91] Jan Struyf and Sašo Džeroski. Constraint based induction of multi-objective regression trees. In *International Workshop on Knowledge Discovery in Inductive Databases*, pages 222–233. Springer, 2005.

[92] Chen Sun, Per Karlsson, Jiajun Wu, Joshua B. Tenenbaum, and Kevin Murphy. Stochastic prediction of multi-agent interactions from partial observations. In *7th International Conference on Learning Representations, ICLR*, 2019.

[93] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge, 1998.

[94] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[95] Tim B Swartz and Adriano Arce. New insights involving the home team advantage. *International Journal of Sports Science & Coaching*, 9(4):681–692, 2014.

[96] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[97] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[98] John N Tsitsiklis and Benjamin Van Roy. Analysis of temporal-diffference learning with function approximation. In *Advances in neural information processing systems*, pages 1075–1081, 1997.

[99] William TB Uther and Manuela M Veloso. Tree based discretization for continuous state space reinforcement learning. In *AAAI/IAAI*, pages 769–774, 1998.

[100] Jan Van Haaren, Guy Van den Broeck, Wannes Meert, and Jesse Davis. Lifted generative learning of markov logic networks. *Machine Learning*, 103(1):27–55, 2016.

[101] Celine Vens and Hendrik Blockeel. A simple regression based heuristic for learning model trees. *Intelligent Data Analysis*, 10(3):215–236, 2006.

[102] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VII*, pages 835–851, 2016.

[103] William Whitney, Rajat Agarwal, Kyunghyun Cho, and Abhinav Gupta. Dynamics-aware embeddings. In *8th International Conference on Learning Representations, ICLR*, 2020.

[104] Eric Zhan, Stephan Zheng, Yisong Yue, Long Sha, and Patrick Lucey. Generating multi-agent trajectories using programmatic weak supervision. In *7th International Conference on Learning Representations, ICLR*, 2019.

[105] Qile Zhu, Wei Bi, Xiaojiang Liu, Xiyao Ma, Xiaolin Li, and Dapeng Oliver Wu. A batch normalized inference network keeps the KL vanishing away. *CoRR*, abs/2004.12585, 2020.

[106] Luisa Zintgraf, Maximilian Igl, Kyriacos Shiarlis, Anuj Mahajan, Katja Hofmann, and Shimon Whiteson. Variational task embeddings for fast adapta-tion in deep reinforcement learning. In *International Conference on Learning Representations Workshop on Structure & Priors in Reinforcement Learning*, 2019.

Appendix A

Appendix

A.1 Proof of Proposition 1

The data record transitions from a state-action-player triple to another, possibly resulting in a non-zero reward (score or point in the context of sports). We denote the number of times such a transition occurs as

$$n_D[s, a, pl, s', a', pl']$$

where the $'$ indicates the successor triple. We freely use this notation for marginal counts as well, for instance

$$n_D[s', a', pl'] = \sum_{s,a,pl} n_D[s, a, pl, s', a', pl']$$

From the section 4.5.2, we have the following equations for the Q-value-above-replacement and the GIM metrics:

$$QAR^i(D) = \sum_{s,a} n_D[s, a, pl' = i]\big(\mathbb{E}_{s',a'}[Q^{team}(s', a'|s, a, pl' = i)] - \mathbb{E}_{s',a'}[Q^{team}(s', a')|s, a]\big) \tag{A.1}$$

$$GIM^i(D) = \sum_{s,a,s',a'} n[s, a, s', a', pl' = i; D] \cdot \Big[Q^{team}(s', a') - \mathbb{E}_{s'_E,a'_E}[Q^{team}(s'_E, a'_E)|s, a]\Big] \tag{A.2}$$

Now we have

$$
\begin{aligned}
GIM^i(D) &\overset{Eq.2}{=} \sum_{s,a}\sum_{s',a'} n_D[s,a,s',a',pl'=i]\Big(Q^{team}(s',a') - \mathbb{E}_{s'_E,a'_E}[Q^{team}(s'_E,a'_E)|s,a]\Big) \\
&= \sum_{s,a} n_D[s,a,pl'=i] \sum_{s',a'} \frac{n_D[s,a,s',a',pl'=i]}{n_D[s,a,pl'=i]} Q^{team}(s',a') \\
&\quad - \sum_{s,a} n_D[s,a,pl'=i]\,\mathbb{E}_{s'_E,a'_E}[Q^{team}(s'_E,a'_E)|s,a] \qquad \text{(A.3)} \\
&= \sum_{s,a} n_D[s,a,pl'=i] E[Q^{team}(s',a'|s,a,pl'=i)] \qquad \text{(A.4)} \\
&\quad - \sum_{s,a} n_D[s,a,pl'=i]\,\mathbb{E}_{s'_E,a'_E}[Q^{team}(s'_E,a'_E)|s,a] \\
&= \sum_{s,a} n_D[s,a,pl'=i]\big(\mathbb{E}_{s'_E,a'_E}[Q^{team}(s'_E,a'_E|s,a,pl'=i)] - \mathbb{E}_{s'_E,a'_E}[Q^{team}(s'_E,a'_E)|s,a]\big) \\
&\overset{Eq.1}{=} QAR^i(D) \qquad \text{(A.5)}
\end{aligned}
$$

Step (A.3) holds because the expectation $E[Q^{team}(s',a'|s,a)]$ depends only on s,a, not on s',a'. Line (A.4) uses the empirical estimate of the expected Q-value $Q^{team}(s',a')]$ given that player i acts next, computed from the maximum likelihood estimates of the transition probabilities:

$$\hat{\sigma}(s',a'|s,a,pl'=i) = \frac{n_D[s,a,s',a',pl'=i]}{n_D[s,a,pl'=i]}$$

The final conclusion (A.5) applies Equation (A.1).

A.2 VAEP Implementation

VAEP probabilities are estimated by building a probabilistic binary classifier for predicting whether a given possession will end in a goal. The VAEP work [29] applied a gradient-boosted tree to fit a dataset of over 11K games, but we were not able to scale the on-line code to our dataset[1]. Instead, we reimplemented the VAEP method, utilizing a neural network with an LSTM layer followed by two fully connected layers (100 and 50 ReLU nodes), and a sigmoid output layer. The trace length of LSTM is 10, corresponding to the VAEP default look-ahead of k = 10. We trained for 10 epochs on the whole dataset, using the same features for VAEP as for Sarsa.

A.3 Ranking for the Premier League

Applying the similar fine-tuning technique mentioned in our main paper, we compute GIMs for the players in Premier League and report the ranking results in the following tables :

[1] `https://github.com/ML-KULeuven/socceraction`

name	team	GIM	Goal	Assist
Riyad Mahrez	Leicester	14.313	12	10
Kevin De Bruyne	Manchester City	14.235	8	16
Christian Eriksen	Tottenham	11.002	10	10
Abdoulaye Doucouré	Watford	10.605	7	3
Joe Allen	Stoke	9.895	2	6
Sadio Mané	Liverpool	9.647	10	7
Jesse Lingard	Manchester United	8.874	8	5
Alex Oxlade-Chamberlain	Liverpool	8.73	3	7
Anthony Knockaert	Brighton	8.524	3	1
Pascal Groß	Brighton	8.089	7	8

Table A.1: 2017-2018 season top-10 Player Impact Scores for players in Premier League season.

name	GIM	Goal
Roberto Firmino	2.632	15
Christian Eriksen	1.94	10
Javier Hernández	1.806	8
Romelu Lukaku	1.759	16
Wilfried Zaha	1.533	9
Juan Mata	1.382	3
Kevin De Bruyne	1.373	8
Eden Hazard	1.333	12
Nacho Monreal	1.282	5
Bernardo Silva	1.255	6

Table A.2: Top-10 soccer players with largest shot impact in 2017-2018 Premier League season.

name	GIM	Assist
Riyad Mahrez	4.532	10
Joe Allen	4.159	6
Kevin De Bruyne	4.063	16
James McArthur	3.653	1
Christian Eriksen	3.639	10
Dwight Gayle	3.456	3
Jordan Ayew	3.387	2
Alex Oxlade-Chamberlain	3.305	7
Kyle Walker	3.206	6
Anthony Knockaert	3.181	1

Table A.3: Top-10 soccer players with largest pass impact in 2017-2018 Premier League season.

A.4 Actions Details

Our soccer dataset records a total of 43 actions including caught-offside, pass_from_fk, cross_from_fk, pass_from_corner, cross_from_corner, cross, throw-in, through-ball, switch-of-play, long-ball, simple-pass, take-on_drible, skill, tackle, interception, aerial-challenge, clearance, ball-recovery, offside-provoked, own-goal, penalty_shot, fk_shot, corner_shot, standard_shot, blocked_shot, save, claim, punch, pick-up, smother, keeper-sweeper, penalty_save, penalising_foul, minor_foul, penalty_obtained, dangerous_foul, dangerous_foul_obtained, run_with_ball, dispossessed, bad-touch, miss, error and goal.

A.5 Generating the Spatial Projections

The spatial projections (Fig.5) are generated as follows. We plot the $\hat{Q}_{Home}(s_\ell, action)$ values for an $\ell = (x, y)$ grid that runs in unit steps from $x \in [-100, 100]$ and $y \in [-100, 100]$. Here s_ℓ represents the state (contains current observation and play history) for a shot at location ℓ, which we compute as follows. 1) For a given location ℓ, we can determine the values of some observed features (e.g., angle between ball and goal). Other features are assigned as their mean value over the entire dataset. This defines an observed feature vector x_ℓ. 2) Given a current action a_t, and previous action a_{t-1}, we compute the average offset vector $d_{\ell_t,\ell_{t-1}}$ between locations of the current and the previous action. Then the previous location is set to . 3) Thus given a current location and an action history, $\ell_t, a_t, a_{t-1}, a_{t-2}, \ldots$, we can compute a complete history of previous locations with average offsets, and therefore a complete state $s_\ell = x_{\ell_t}, a_t, x_{\ell_{t-1}}, a_{t-1}, x_{\ell_{t-2}}, a_{t-2}, \ldots$.

A.6 A Spatial Illustration for the Shot Attempts

We randomly sample 20 games from the training data and show a spatial illustration of shots that happened during these games in Figure A.1. This plot is consistent with our description (section 7.5.3) that the training data is highly imbalanced and only a few shot attempts lead to a goal. The plot also shows that the locations of the successful and the unsuccessful shots are highly overlapped. Without knowing the identity of the acting player, it is hard to determine whether the shot can be made or not.

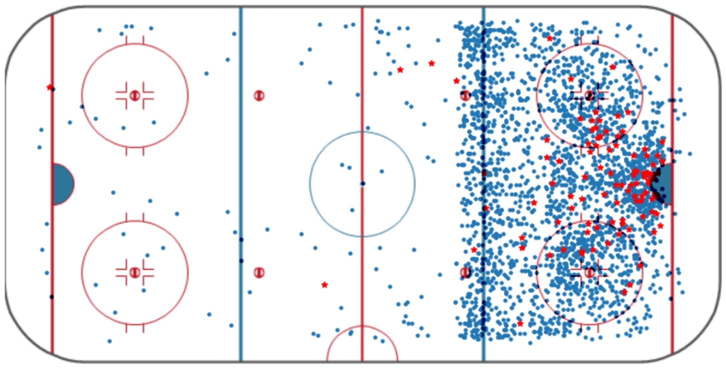


Figure A.1: The spatial illustration of shot attempts on a hockey rink. We apply the adjusted coordinate and the play always flows from left to right. Blue circles represent unsuccessful shots and red stars indicate successful shot.