Automated Classification of Attacking in Football

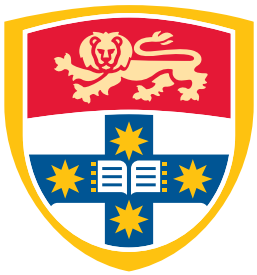
XIANG DAI
SID: 450095834

Supervisor: Dr. Joachim Gudmundsson

This thesis is submitted in partial fulfillment of
the requirements for the degree of
Master of Information Technology and Master of Information Technology Management

School of Information Technologies
The University of Sydney
Australia

23 June 2017



THE UNIVERSITY OF
SYDNEY

Student Plagiarism: Compliance Statement

I certify that:

I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure;

I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to the University commencing proceedings against me for potential student misconduct under Chapter 8 of the University of Sydney By-Law 1999 (as amended);

This Work is substantially my own, and to the extent that any part of this Work is not my own I have indicated that it is not my own by Acknowledging the Source of that part or those parts of the Work.

Name: Xiang Dai

Signature: **Date**:

Abstract

An expert observer of a football (soccer) match can identify attacks, and classify attacks into different categories, however, this kind of manual analysis is highly subjective and labour intensive. In this thesis, we propose two approaches, a neural network approach and a rule-based feature approach, to automatically classify attacks in football.

We investigate the effectiveness of a recurrent neural network classifier to solve the classification problem. We build the classifier on top of raw trajectory data, and examine the impact of different hyperparameters on classification accuracy.

We also present a model that utilizes methods from computational geometry to construct predictor variables, and then apply conventional machine learning classifiers to learn the classification function. Furthermore, we investigate the importance of different features.

Finally, we propose several methods to combine these two approaches, including training classifiers on a large dataset which is generated using random forest classifier, and combining rule-based features to recurrent neural network.

Experimental results show that the rule-based feature approach outperforms the neural network approach in our classifcation task. Random forests classifier achieves the highest accuracy of 83.6% on classifying attacking tactics.

Acknowledgements

Firstly, I would like to thank my supervisor Dr. Joachim Gudmundsson. When I took this project last year, I was only interested in football. Joachim made me realize the amazing world of trajectory research. As well as Joachim, I thank Michael Horton for guding my work, espeically in the machine learning domain. Both are wonderful at inspiring my work throughout the project, and helping me to improve my thesis.

Also I must thank Dr. Sarvnaz Karimi, who was my supervisor during my summer research in Data61, CSIRO. Again I was only interested in social network analysis at the beginning of that three months. Now I am a big fan of deep learning.

Finally, to my family for all the encouragement and support. I would not have been here without your help.

CONTENTS

List of Figures

List of Tables

CHAPTER 1

Introduction

Over the past fifteen years, due to the development of vision-based tracking technologies, a plethora of spatiotemporal data has been generated in the domain of sports analysis. However, the value of these data has not been fully exploited, especially in the term of strategy and tactics analysis.

Nowadays, much of the sophisticated analysis is still performed by expert analysts who are responsible for providing insights to facilitate the decision making by coaches, scouts, and media commentators. The limitation of this manual analysis is that it is highly subjective due to individual's single observation and therefore unreliable. Franks and Miller (Franks and Miller, 1986) pointed out that less than half of all significant match events had been recollected by expert observers after the match. In addition, manual analysis is too labor intensive to cover a large set of matches.

In recent years, researchers from both commercial vendors and academia have taken great interest in solutions to provide automated analysis for sports play, and the number of such systems have increased rapidly (Beetz et al., 2009; Gudmundsson and Wolle, 2014; Perin et al., 2013). These systems may not replace all human analysis within the near future, but it definitely could provide a solid base for further human analysis of complex interactions in sports play.

In this thesis, we present methods for analysing attacks during football matches. Specifically, we utilize spatiotemporal data to classify attacking tactics performed in the match.

1.1 Challenges

The major challenge of our research is that, in continuous sports, a single event or a series of movements of a player is meaningless unless it is contextualized. For example, Figure 1.1 shows the trajectories of the players of a team in one match. It is clear that a visual inspection of this data would be impossible. So our task should be to firstly solve the problem of aligning trajectories of individual players and the

ball. Concretely, we need to assign specific events and movements of players into a high-level team behavior, e.g. associating the movements of several players to one attack instance and labelling two events as the starting and ending point of this attack respectively.

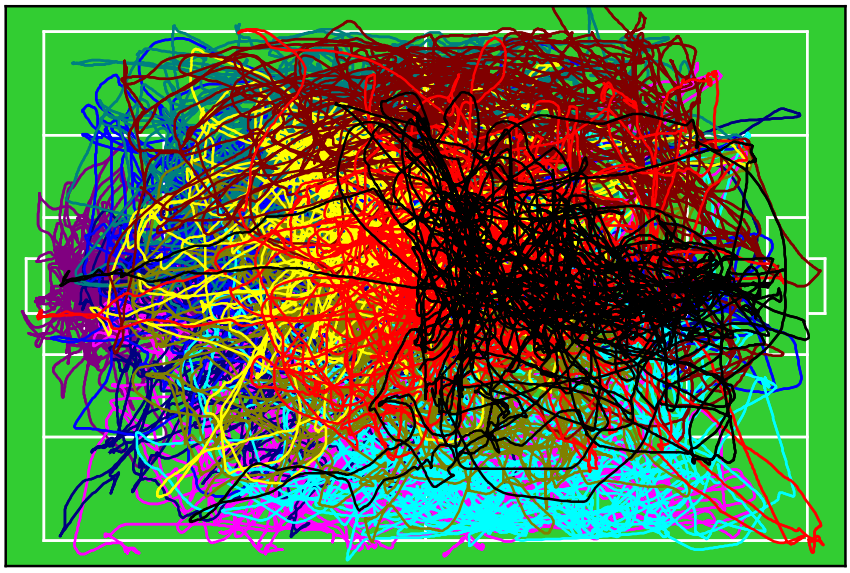


FIGURE 1.1: Trajectories of *Bayern München* players in the first half of the match against *A.S. Roma*. Different colors represent different individuals.

Another challenge in the football analysis field, in contrast to basketball, is due to its low-scoring nature. The number of events which we are interested in is rather low, especially when the available data is distributed unevenly in different categories. For example, in the classification of passing research by Horton et al (Horton et al., 2015), the type of the majority of passes is 'OK', only 193 out of 2932 passes belong to the 'good' category. This class imbalance phenomenon may compound the difficulty of the later classification task. The situation is similar in our research. Only a small number of attacks have forged a clear threat to the opponent's goal and achieved distinguishable tactic outcome. In addition, some particular tactics are seldom performed, or performed only by a small number of teams.

These challenges make most of the research in the sports analysis domain highly dependent on domain-specific knowledge. In terms of our classification task, effectiveness of conventional classification methods greatly depend on sophisticated feature engineering, and the selection of feature functions cannot be independent of understanding football events, and patterns of attacking tactics.

On the other hand, deep learning method has enjoyed great popularity since early 2010s, especially in the speech recognition (Dahl et al., 2012), computer vision (Krizhevsky et al., 2012), and natural language processing (Mikolov et al., 2013) domains. In contrast to conventional machine learning techniques, deep learning is claimed to be able to learn directly from raw natural data, and therefore building a model using deep learning method potentially requires less dependency on domain expertise.

Deep learning allows a complex model with multiple layers. Through the backpropagation algorithm, the model can learn optimal weights in all layers so that the overall output error rate will be minimized (Bishop, 2013). There are two states of the deep learning architecture: convolutional neural networks (CNNs) and recurrent neural networks (RNNs). The former achieved breakthroughs in image classification and some other computer vision tasks, such as automated photo tagging and self-driving, while the latter has gained successes on sequential data such as text (LeCun et al., 2015).

1.2 Contributions

Our main contributions to the sports analysis domain can be summarized as:

- We present two approaches for classifying attacks in football matches: a neural network approach and a rule-based feature approach.
- We investigate the effectiveness of the recurrent neural network classifier to solve the classification problem. In addition, we explore hyperparameter optimisation through grid search to find optimal values of hyperparameters, and analyse the impact of these hyperparameters on the classification accuracy.
- We use methods from comuptational geometry to construct predictor variables, and then apply conventional machine learning classifiers to solve the classification problem. We also investigate the importance of different features.
- We propose two methods to combine deep learning method and conventional classification methods.
  (1) We train the random forest classifier on a small, manually labelled, dataset, then generate a large dataset using the trained classifier, and finally train a neural network on this large dataset.

(2) We combine rule-based features to a neural network as part of the input of its last fully-connected layer, and train the neural network classifier using both raw trajectory data and manually designed features.

1.3 Thesis Structure

In Chpater 2, we summarise the findings of related research in the sports analysis domain. The problem is formally defined in Chapter 3, and the dataset we used is described in Chapter 4. In Chapter 5, the algorithm to identify attacks is briefly presented. In Chapter 6 and 7, we explain the neural network approach and the rule-based feature approach, and report the experimental results. In Chapter 8, we propose two methods to combine these two approaches.

An analysis of the results is provided in Chapter 9, and conclusion and future work is presented in Chapter 10.

CHAPTER 2

Literature Review

The state of art research on sports analysis can be divided into different categories based on different criteria.

In terms of aims of the research, there are two main categories of analysis. The first one is performance analysis, the concrete tasks in this category may include the evaluation of performance of an individual player or a team (Duch et al., 2010; Le et al., 2017), and the evaluation of particular events, such as passing (Horton et al., 2015) and shooting (Chang et al., 2014).

The other direction of analysis is to discover team strategy and team style, which includes discovering frequent attacking patterns (Gudmundsson and Wolle, 2014; Miller et al., 2014), and identifying player roles and team formation (Bialkowski et al., 2014; Lucey et al., 2013).

In terms of technologies used, there are also two large toolkits from which researchers can use methods: network tools inspired from graph algorithms (Duch et al., 2010; Gudmundsson and Wolle, 2014) and machine learning approach (Yue et al., 2014). Recently, deep learning tools are also starting to be exploited (Le et al., 2017).

Gudmundsson and Wolle (Gudmundsson and Wolle, 2014) proposed several algorithms to analyze the quality of passing and to discover frequent patterns of both passing and player movements. In their research, they specified many explicit rules, for example, they specified that passing patterns should be " length at least τ that occurs at least o times", where the parameters can be customized. Because of these very detailed rules, the development of these algorithms requires input from domain experts, which is usually not easy to get.

Duch et al. (Duch et al., 2010) were inspired by social network analysis, and have established an effective framework based on flow centrality to objectively quantify the performance of players and teams. They used the data from the Euro Cup 2008 tournament, and the result showed that their analysis achieved

significant consensus compared to the analysts and spectators. Figure 2.1 is a simple illustration of their results. Their work also had realistic applications in other domains where people concerned about how individual members contribute to the team work, and further how to quantify their contribution.

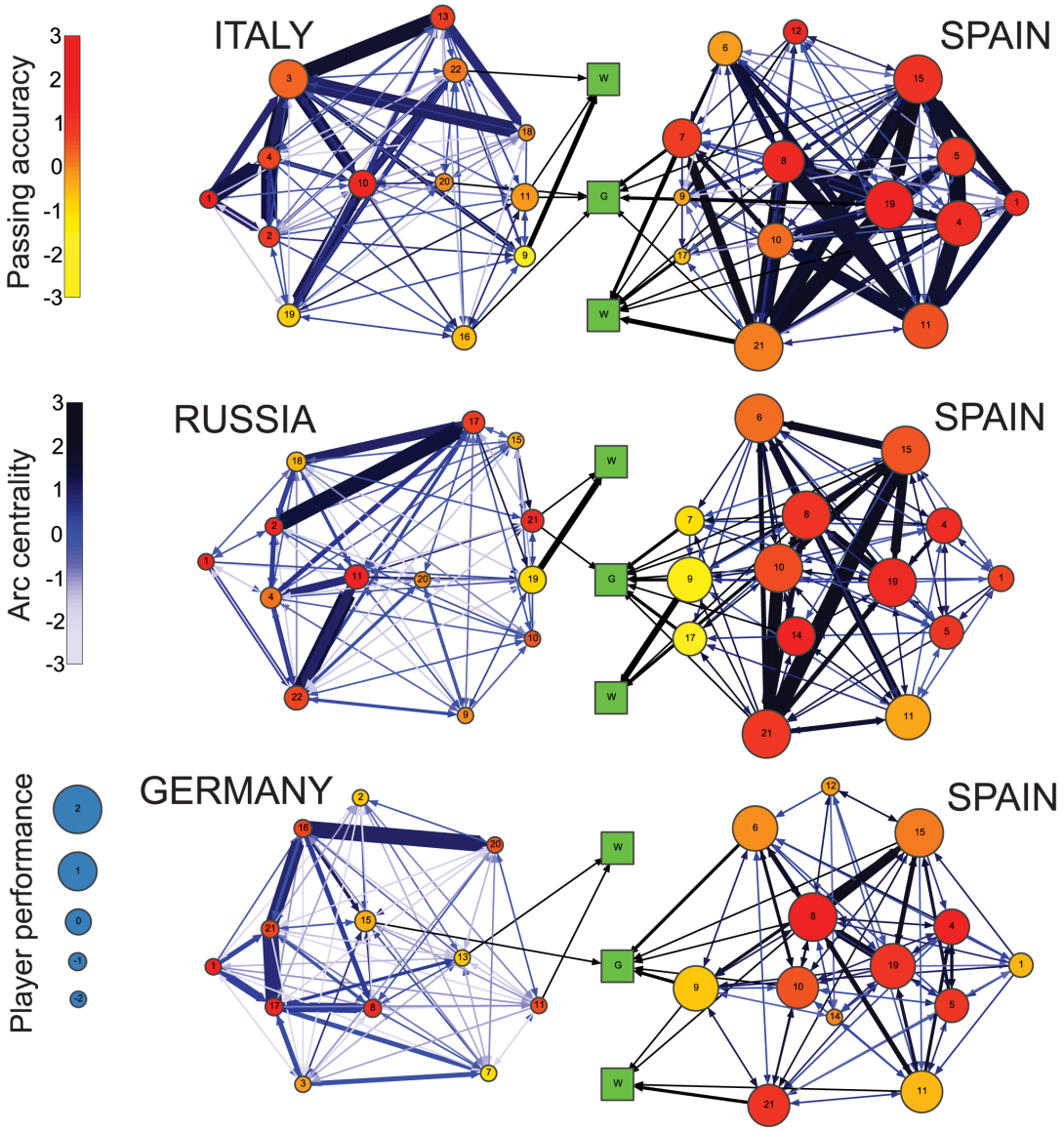


FIGURE 2.1: Duch et al. (Duch et al., 2010) use network tools to evaluate the performance of individual players. Here, each row is the visualization of one match played by the Spanish team. Players are represented by nodes in the graph, and the node location is determined by the player's field position.

Horton et al. (Horton et al., 2015) used feature functions based on methods from computational geometry, and trained classifiers to rate the quality of passes made during a football match. They investigated the effectiveness of the dominant region theory (Taki and Hasegawa, 2000) in their classification task.

In addition, because the dominant region is costly to compute, Horton et al. proposed an approximation algorithm to compute the dominant region. The result showed that the SVM classifier can achieve 85.8% accuracy on classifying passes as three categories: *Good*, *OK*, and *Bad*.

CHAPTER 3

Problem Definition

We formally set up the problem by first defining the notation.

For a given match, we assume it starts at time step 0, and ends at time step λ. Let Γ be a set of trajectories for the ball and each player on the pitch. At time step t, we will use the notation Γ_t^i to represent the location of a player p_i or the ball, if i is 0.

Each player belongs to a team (home team or away team), and each team has maximum 11 players on the pitch. We therefore use $P_H = \{p_1, p_2, \ldots, p_{11}\}$ to denote players of the home team, and $P_A = \{p_{12}, p_{13}, \ldots, p_{22}\}$ the players of the away team. In addition, there is a sequence of events that occured during the match, and we denote it using $\mathbb{E}$. Each event $e_t \in \mathbb{E}$ has a time step t which represents when the event occured.

Now, the problem of identifying and classifying attacks in a football match can be defined as: given a set of trajectories for the ball and players: Γ, and a sequence of events that occured during the match: $\mathbb{E}$, output a sequence of 4-tuples: T, where each tuple $\tau = (\tau_s, \tau_e, \tau_c, \tau_l)$, represents an attack instance.

The four elements in the tuple τ represent the time step when the attack starts (τ_s), the time step when the attack ends (τ_e), the team who is attacking (τ_c), and the attacking tactic (τ_l) used.

To simplify our work, we divide the whole task into two seperate tasks: identifying attacks and classifying attacks.

3.1 Identify Attacks

The attack recognition stage will only take the event logs $\mathbb{E}$ as input, and output a partial result $\tilde{T}$, where the information about the attacking tactic τ_l is missing.

Here, we define the recognition problem as:

$$\tilde{T} = \begin{matrix} (\tau_s^0, \tau_e^0, \tau_c^0) \\ (\tau_s^1, \tau_e^1, \tau_c^1) \\ \vdots \\ (\tau_s^n, \tau_e^n, \tau_c^n) \end{matrix} = f(\mathbb{E}), \tag{3.1}$$

where τ_e^i is always equal to τ_s^{i+1}, τ_c^i is always unequal to τ_c^{i+1}, and τ_e^n is the time step when the last event in $\mathbb{E}$ occured.

3.2 Classify Attacks

Based on the results of previous attack recognition task, we can now re-define our classification task as: for a given attack instance τ^i, the time step when the attack starts (τ_s^i), the time step when the attack ends (τ_e^i), the team (τ_c^i) who is attacking, a set of trajectories: Γ_r, and a sequence of events: $\mathbb{E}_r$, output a label τ_l^i which represents an attacking tactic.

Here, Γ_r and $\mathbb{E}_r$ are the subsets of Γ and $\mathbb{E}$, and they can be defined as:

$$\Gamma_r = \{\Gamma_{\tau_s^i}, \Gamma_{\tau_s^i+1}, \ldots, \Gamma_{\tau_e^i-1}, \Gamma_{\tau_e^i}\}, \tag{3.2}$$

and

$$\mathbb{E}_r = \{e_t : \tau_s^i \leq t \leq \tau_e^i\}. \tag{3.3}$$

Finally, we define the classification problem as:

$$\tau_l^i = \arg\max_l p(l|\tau_s^i, \tau_e^i, \tau_c^i, \Gamma_r, \mathbb{E}_r), \tag{3.4}$$

where $p(l|\tau_s^i, \tau_e^i, \tau_c^i, \Gamma_r, \mathbb{E}_r)$ is the probability that the attack instance τ^i is of class l.

3.3 Output Labels: Attacking Tactics

The output label of the classification task is drawn from a discrete set which corresponds to different attacking tactics. Here, we design our classification scheme based on two considerations:

Utilizing real-word categories: In football tactic vocabulary, there has been a large amount of research in distinguishing different attacking tactics. We want to utilize these real-word categories so that the classification of attacking tactics can be directly used to support decision-making for coaches and scouts.

Exploring only predictable categories: The attacking tactics are not always exlusive. Some tactics are variants of others, and some tactics are components of others. Furthermore, some tactics are rarely performed only by few teams. Therefore it is difficult to collect enough samples of these attacking tactics. We exclude categories of low frequency so that our classifiers can be more reliable.

Based on this trade-off, we pick five basic attacking tactics used as our output labels.

Possessing ball: The aim of this basic tactic is, through controlling the ball as long as possible, to disorganize the opponent's defensive line. Then an open space can be exploited for a free shot. Another implicit objective of this tactic is to wear down the defenders' stamina through mastering the pace of the match.

Long through ball: This simple tactic is usually used by teams with strong and fast forwards. After the ball is passed from a team's own half, the attackers will chase the ball, try to acquire the ball and hopefully get a chance to shoot. This tactic may be not so effective, but after repeating long passes frequently, a mistake by a defender may be anticipated and exploited.

Winger attack: The attackers make use of the width of the field to drag defenders out of their positions, and then through a crossing ball from deep or by cutting inside, the wingers can feed the ball to the striker for a shot. This tactic is sometimes accompanied by switching the wing men that may cause confusion of their markers.

Counter attack: In contrast to *possessing ball*, counter attack is to advance towards the goal as soon as possible after possessing the ball so that the defensive line do not have time to organise themselves. This tactic usually involves lots of players withdrawn into their own half, and only one or two players are left in attacking positions.

Set pieces: Free kicks, throw-ins and corners can threaten the opponent's goal through shooting directly or passing the ball into the opponent's penalty area. These tactics are very efficient for teams who cannot afford many offensive players on the pitch or do not have sufficient capacity to retain possession.

In addition to these tactics, we include two other output labels.

Not an attack: A possession instance is not always evolving into an attack. Some possession instances which have not formed a threat to the opponent's goal, and cannot be classifed as any of previous mentioned five attacking tactics. These instances will be classified into this category.

Not running: Within some time intervals, the match may not be running due to some technical events, such as fouls, player substitutions, and ball out of bounds. We also want to automatically identify these intervals and assign a proper label to them.

CHAPTER 4

Dataset

Our research uses ball trajectory, players trajectories and event logs as the primary input data. The data we use is from 43 matches played by Bayern Munich from 2009 to 2011, which consists of 33 matches at home and 10 matches away.

4.1 Trajectories

Several commercial vendors, such as Amisco [1], who provided us with the dataset, have developed optical tracking systems to capture the movement of players. The optical tracking systems use images taken from several cameras which are installed around the stadium to compute the location of the observed objects (Gudmundsson and Horton, 2017).

The locations of every player on the pitch are recorded ten times per second, and we can directly draw the trajectory of each player during a particular time period from this data. Figure 4.1 is an example of a player trajecory.

The ball is not traced, but the location of the ball can be interpolated from event logs via a simple linear model.

4.2 Event Logs

The event logs are sequences of events during a match. Usually, events are manually annotated by trained analysts, so they can capture rich semantics including time, location, players involved and the type of event.

[1] https://www.stats.com/

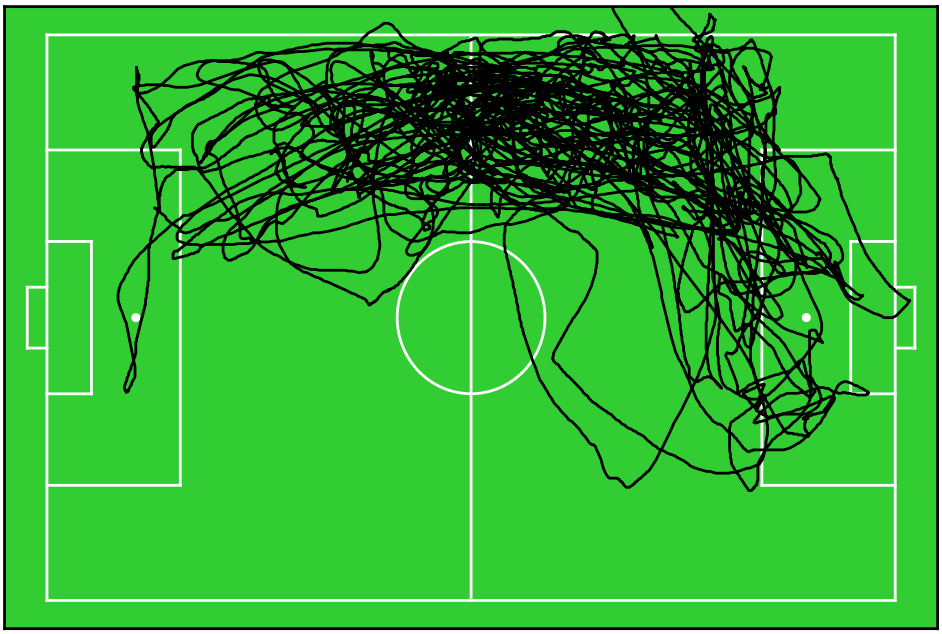


FIGURE 4.1: Trajectory of *Thomas Müller* in the first half of the match against *A.S. Roma*.

In our dataset, there are 37 event types which are listed in Table 4.1. Different event types can play different roles in our classification task. Some event types can immediatly indicate the output label of the following attack instance. For example, when a *foul-penalty* event happens, we can be sure that there will be a *not-running* followed by a *set piece*. Some event types can indicate the start/end of an attack. For example, an *offside* event clearly indicates the end of an attack.

However, we realize that different datasets provided by different vendors may include events of different levels of granularity. For example, *cross* events may be integrated into *pass* events in some datasets. On the other hand, some event types are not useful to our classification task at all, such as goalkeeper events. Therefore, we decide to utilize only a small set of event types so that our research can be adapted to different datasets.

Concretely, we group *pass* events and *cross* events into type *pass*, *shot on target* events and *shot not on target* events into type *shot*, *neutral contact* events, *neutral clearance* events and *crossbar* events into type *neutral*, all referee events into type *referee*, and all other events into a new type *other*.

Player events	Goalkeeper events	Referee events	Technical events
Pass	High deflection gk	Referee	Player out
Cross	Low deflection gk	Red card	Player in
Reception	High catch gk	Yellow card	Out for throw-in
Shot on target	Low catch gk	Foul - ind. free-kick	Out for corner
Shot not on target	Hold of ball gk	Foul - penalty	Out for goal kick
Running with ball	Drop of ball gk	Foul - dir. free-kick	Goal
Neutral contact	High catch drop gk	Another foul	Own goal
Neutral clearance	Low catch drop gk	Offside	Goal post
Clearance	Substitution (gk)		Crossbar
			Interruption game
			End of half

TABLE 4.1: 37 event types in our dataset.

4.3 Labelling

We try to solve the classification task using a supervised machine learning approach, and thus labelled training data is necessary. The labels are created by a human observer watching animated video of trajectories of the ball and players. Figure 4.2 is an example of such an animation at a particular time step.

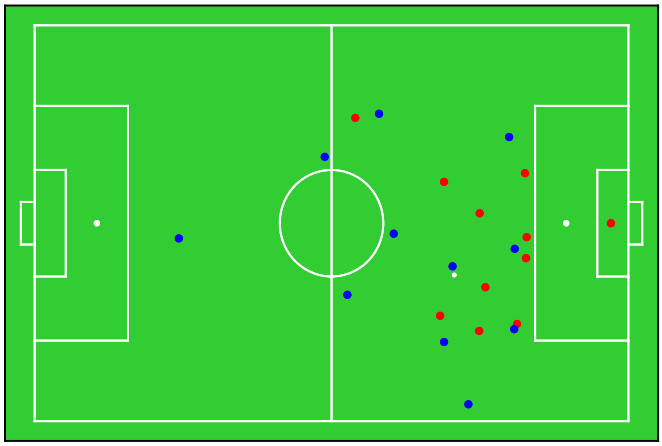


FIGURE 4.2: An animation of trajectories of the ball and players.

For each attack, one or more labels are selected as potential labels and one is selected as the gold standard. The intention is to identify all relevant attacking tactics, and also to prioritize different labels so that the most significant tactic can be identified.

The labelling process to select gold standard is performed using the following rules:

Intention takes priority over actual outcome, when outcome is of *not-an-attack*. In terms of decision-making support, *not-an-attack* instances do not provide insight into a team's attacking tactics. If we can identify an intended attacking tactic rathen than the outcome of a possession, it will be more helpful to understand the team's strategy. For example, the attacking team may intend to threaten the opponent's goal via controlling the ball and exploiting open space. However, in most cases, they may fail to actually threaten the opponent's goal and the possession ends because of a bad pass for example. In this case, if we can identify *possessing ball* is performed rather than *not-an-attack*, it will provide useful insights for decision-makers.

Causal tactics take priority over effect tactics. Several attacking tactics may appear within one attack. For example, an attack may start as a *counter-attack*, and afterwards end with a crossing from winger that implies a *winger-attack*. However, these tactics are not independent. The effect tactic, *winger-attack* in this example, is a result of the causal tactic, i.e. *counter-attack*. Similarly, a *possessing ball* may end with a *winger-attack*, if an open space is found on the side of the pitch when defenders are disrupted. In these cases, causal tactics actually capture more insights into a team's attacking strategy.

The later tactics take priority over the earlier tactics, if these tactics have no causal relationship. An attack, especially of long duration possession, may contain several different stages which have no clear causation. For example, a *counter-attack* may be stopped when defenders quickly organize a solid defensive formation, then the attack may evolve into *possessing ball*. In this case, no clear causation exists between these two tactics, since *possessing ball* is not a direct result of *counter-attack*. Intuitively, the latter tactic is a more explicit threat to the opponent's goal than the previous tactic, so we award the latter tactic with higher priroity.

4.4 Statistics of the Dataset

There are a total of 18,184 samples in our dataset, each of which represents an attack instance. 434 samples are labelled by a human analyst, with one gold standard representing the attacking tactic. Table 4.2 shows the distribution of labels in the labelled dataset.

Category	**Sample size**
Possessing ball	77
Long through ball	20
Winger attack	55
Counter attack	60
Set pieces	68
Not an attack	65
Not running	89
Total	**434**

TABLE 4.2: Distribution of labels in the labelled dataset.

CHAPTER 5

Identify Attacks

A given match can be divided into many continuous time intervals. During each time interval, there will be only one team who possesses the ball or the match is not running.

To identify these possession boundaries, we use a purely rule-based procedure. The algorithm we use will take as input the event logs $\mathbb{E}$, and output a list of triples, each of which, τ^i, consists of the time step τ_s^i when the attack starts, the time step τ_e^i when the attack ends, and the team τ_c^i who is possessing the ball.

The algorithm is shown as follow:

Algorithm 1 Procedure to identify attacks

Require: $\mathbb{E}$ is sorted by time t

1: **function** IDENTIFY_ATTACKS($\mathbb{E}$)
2: $T \leftarrow \varnothing$
3: $pre_c, \tau_s \leftarrow$ None, None
4: **for** $i \leftarrow 1$ to n **do**
5: $e \leftarrow \mathbb{E}[i]$
6: **if** $e.type =$ neutral **then**
7: **continue**
8: **end if**
9: $p \leftarrow e.player$
10: $c \leftarrow$ **Call** Player_to_Team(p)
11: **if** $pre_c =$ None **then**
12: $pre_c \leftarrow c$
13: $\tau_s \leftarrow e.time$
14: **else if** $c \neq pre_c$ **then**
15: $\tau \leftarrow (\tau_s, e.time, pre_c)$
16: **Add** τ to T
17: $\tau_c \leftarrow c$
18: $\tau_s \leftarrow e.time$
19: **else if** $i = n$ **then**
20: $\tau \leftarrow (\tau_s, e.time, pre_c)$
21: **Add** τ to T
22: $\tau_c \leftarrow$ None
23: **end if**
24: **end for**
25: **return** T
26: **end function**

As we mentioned in Section 4.2, there are some events of *neutral* type. It means a player from the opponent team touched the ball, but not successfully controlled the ball. We do not consider this kind of touch to be an interruption of the incumbent possession. Therefore in line 6 of Algorithm 1, we ignore all events of *neutral* type.

In line 10 of Algorithm 1, we call another function to decide which team a player belongs to. The implementation of this function can be different when using different datasets. The dataset we use have a very simple rule that players of *home team* have id range from 1 to 23, while *away team* from 24 to 46. Besides, if the match is not running, we define the team who will re-start the match as τ_c^i.

Using the output of this procedure, we can now move to the classification task: given an attack $(\tau_s^i, \tau_e^i, \tau_c^i)$, trajectories Γ_r, and event logs $\mathbb{E}_r$, output the attacking tactic of this attack instance τ_l^i.

CHAPTER 6

Neural Network Approach

Rather than using a model built with domain-specific feature engineering, we investigated whether deep learning methods can be applied to our classification task. Deep learning allows a complex model with multiple layers, each of which is an abstract representation of previous layer (LeCun et al., 2015). Figure 6.1 is a summary illustration of the neural network we built: one input layer, one recurrent layer, one fully-connected layer, and one output layer.

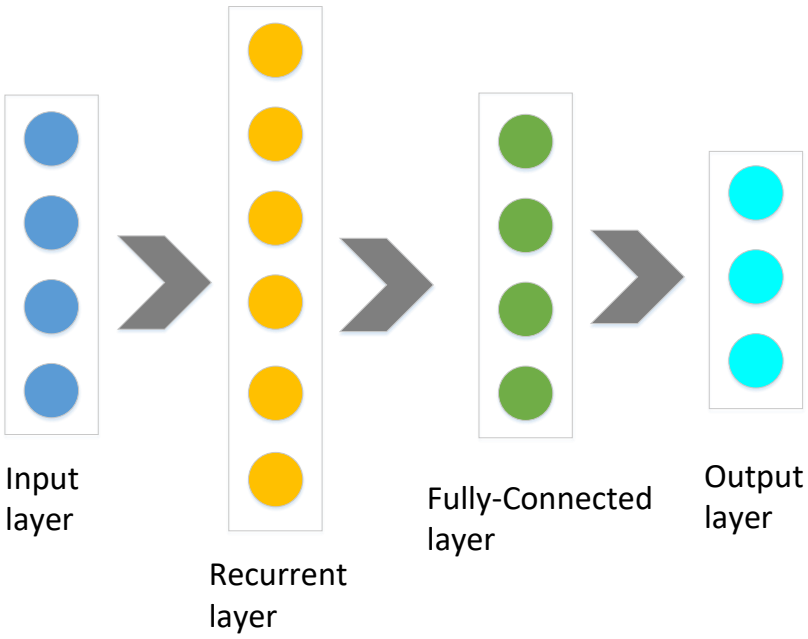
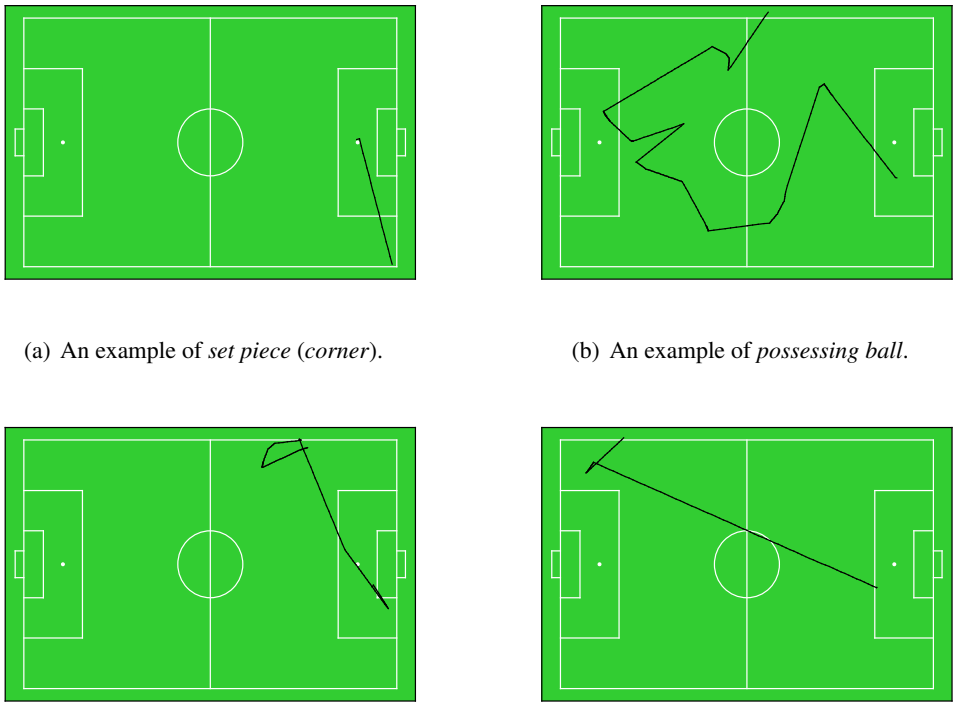


FIGURE 6.1: A high-level illustration of the neural network we built.

6.1 Input Layer

The raw spatiotemporal data related to an attack instance are of two types: ball trajectory and players trajectories. We build two different models using these two types of trajectories.

Ball Trajectory. The movement of the ball can be a very simple indicator of how a team attacks. Several simple examples are shown in Figure 6.2, from which a skilled observer can easily classify the attack type.



(a) An example of *set piece* (*corner*).

(b) An example of *possessing ball*.

(c) An example of *winger-attack*.

(d) An example of *long through ball*.

FIGURE 6.2: Some examples that attacks can be classified using only the movement of the ball.

For each attack instance, we convert the ball trajectory into a $(D \times 2)$ matrix, where D is the length of time that the attack lasts. For the elements a_{ij} of the matrix, i represents the time step and j the x-axis or y-axis of the location.

Player Trajectories. Inspired by the bag-of-words representation from the natural language processing and computer vision domains (Bosch et al., 2007; Tsai, 2012), we use the scatter distribution of attacking players on the field as the input layer.

A fundamental assumption of this representation is that the role of a player during the match is dynamic and can be inferred based on the distribution of positions of the player and the other players of his team. For example, a player who starts as a *center-back* can be considered a *centre forward* if he enters the opponent's penalty area, is involved in an attack, and takes a shot. In other words, assignment of players to nominal positions can be discarded and the team that possesses the ball in a time step can be represented as a **bag of players** described next.

We divide the field into 4×6 regions and assign an index to each region, as illustrated in Figure 6.3. Then, for each time step in the attack instance, a vector in $\mathbb{R}^d$ is used to represent the distribution of the attacking players. Here $d = 24$, is the number of regions we divide the field into. The k-th element of the vector is the number of players occupying the kth region at that time step.

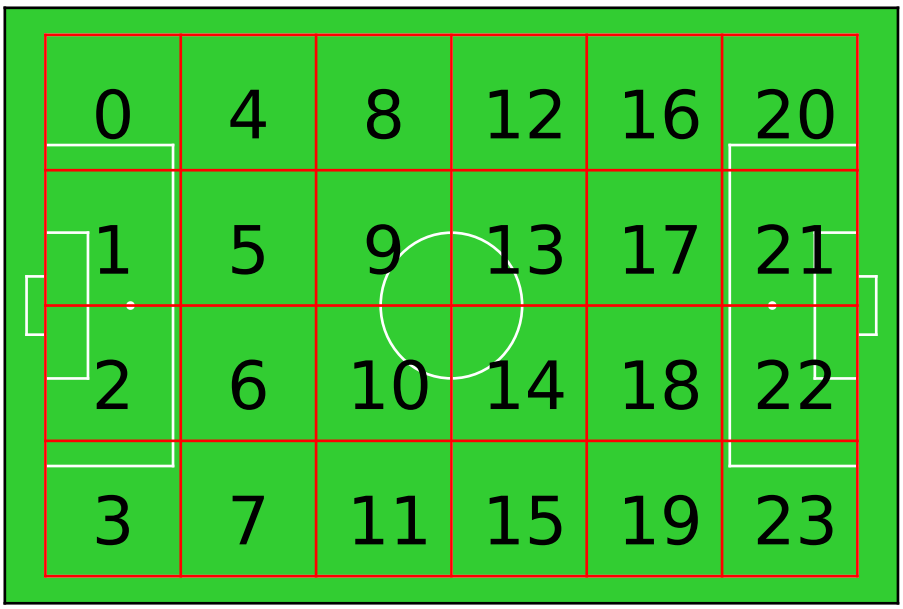
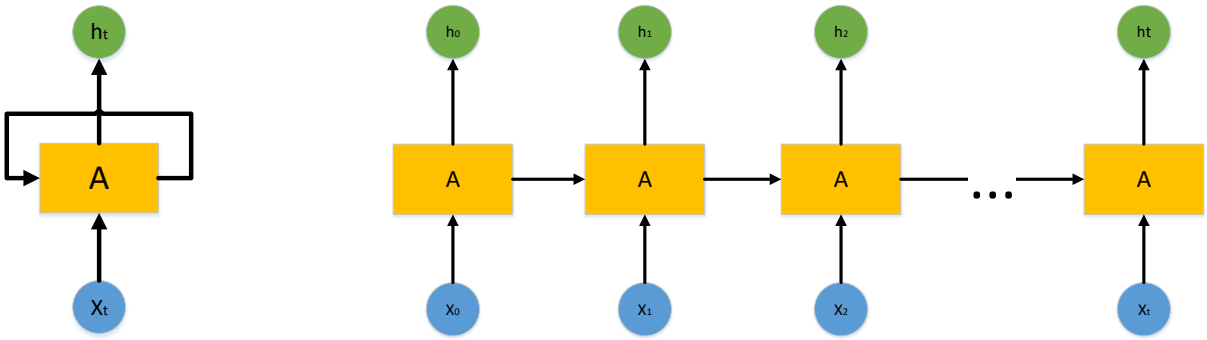


FIGURE 6.3: The field is divided into 4×6 regions, and an index is assigned to each region.

6.2 Recurrent Layer

Recurrent layer takes the output of the computational unit (neuron) as the input to itself. Intuitively, it is utilizing learned information for a sequential task. Figure 6.4 is an illustration showing that a unit with a loop can be equivalent to a chain of repeating units.



(a) A unit with a loop.

(b) A chain of repeating units.

FIGURE 6.4: A unit with a loop is equivalent to a chain of repeating units. Here, x_t is the neuron's input at time step t, and h_t is its output.

The computation in each unit at a time step t can be defined as:

$$h_t = f(W^h h_{t-1} + W^x x_t + b), \tag{6.1}$$

where x_t and h_t are the input and output of the unit, h_{t-1} is the learned information passed from last time step, and f is a non-linear function (e.g. $tanh()$).

In theory, this architecture might be helpful since it can utilise previous information. For example, when we try to understand a specific video frame of a football match, it would be better for us to have seen several previous frames and to keep that information in mind. However, in practice, when the distance between the current task and past context grows, it becomes difficult for the standard RNN model to learn long-distance correlations in a sequence (Tai et al., 2015).

6.2.1 Long-Short-Term-Memories (LSTMs)

LSTMs is an RNN variant proposed by Hochreiter and Schmidhuber to solve the long-term dependencies problem (Hochreiter and Schmidhuber, 1997). By introducing a **memory cell**, LSTMs allow a more complex computation than the standard RNN model. Figure 6.5 is an illustration of LSTMs, where C_t

represents the state of the memory cell at time step t. The difference between h_t and C_t is that C_t is the internal state which can only be used at the next time step within the neuron, while h_t is the external state which also can be used by the next layer.

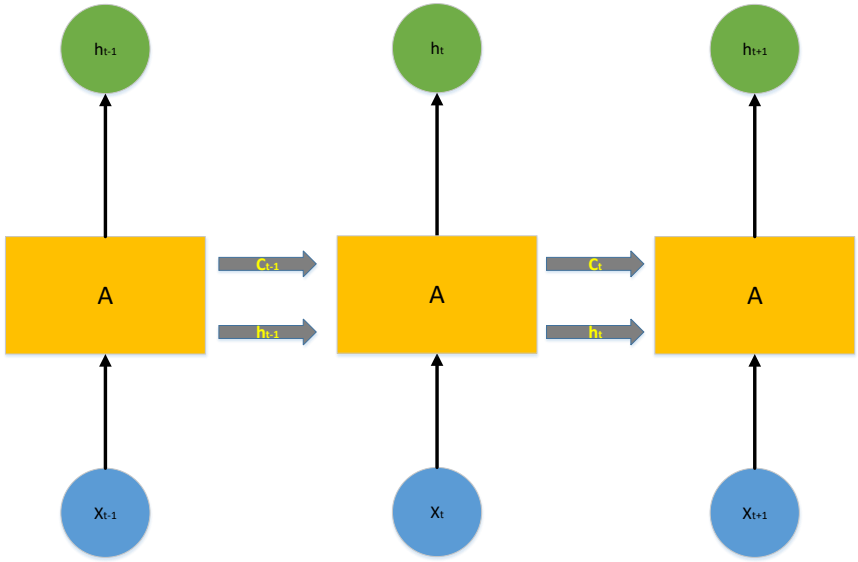


FIGURE 6.5: By introducing a memory cell, LSTMs allow a more complex computation than the standard RNN model.

Some new components are added to the computation unit in LSTMs:

Forget gate: The forget gate determines how much previous information (memory cell) should be kept, and can be calculated using:

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f). \tag{6.2}$$

Input gate: The input gate determines how much current information (input) should be taken into consideration, and can be calculated using:

$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i). \tag{6.3}$$

Output gate: The output gate determines how much information should be exposed (output), and can be calculated using:

$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o). \tag{6.4}$$

Now, we can use the formula which is similar to that in the standard RNNs:

$$\tilde{c_t} = tanh(W^u x_t + U^u h_{t-1} + b^u), \tag{6.5}$$

then combine with the forget gate, previous memory cell and input gate to calculate the new memory cell:

$$c_t = f_t * c_{t-1} + i_t * \tilde{c_t}. \tag{6.6}$$

Finally, based on the internal memory cell and output gate, the gated state of the memory cell will be output:

$$h_t = o_t * tanh(c_t). \tag{6.7}$$

6.3 Fully-Connected Layer and Output Layer

A fully-Connected Layer is widely used in different neural networks, playing a role as extracting meaningful features for the final classifier. The computation in the fully-connected layer can be as simple as a dot product of the input and weight follwed by a pointwise non-linearity:

$$y = f(W^T x), \tag{6.8}$$

where a common example of $f(x)$ is the rectifier function:

$$f(x) = max(0, x). \tag{6.9}$$

In a multiclass classification task, the last fully-connected layer, which can be considered the final classifier, usually uses the softmax function:

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \tag{6.10}$$

where z represents a K-dimensional vector, and K is the total number of target labels. Finally the output layer will be a list of probability values corresponding to each label.

6.4 Learning Algorithm

The idea of training a neural network classifier is similar to many other machine learning methods.

Usually, we randomly initialize the weights of the classifier so that the classifier initially outputs random predictions. Then labelled samples are gradually fed to the classifier. For each sample, the classifier will predict the output label based on the input data and the current weights of the classifier. For example, the classifier may predict that an attack instance is *possessing ball* with 20% probability, *winger-attack* with 75% probability, and *set piece* with 5% probability. This process is formally called forward-propagation (Bishop, 2013).

To improve the performance of the classifier, we need to utilize the "feedback", the error predictions made by the classifier, to adjust the weights of the classifier. This adjustment process first needs an object function which measures the difference between the predicted output and the actual label. A widely used loss function can simply be defined as:

$$E = \frac{1}{2}(t - y)^2, \tag{6.11}$$

where t is the actual label, and y is the predicted output. Then we can calculate the gradient of the loss function using the chain rule:

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w}. \tag{6.12}$$

Finally, we obtain the updated equation:

$$w^{new} = w^{old} - \eta\frac{\partial E}{\partial w}, \tag{6.13}$$

where η is step size (also called the learning rate in the machine learning domain). This process is formally called back-propagation (Hecht-Nielsen et al., 1988).

The aim of the learning algorithm can be summarized as to adjust the weights W so that they can converge to their optimal value, where further learning cannot reduce the loss function, or the error prediction, any more.

6.5 Experiments and Results

For evaluation, we use stratified 10-fold cross-validation, and measure classifcation accuracy, precision, recall, and F1-score by macro-averaging. Stratified cross-validation is used to make label distribution in each training and validation fold as consistent as possible (Kohavi, 1995).

The effect on varying different hyperparameters on the classification accuracy is examined by a grid search method that incrementally changes the values of the hyperparameters. We start from the default setting as shown in Table 6.1, change one parameter at a time, and then analyse the results to find the optimal hyperparameter values. All experiments are run 5 times, and the reported results are collected through averaging results from repeated experiments.

Parameter	**Definition**	**Initial value**
Batch size	Number of samples that will be propagated through the network at each point in time	8
Number of epochs	Epoch is one forward pass and one backward pass of all training data	10
Activation function on the recurrent layer	Non-linearity function applied on the output of recurrent layer neurons	tanh
Activation function on the fully-connected layer	Non-linearity function applied on the output of neurons in the fully-connected layer	softmax
Dropout rate	At each training stage, a node can be dropped out of the network with probability $1 - p$. The reduced network is then trained on the data in that stage	0.5
Number of LSTM units	The number of neurons in the recurrent layer	100

TABLE 6.1: Initial values for neural network hyperparameters.

We report the results of our experiments when varying values of different hyperparameters and measure their impact on the classification accuracy.

6.5.1 Effect of Batch Size and Number of Epochs

Batch size and number of epochs directly impact the total number of iterations needed during the training stage. These relationships can be described using Equation 6.14 and Equation 6.15:

$$Number\,of\,iterations\,per\,epoch = \frac{Size\,of\,training\,data}{Batch\,size}, \tag{6.14}$$

$$Number\,of\,total\,iterations = Number\,of\,iterations\,per\,epoch \times Number\,of\,epochs. \tag{6.15}$$

We investigate the effect of using different batch sizes, from 4 to 64, and different number of epochs, from 1 to 60. Results are shown in Table 6.2 and Table 6.3.

From Table 6.2, we observe that the accuracy decreases if the batch size increases. On the other hand, a large number of epochs, illustrated in Table 6.3, achieves the best performance. We note that when the number of epochs reaches a threshold (around 40), the larger number does not considerably improve the accuracy. We also know that a larger number of iterations will cause longer training time.

The results show that the longer the classifier is trained, the higher accuracy it can achieve. However, we note that a very large number of total iterations may cause overfitting issue, especially since our dataset is relatively small.

	Input data	
Batch Size	**Ball trajectory**	**Players trajectories**
4	**47.1**	**49.2**
8	46.3	48.6
16	44.2	48.3
32	42.6	48.1
64	37.9	44

TABLE 6.2: Impact of batch size on classification accuracy.

Number	**Input data**	
of Epochs	**Ball trajectory**	**Players trajectories**
1	23.5	39.2
3	39	44.1
5	43	47.1
10	46	49.1
20	49.2	51.5
40	49.6	53.9
60	**50.5**	**54**

TABLE 6.3: Impact of number of epochs on classification accuracy.

6.5.2 Effect of Activation Functions

We investigate the effect of changing activation functions in both the recurrent layer and the final fully-connected layer. We consider the following popular functions:

- linear function, which does not change the incoming value;

- rectifier (ReLU), which performs operation: $f(x) = max(x, 0)$;
- tanh function, which element-wisely computes hyperbolic tangent of incoming value;
- sigmoid function, which performs operation: $f(x) = 1/(1 + exp(-x))$;
- softmax function, which is defined in Equation 6.10;
- softplus function, which is a smooth approximation to ReLU: $f(x) = ln(1 + e^x)$;
- softsign function, which performs operation: $f(x) = x/(1 + |x|)$; and
- exponential linear unit (ELU) (Clevert et al., 2015).

In our experiments, we observed dramatic differences in the classification accuracy when different activation functions in the recurrent layer were used. Results are shown in Table 6.4, where the best activation function in the recurrent layer was softsign.

Multinomial logistic regression (softmax regression) is a common classifier to perform multiclass classification. It calculates the cross-entropy between the normalized predictions and a 1-hot encoding of the label. Table 6.5 shows that it outperforms other options when used in the fully-connected layer.

Function	**Input data**	
	Ball trajectory	**Players trajectories**
ELU	13.8	17.8
softmax	23.8	36.5
linear	13.8	13.8
ReLU	18.7	17.5
sigmoid	36.2	42.6
softplus	14.2	13.9
softsign	**47.4**	**52.4**
tanh	46.1	49.5

TABLE 6.4: Impact of different activation functions in the recurrent layer on classification accuracy.

6.5.3 Effect of Number of LSTM units

Results from our investigation of the impact of the number of LSTM units is shown in Table 6.6. From this table, we observe that the best number of LSTM units when different trajectory data is used varies. When the ball trajectory is used as the input data, more neurons may achieve higher accuracy. When the players trajectories were used, a higher accuracy was achieved with a smaller number of LSTM units.

Function	Input data	
	Ball trajectory	**Players trajectories**
ELU	31.9	39.5
softmax	**48.3**	**53**
linear	32.6	40
ReLU	28.5	40.5
sigmoid	45.3	51.5
softplus	36.3	46.7
softsign	36.8	39.3
tanh	36.9	40.6

TABLE 6.5: Impact of different activation functions in the fully-connected layer on classification accuracy.

Number	Input data	
of LSTM units	**Ball trajectory**	**Players trajectories**
50	42.7	48.8
100	47.6	**51.6**
200	50.3	51.2
400	53.3	49.1
800	**54.3**	43.2

TABLE 6.6: Impact of number of LSTM units on classification accuracy.

The reason for this could be that when the ball trajectory is used, the input data is continuous values representing the x and y coordinates of the ball. In this case, the classifier needs more neurons to extract useful representation from this wide range of data. On the other hand, when players trajectories are used, the input data of the neural network is the distribution of players on the field, concretely, the number of players occupying each pre-defined regions. The discrete information could be effectively learnt by a small number of neurons.

6.5.4 Effect of Dropout Rate

Dropout is a technique for addressing the overfitting issue. Through randomly dropping units from the neural network during training, it can reduce co-adaptations among units (Srivastava et al., 2014). The choice of dropout rate depends on both the complexity of the model and the size of training data. From table 6.7, we can see, that our model is not complex enough, so keeping more neurons (lower dropout rate) will improve the performance.

Droput Rate	Input data	
	Ball trajectory	**Players trajectories**
0	**49.4**	**53.7**
0.3	49.1	52.6
0.5	48.3	51.2
0.7	44.1	50.7
0.9	36.8	45.8

TABLE 6.7: Impact of dropout rate on classification accuracy.

6.5.5 Summary

Through the grid search of varying values of different hyperparameters, we report our best hyperparameter values in Table 6.8, and the best performance of the neural network in Table 6.9.

Parameter	Input data	
	Ball trajectory	**Players trajectories**
Batch size	4	4
Number of epochs	60	60
Activation function on the recurrent layer	softsign	softsign
Activation function on the fully-connected layer	softmax	softmax
Dropout rate	0	0
Number of LSTM units	800	100

TABLE 6.8: Best hyperparameter values of the neural network we built.

Input data	Accuracy	Precision	Recall	F1-score
Ball trajectory	54.3	51.1	54.3	52.9
Players trajectories	54	52.6	54	54.7

TABLE 6.9: The performance of the neural network approach.

Since the results shown in Table 6.9 are not satisfactory, we then resorted to exploit the conventional classification methods.

CHAPTER 7

Rule-Based Features Approach

In contrast to the neural network approach, which directly works on raw data, the rule-based feature approach is to transform the raw spatiotemporal data to appropriate features which are built using domain-specific knowledge. These features will then be fed to machine learning classifiers to train the model. Using the trained model, predictions can be performed on unseen data.

7.1 Feature Engineering

The aim of feature engineering is to design appropriate feature functions so that sufficient predictor variables can be extracted and then fed to the classifiers. The design of predictor variables is inspired by how a human observer of a football match would perform a similar classification task.

The match observer would consider the fundamentals of the attack instance, such as the time duration of the attack, the number of passes during the attack, and the number of attacking players who are involved. Intiuitively, the longer the possession, the larger number of passes, the larger number of attacking players involved, the more likely the instance is of *possessing ball*. On the other hand, type *counter-attack* is usually of shorter time duration, less number of passes and less number of attacking players involved.

In addition, some basic geometric predictor variables are also helpful for the observer to identify some types of attacks. These variables include the start point of the attack instance, and whether the ball has once entered attacking third area that implies a threat to the opponent's goal. For example, if an attack starts from the corner area, the instance is very likely to be *set piece*. If the ball has never entered attacking third area, the more likely the instance is *not-an-attack*.

Finally, the last pass within the attack may have greater impact on the decision of the observer. For example, if the last pass of the attack is directly from the side of the pitch to the penalty box, the

observer may therefore classify the attack as *winger-attack*, even though the attack may have lasted a long time indicating it could be a *possessing ball*.

Utilizing this domain-specific knowledge, we implemented several feature functions, which are listed in Table 7.1.

7.2 Classifiers

Several common supervised machine learning classifiers are evaluated for our classification task, including logistic regression, support vector machine, and random forest.

Informally, a classifier can be considered a hypothesis function $h_w(x)$, which takes as input vector x and outputs variable y. Therefore, the training of the classifier is to find an optimal parameter set w so that the prediction error on a given set of labelled training data is minimised.

7.2.1 Logistic Regression

Logistic regression is a regression model used to estimate the probability of a output label y, based on input features x (Hilbe, 2009). The estimation can be calculated using:

$$p(y = j|x, w) = \sigma(w_j^T x), \tag{7.1}$$

where w_j is the weigth vector of the classifier in terms of class j, and the logistic function σ is defined as follows:

$$\sigma(t) = \frac{1}{1 + e^{-t}}. \tag{7.2}$$

The output of the logistic function always takes the value between zero and one. Therefore, we can determine a data point is of class j if

$$p(y = j|x, w) \geq \nu_j, \tag{7.3}$$

where ν_j is a threshold we defined for class j.

In the term of multiclass classification, we can determine the output label using:

$$y = \arg\max_j p(y = j|x, w). \tag{7.4}$$

ID	Feature Name	Description	Variable Type	Domain
1	Attacking time duration	The time duration of the attack instance	Discrete	N
2	Number of passes	The number of passes during the attack	Discrete	N
3	Number of players	The number of players who are involved in the attack	Discrete	N
4	Starting point x-coordinate	The x-coordinate of the ball when the attack starts	Continuous	Z
5	Starting point y-coordinate	The y-coordinate of the ball when the attack starts	Continuous	Z
6	Number of attackers when attack starts	The number of players who locate between the ball and the opponent's goal when the attack starts	Discrete	N
7	Nearest defender when attack starts	The distance between the ball and the nearest opponent player when the attack starts	Continuous	N
8	End point x-coordinate	The x-coordinate of the ball when the attack ends	Continuous	Z
9	End point y-coordinate	The y-coordinate of the ball when the attack ends	Continuous	Z
10	Starting point of the last pass x-coordinate	The x-coordinate of the ball when the last pass within the attack starts	Continuous	Z
11	Starting point of the last pass y-coordinate	The y-coordinate of the ball when the last pass within the attack starts	Continuous	Z
12	End point of the last pass x-coordinate	The x-coordinate of the ball when the last pass within the attack ends	Continuous	Z
13	End point of the last pass y-coordinate	The y-coordinate of the ball when the last pass within the attack ends	Continuous	Z
14	Last pass distance	The distance of the last pass within the attack	Continuous	$\mathbb{R}$
15	Last pass speed	The speed of the last pass within the attack	Continuous	$\mathbb{R}$
16	Threat area	Indicator to determine whether the ball has once entered the attacking third area	Binary	$\{0, 1\}$
17	Total ball trajectory length	The total distance the ball runs during the attack instance	Continuous	Z

TABLE 7.1: The feature functions implmented in our classification model.

Given a set of training data points, the aim of training the logistic regression classifier can be defined as the one to find an optimal weight w so that the likelihood for all labelled data points can be maximized:

$$w^* = \arg\max_w \prod_{i=1}^{n} P(y_i|x_i, w). \tag{7.5}$$

The cost function which measures how well a given weight w fits the training data can be given using the logarithm of likelihood in Equation 7.5:

$$\begin{aligned} L(w) &= \sum_{i=1}^{n} \log P(y_i|x_i, w) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{k} [1_{\{y^i=j\}} \log P(y_i = j|x_i, w)]. \end{aligned} \tag{7.6}$$

Then the parameterisation of w is learned by minimising the cost function $L(w)$.

7.2.2 Support Vector Machine

A support vector machine (SVM) is a robust supervised learning method that is widely used for various tasks, including classification, regression and outlier detection (Bishop, 2013).

The intuition of SVM is to define a hyperplane that can separate different classes. Therefore the training of SVM is to find the optimal hyperplane that represents the largest margin between different classes. In other words, this optimal hyperplane is the one which has the largest distance to its nearest data points.

Figure 7.1 is a simple example of a hyperplane and margins for an SVM. Let a candidate hyperplane be denoted using: $f(x) = w^T x + b$, where w is weight vector and b is the bias. Then the distance between a data point x_i and this hyperplane can be calculated using:

$$distance = \frac{w^T x_i + b}{\parallel w \parallel}. \tag{7.7}$$

For a candidate hyperplane, we call its nearest data points support vectors. Here, we use x_s to denote a support vector.

If we define $w^T x_s + b = 1$, then the distance between the hyperplane to its support vectors will be $\frac{1}{\|w\|}$. Meanwhile since support vectors are data points which are nearest to the hyperplane, The distance between the hyperplane to all data points should be no smaller than $\frac{1}{\|w\|}$.

Now, the training of an SVM problem can be re-defined as to maximize the distance between the hyperplane to its support vectors:

$$\arg\max_{w,b} \frac{1}{\parallel w \parallel}. \tag{7.8}$$

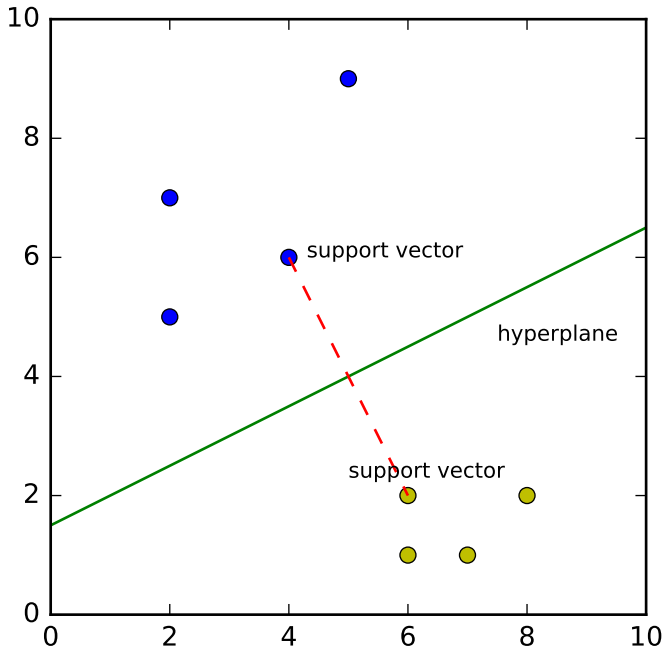


FIGURE 7.1: A simple example of hyperplane and margins for a binary SVM.

This problem is equivalent to a problem of Lagrangian optimization that minimizes a function:

$$\arg\min_{w,b} \frac{1}{2} \| w \|^2, \tag{7.9}$$

with a constraint that for all data points x_i:

$$y_i(w^T x_i + b) \geq 1, \tag{7.10}$$

where y_i is the label of data point x_i.

7.2.3 Random Forest

Random forest is an ensemble learning method that can be traced back to the decision tree method. A number of decision trees are constructed during the training stage, and then decisions of individual trees will be combined during the prediction stage. In addition, random forest utilize sampling and randomness to solve the issue of overfitting which often occurs in the decision tree method (Ho, 1995).

In terms of training, a decision tree is built through recursively selecting features to split the training data so that samples of the same labels are grouped together. After the tree is constructed, the prediction stage will be a straightforward flowchart-like path from root to leaf. In this path, each internal node performs a test on a feature, and the leaf node represents a class label. Figure 7.2 shows a simple prediction using a decision tree.

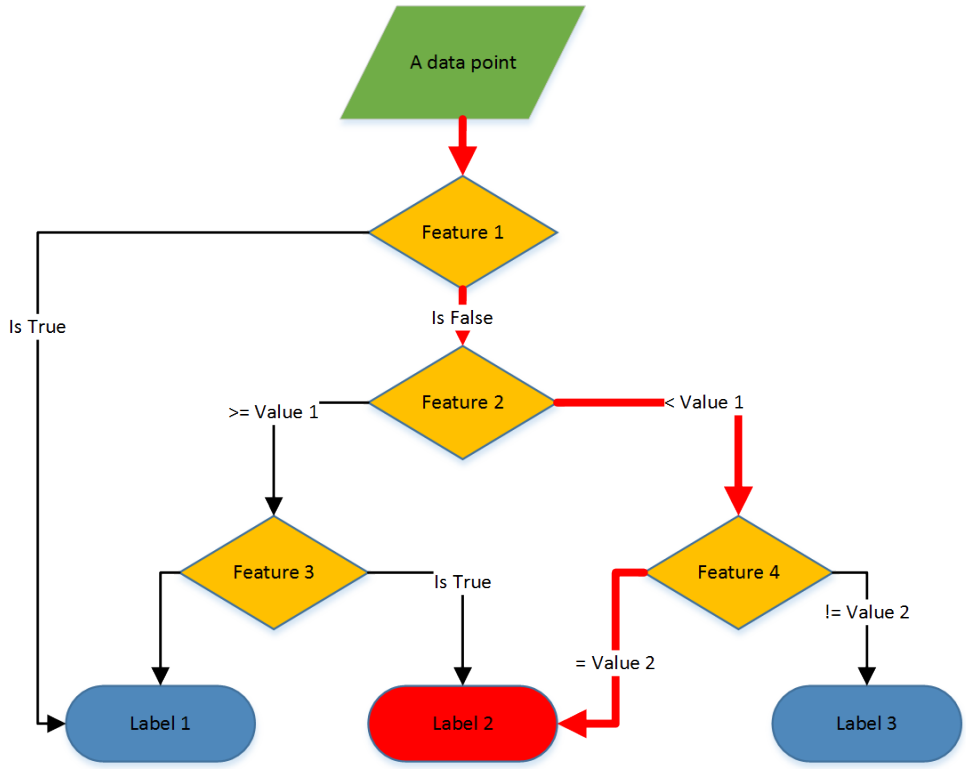


FIGURE 7.2: A simple prediction using a decision tree. In this example, Feature 1 is a binary variable, Feature 2 is continuous, etc. Finally, the input data point is predicted as *Label 2*.

Random forests classifier is a variant of the decision tree classifier in terms of two improvements.

Bootstrap aggregating: Given a training dataset, several datasets with different size will be generated via sampling from the original training dataset. The advantage of training on these different datasets is to make built trees not correlated and therefore robust to noise in the original training dataset.

Feature bagging: Selecting not only a subset of samples, but also a subset of features is another characteristic of the random forest classifier. The intuition is that one or a few features which are strong predicators may become correlated in many of the trained trees. Through randomly

selecting a subset of features, the correlations could be broken, and the model generalises better.

7.3 Experiments and Results

The experimental setup used in this section is the same as what we described in Section 6.5. Stratified 10-fold cross-validatoin is used, and metrics including accuracy, precision, recall and F1-score are collected through macro-averaging.

7.3.1 Classifier Performance

Table 7.2 lists the performance of different classifiers built on top of rule-based features. The random forest classifier achieves the highest accuracy among all classifiers with an accuracy of up to to 76.6 %.

Classifier	Accuracy	Precision	Recall	F1-score
SVM	75.2	70.9	68.2	67.7
Random Forests	**76.6**	**73.9**	**70.3**	**70.4**
Logistic Regression	75.8	69.3	69	67.7

TABLE 7.2: Comparison of the performance of different classifiers built on rule-based features.

In addition, the rule-based feature approach greatly outperforms the previous neural network approach.

7.3.2 Feature Importance

Features used in this approach are designed based on our understaning of football attacking tactics. However, different features are computed of varying complexity. For example, the time duration of the attack instance is easily computed via the subtraction of the start time step from the end time step, while some other features are calculated based on methods from computational geometry whose computatoin are very expensive, such as ball trajectory length.

This leads to the question whether all features used result in an improved performance, especially these features whose computation are expensive.

Several methods can be used to determine the importance of a particular feature in terms of different classifiers. We investigate two methods in this research.

ℓ_1 regularization of logistic regression classifier. Regularizing by the 1-norm is to induce sparsity (Jenatton et al., 2011). It means that the weights of features which are of low importance will be exactly equal to zero, depending on the strength of the regularization. Therefore, we examine the feature importance via the weigths vector for the logsitic regression using ℓ_1 regularization.

Table 7.3 lists the most and least important features for each class.

Class	**Important features**	**Not important features**
Counter attack	Threat area Number of attackers when attack starts	Number of passes End point y-coordinate Starting point y-coordinate
Long through ball	Threat area Number of attackers when attack starts Ball trajectory length	End point y-coordinate Starting point y-coordinate Attacking time duration
Not an attack	Threat area Last pass speed Number of players	End point x-coordinate Starting point x-coordinate Last pass distance
Not running	Ball trajectory length Number of passes Starting point x-coordinate	Number of players Number of attackers when attack starts Last pass speed Threat are
Possessing ball	Threat area Number of passes Number of players Ball trajectory length	Starting point y-coordinate End point y-coordinate
Set pieces	Number of players Number of attackers when attack starts Ball trajectory length Last pass speed	Threat area End point y-coordinate
Winger attack	Number of players threat area Ball trajectory length Last pass speed	End point of the last pass y-coordinate End point of the last pass x-coordinate

TABLE 7.3: The importance of features vary in the term of predicting different classes.

Mean decrease impurity of random forest classifier. Random forests consists of a host of decision trees, every node of which represents a condition checking on a particular feature. The training of the random forest is therefore to choose optimal condition for each node. based on these conditions, the dataset can be splitted so that consistent predictions can be made on the similar predictor variables.

Node impurity measures how well the node splits the dataset, in other words, how useful a particular feature is to improve the performance of the classifier. Using the $RandomForestRegressor$ from $sklearn.ensemble$, we can calculate how much each feature decrease the weighted impurity.

From Figure 7.3, we can see the most important feature is Feature 17 (Ball trajectory length). In addition, some basic features, such as Feature 1 (attacking time duration), Feature 2 (number of passes), and Feature 3 (number of players), are also very imformative features.

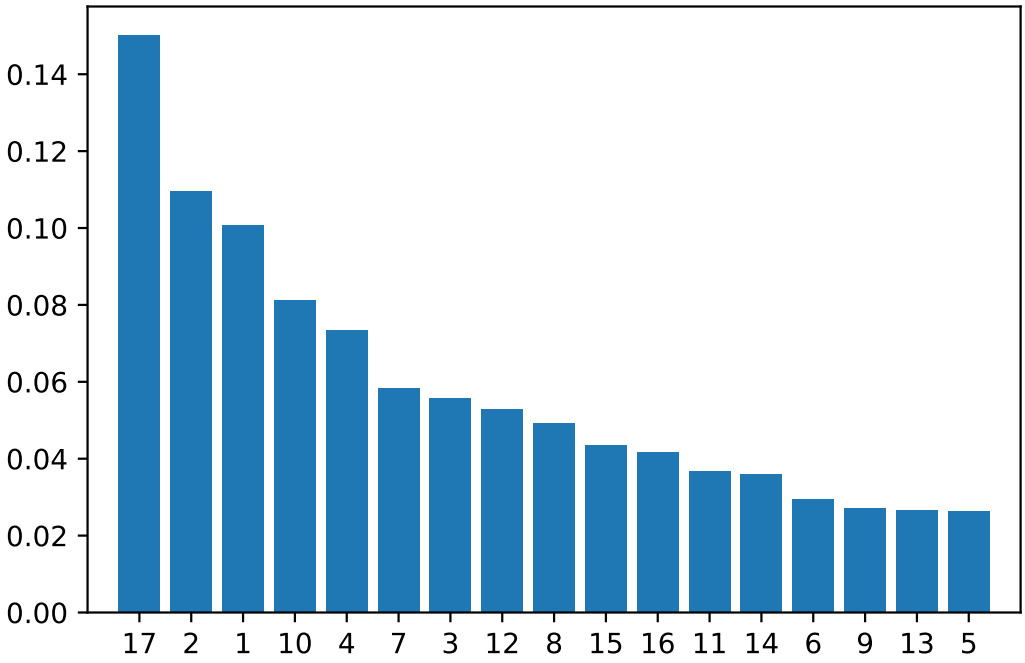


FIGURE 7.3: Feature importance computed using mean decrease impurity of random forest classifier. The corresponding meaning of feature ID can be found in Table 7.1.

Combining the results from these two methods, we find that some basic features can be very useful to our classification task, such as attacking time duration, number of passes, and number of players. The other finding is that y-coordinate of the location is less important than the corresponding x-coordinate. The reason of this could be that the x-coordinate is a more direct indicator on the distance, therefore the threat, to the opponent's goal.

CHAPTER 8

Further Exploitation of Deep Learning

The results show that rule-based feature approach outperforms the neural network approach, and the gap between them is large. We are interested to see if the deep learning method can be improved, with the help of conventional classification methods.

8.1 Utilizing Larger Data Set

In our dataset, there are only 434 labelled samples out of a total of 18,184 samples. On the other hand, deep learning is considered to benefit from a large volume of data (Bengio, 2009; LeCun et al., 2015; Schmidhuber, 2015). However, watching animations and manually annotating attacks is very labor-intensive so it is difficult to create a large enough training dataset for a neural network classifier.

Inspired by the progressive training strategy described in (You et al., 2015), we utilize the trained conventional classifier to generate a larger training dataset.

Figure 8.1 is an illustration of how we generate the training dataset and how to perform the evaluation. Concretely, we select the random forest classifier which achieves the best performance on previous experiments as our generator classifier. The manually labelled samples are used to train the generator classifier, and then the trained classifier will perform predictions on a large volume of unlabelled samples. With the predicted labels, these samples will then be used to train the neural network classifer, and finally the neural network classifier will be evaluated on the original manually labelled samples.

8.1.1 Experiment Result

With a large volume of training data, we investigate the effect of the size of training dataset on the performance of both the neural network classifier and the conventional classifiers.

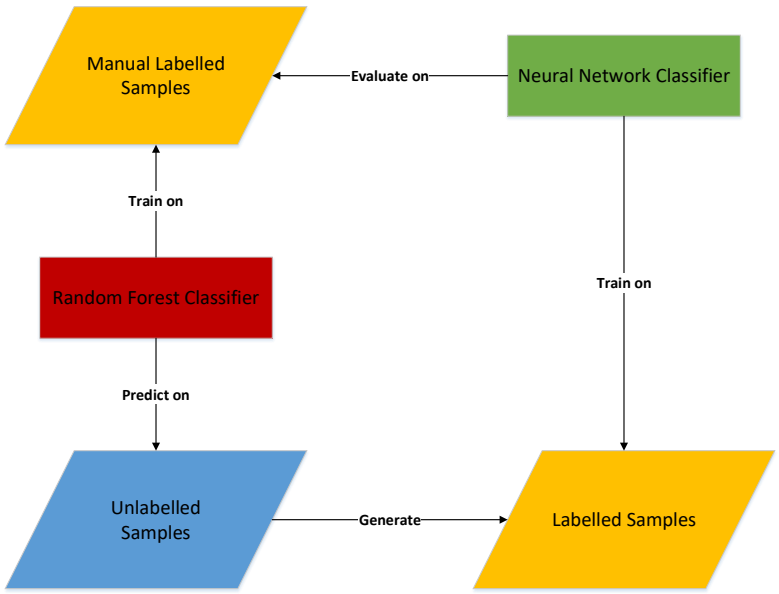


FIGURE 8.1: An illustration of how to generate larger training data and how to perform the evaluation.

Table 8.1 lists the performance of different classifiers when they are trained on datasets of different sizes. We note that all classifiers can benefit from a larger dataset, even if this dataset is generated by a trained classifier, and therefore, of lower quality than the original manaully labelled dataset.

From Table 8.1, we observe that SVM and logistic regression do not improve further when the size of the training dataset reaches a threshold 5,000. On the other hand, the neural network classifier continues to improve while increasing the size of the training dataset.

Classifier	Size of Training Data			
	500	2000	5000	10000
SVM	75.6	77.9	78.3	78.3
Logistic Regression	74.4	78.1	78.8	78.8
Random forests	75.3	80	82.9	83.6
LSTM using ball trajectory	31.8	44.4	47.6	53.2
LSTM using players trajectories	36.2	51.5	60.9	65

TABLE 8.1: Comparison of the accuracy of different classifiers when they are trained on datasets of different sizes.

8.2 Combine Input Data

Another direction we investigated is to combine all input data together: ball trajectory, players trajectories, and rule-based features, and to observe whether the neural network can be improved through multiple data sources. Figure 8.2 is a summary illustration of the combined neural network architecture.

The basic idea of the combination is to let a recurrent layer learn the representations of the ball trajectory and the players trajectories respectively, and then combine these representations and rule-based features together as the input of the last fully-connected layer.

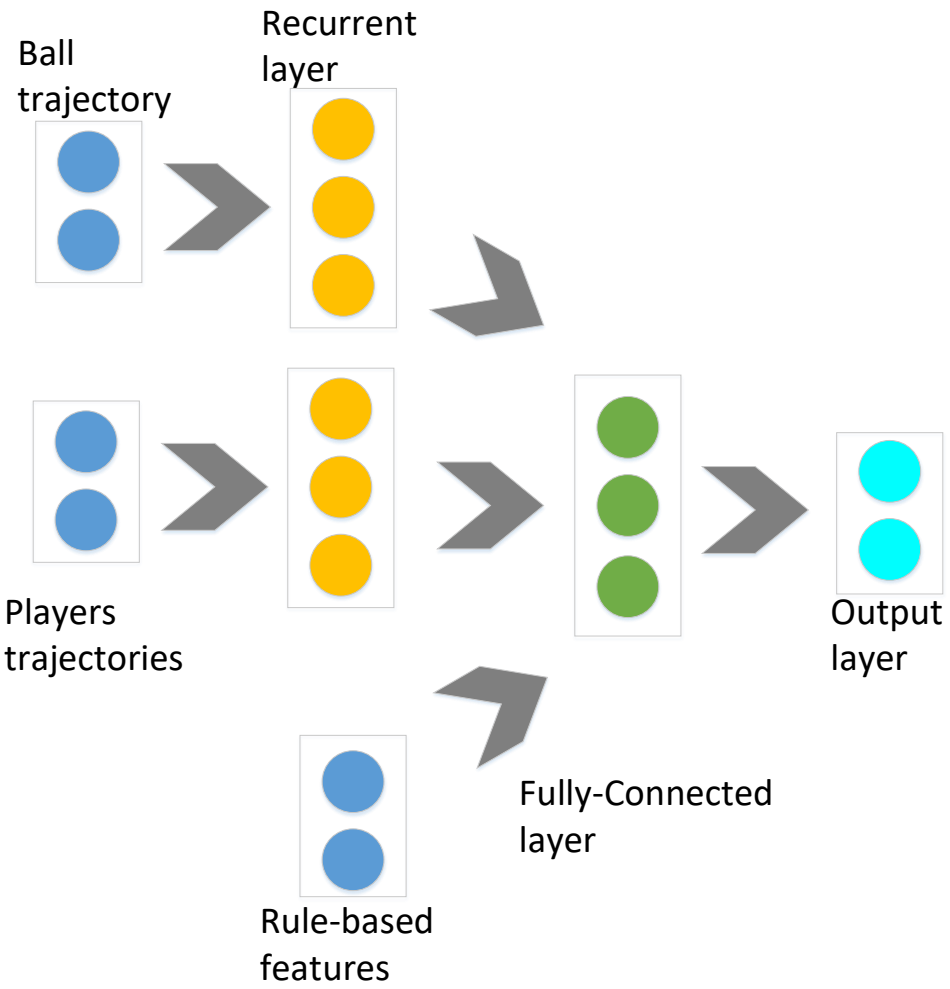


FIGURE 8.2: An illustration of how the neural network classifier can use multiple data sources.

8.2.1 Experiment Result

We investigate the performance when considering all three types of input data: ball trajectory, players trajectories, and rule-based features. We also test the performance when only the ball trajectory and the players trajectories are used. The results are listed in Table 8.2, and show that when the ball trajectory and the players trajectories are used, the performance of the neural network classifier achieves significant improvement. However, the addition of the rule-based features does not improve the performance of the classifier.

Note that these experiments are conducted using training data described in Section 8.1, and hyperparameter values listed in Table 6.1.

Input data	Accuracy	Precision	Recall	F1-score
Ball trajectory only	53.2	48.4	53.2	48.3
Players trajectories only	65	62.3	65	62.6
Ball trajectory and players trajectories	**75.6**	**73.2**	**75.6**	**72.6**
Two trajectories and rule-based features	66.4	60.8	66.4	59.4

TABLE 8.2: Comparison of the performance of the neural network when different input data are used.

8.3 Summary

In this chapter, we investigated two methods to improve the neural network with the help of conventional classification methods. A large dataset generated by a trained conventional classifier can improve the performance of neural network which usually benefits from a large volume of data. On the other hand, feeding a neural network with manually designed features and combining them with raw data does not improve the performance. The reason could be that manaully designed features have no inherent correlation with raw trajectory data so that the neural network cannot find an effective way to learn them together. In other words, multiple data sources, which have inherent correlation, such as the ball trajectory and the players trajectories, can achieve better performance compared to using only a single data source.

CHAPTER 9

Discussion

In this chapter, we will discuss what kind of errors our system makes, and why it makes these errors.

A confusion matrix is a table that is used to describe the performance of a classifier on test data. Table 9.1 is a confusion matrix that reports the performance of the random forest classifier which achieves the best performance in our classifaction task. The classifier is trained on the large dataset described in Section 8.1, and is evaluated on the manually annotated dataset.

	Counter attack	Long through ball	Not an attack	Not running	Possessing ball	Set pieces	Winger attack
Counter attack	52	0	3	0	1	1	3
Long through ball	6	7	0	0	5	0	2
Not an attack	7	1	51	0	5	1	0
Not running	0	0	1	88	0	0	0
Possessing ball	1	0	1	0	74	0	1
Set pieces	3	0	0	0	2	63	0
Winger attack	8	0	1	0	15	4	27

TABLE 9.1: Confusion matrix of random forest classifier. The entries $C_{i,j}$ in this matrix tablulate the number of attacks whose true label is class i while classified to be class j by the random forest classifier.

From Table 9.1, we note that lots of *winger-attacks* are misclassified as *possessing ball*. We realize that these attack instances usually consist of different stages where different tactics are performed. Commonly, the attack may start as *possessing ball*, and end with *winger-attack*. These error predicions show that it is difficult for the classifier to understand the relationship between different tactics performed at different stages. For example, it is difficult to know whether the final crossing from side of the pitch can be considered a reasonable outcome of the *possessing ball*, or if it is a total independent *winger-attack* since the team cannot find open space, and therefore change to a new tactic.

Among all labels, *not running* has the clearest decision boundary with other categories. From Table 7.3, we can see that the feature *ball trajectory length* is a very strong predictor for type *not running*.

However, we note that the location of the ball is not automatically traced, so the performance on label *not running* may overestimate the effectiveness of our model to reduce human efforts.

On the other hand, the *long through ball* is the category with worst performance. A major reason is from the imbalance distribution between different categories that the number of labelled samples of label *long through ball* is the smallest. Our dataset is from matches played by Bayern Munich, which is the best football team in German. Most of the time, they intend to control the ball, and seldom perform *long through ball* tactic. The bias of training data makes the classifier misclassify many attacks of *long through ball* to other categories.

Finally, the trajecotry data we use is calculated in the 2D coordinate system. Sometimes, it is very difficult to decide whether the ball is running through the air or the ground from this data. The deficiency of information may also compound the difficulty of the classification task.

CHAPTER 10

Conclusion

In this thesis, we investigated the effectiveness of a recurrent neural network on learning attacking tactics from raw trajectory data. Besides, we analysed the impact of the hyperparameters on the classification accuracy.

We also presented a rule-based feature approach to solve the classification problem, and achieved the best performance with accuracy of 83.6%.

The results show that conventional classification methods still outperform a deep learning method in our classification of attacks. However, we note that the neural network classifier can achieve significantly better performance using larger datasets. On the other hand, the conventional classifiers, such as SVM and logistic regression will not continue to be improved after the size of training data reaches a threshold.

We also investigated the effectiveness of combing multiple data sources into a neural network. The result shows that when the ball trajectory and the players trajectories are used together to train the neural network, it can achieve a significant improvement with accuracy of 75.6% than when only ball trajectory or players trajectories is used.

In terms of future research, there could be two directions. One is to further exploit other deep learning methods, such as the intention. The intention mechanism is trying to find the most important information in a sequence, and then make predictions based on that most important information. Other directions for conventional classifcation methods is to design new predictor variables so that the errors of the classifiers can be further reduced.

Bibliography

Michael Beetz, Nicolai von Hoyningen-Huene, Bernhard Kirchlechner, Suat Gedikli, Francisco Siles, Murat Durus, and Martin Lames. 2009. Aspogamo: Automated sports game analysis models. *International Journal of Computer Science in Sport*, 8(1):1–21.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Alina Bialkowski, Patrick Lucey, Peter Carr, Yisong Yue, Sridha Sridharan, and Iain Matthews. 2014. Large-scale analysis of soccer matches using spatiotemporal tracking data. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 725–730. IEEE.

Christopher M. Bishop. 2013. *Pattern Recognition and Machine Learning*. Information science and statistics. Springer.

Anna Bosch, Xavier Muñoz, and Robert Martí. 2007. Which is the best way to organize/classify images by content? *Image and vision computing*, 25(6):778–791.

Yu-Han Chang, Rajiv Maheswaran, Jeff Su, Sheldon Kwok, Tal Levy, Adam Wexler, and Kevin Squire. 2014. Quantifying shot quality in the nba. In *Proc. 8th Annual MIT Sloan Sports Analytics Conference*, pages 1–8.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42.

Jordi Duch, Joshua S Waitzman, and Luís A Nunes Amaral. 2010. Quantifying the performance of individual players in a team activity. *PloS one*, 5(6):e10937.

Ian M Franks and Gary Miller. 1986. Eyewitness testimony in sport. *Journal of sport behavior*, 9(1):38.

Joachim Gudmundsson and Michael Horton. 2017. Spatio-temporal analysis of team sports. *ACM Computing Surveys (CSUR)*, 50(2):22.

Joachim Gudmundsson and Thomas Wolle. 2014. Football analysis using spatio-temporal tools. *Computers, Environment and Urban Systems*, 47:16–27.

Robert Hecht-Nielsen et al. 1988. Theory of the backpropagation neural network. *Neural Networks*, 1(Supplement-1):445–448.

Joseph M Hilbe. 2009. *Logistic regression models*. CRC press.

Tin Kam Ho. 1995. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Michael Horton, Joachim Gudmundsson, Sanjay Chawla, and Joël Estephan. 2015. Automated classification of passing in football. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 319–330. Springer.

Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. 2011. Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research*, 12(Oct):2777–2824.

Ron Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Hoang M Le, Peter Carr, Yisong Yue, and Patrick Lucey. 2017. Data-driven ghosting using deep imitation learning. In *MIT Sloan Sports Analytics Conference (SSAC)*.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

Patrick Lucey, Dean Oliver, Peter Carr, Joe Roth, and Iain Matthews. 2013. Assessing team strategy using spatiotemporal data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1366–1374. ACM.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Andrew Miller, Luke Bornn, Ryan Adams, and Kirk Goldsberry. 2014. Factorized point process intensities: A spatial analysis of professional basketball. In *International Conference on Machine Learning*, pages 235–243.

Charles Perin, Romain Vuillemot, and Jean-Daniel Fekete. 2013. Soccerstories: A kick-off for visual soccer analysis. *IEEE transactions on visualization and computer graphics*, 19(12):2506–2515.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566. Association for Computational Linguistics, Beijing, China.

Tsuyoshi Taki and Jun-ichi Hasegawa. 2000. Visualization of dominant region in team games and its application to teamwork analysis. In *Computer Graphics International, 2000. Proceedings*, pages 227–235. IEEE.

Chih-Fong Tsai. 2012. Bag-of-words representation in image annotation: A review. *ISRN Artificial Intelligence*, 2012.

Quanzeng You, Jiebo Luo, Hailin Jin, and Jianchao Yang. 2015. Robust image sentiment analysis using progressively trained and domain transferred deep networks. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 381–388. AAAI Press, Austin, Texas.

Yisong Yue, Patrick Lucey, Peter Carr, Alina Bialkowski, and Iain Matthews. 2014. Learning fine-grained spatial models for dynamic sports play prediction. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 670–679. IEEE.