Universiteit Leiden

Master Computer Science

Automatic Pass Detection in Soccer Tracking Data

Name:	M.F.A. Aarnoutse
Student ID:	2682796
Date:	August 27, 2023
Specialisation:	Artificial Intelligence
1st supervisor:	Dr. A.J. Knobbe
2nd supervisor:	Dr. M. Baratchi

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

In this thesis, we developed a method for automatic pass detection in soccer matches using tracking data as input, and compare it to the event data set of Sportradar. In order to evaluate the performance of our methods, we synchronised the event data set to the tracking data, in which we update the time of the event with the frame number it occurred in. This acts as a ground truth for our methods. In our first approach, we start by defining rules for what could encompass a pass. For each frame we find the nearest player to the ball, if the player is within a certain distance to the ball, we check if the rules apply and predict a pass. In the second approach we apply deep learning techniques to the tracking data to detect passes, specifically a convolutional neural network (CNN) and a recurrent neural network (RNN). We compare the performance of these two approaches and find that a slightly better performance is achieved using the recurrent neural network with bidirectional LSTM nodes than the rule-based system. We also find that the rule-based system performs better than the CNN and that the representation we use for the CNN is not suitable for this task. The CNN is difficult to work with, due to the large size of the data set. The rule-based system does not have this disadvantage and might be easier to understand and improve for experts. The recurrent neural network achieves an F_1-score of 0.78 on the test set, while the rule-based system achieves an F_1-score of 0.75.

Contents

Chapter 1

Introduction

FIFA created the FIFA Football Language in 2021, which will be their blueprint for how they analyse football in the future (Wenger, 2021). In their vision they describe live metrics and new insightful stats that would provide fans with context what is happening on the pitch. This is in line with the current trend in the soccer business where more and more data is collected and used to provide insights into the game. Companies such as Sportradar and Opta provide data to clubs, media and betting companies. However, in the past, customers were only interested in the top leagues. Nowadays, clubs also want the data of lower leagues and youth matches. The volume of matches that need to be analysed during a week therefore increases rapidly. Gathering the data from these matches has been done manually and the search for more efficient alternatives is starting. In this thesis we will explore the possibility of using tracking data to automatically generate the pass events that occurred in a match in order to augment and potentially even replace the manual data collection process.

In soccer there are two types of data, namely event data and tracking data. Event data is the collection of all actions made by all the players and/or referee; identifying the type of action, the location and additional properties of the action. Examples of events are passes, dribbles, shots, fouls and substitutions. In each soccer match roughly $1{,}600$ events are tagged, of those events more than 50% are passes. Shots account for 1.5% of the total events, while goals are even rarer with an occurence of less than 1% (Pappalardo et al., 2019). As these passes are the most common event in a soccer match, automating the detection could potentially relieve the taggers from the most time-consuming task. The other type of data used in this thesis is tracking data, which is the collection of the location of all players and the ball gathered at a certain frequency. It is collected by either wearables on the players and in the ball, or by computer vision where either the cameras are set up live in the stadium or broadcast footage is used to generate tracking data from. Common sample rates are 10 or 25 Hz, which results in over $140{,}000$ data points per match

at the latter sample rate. In this thesis, we use tracking data provided by Sportlogiq, using a sample rate of 25 Hz and collected from broadcast footage.

The collection of tracking data can be automated. This is currently not the case for the events. In order to gather the events in a soccer match, at least four taggers are required to watch the match and keep track of all actions themselves. Besides the fact that this is a time-consuming process and demands a lot of human resources, it is also prone to human error. As tracking data becomes more readily available, this can be used to validate and improve the event collection and potentially replace it with an automated process. In this thesis, we will explore the possibility of using tracking data to automatically detect pass events.

Not all events are suitable for automatic detection, a foul would be difficult to detect automatically as it is not always clear when a foul is committed and depends on the referee's interpretation. However, events like passes, dribbles and shots are all on the ball, which makes the interaction visible in the tracking data. Off-the-ball events, such as defensive actions or substitutions, are either more difficult or impossible to detect automatically without additional information. In this thesis, we will focus on the automatic detection of passes as these are by far the most common events in a soccer match (Pappalardo et al., 2019). This results in a bigger data set to work with, and could relieve the taggers from the most time-consuming task.

1.1 Problem Statement

In this thesis we explore the possibility of using tracking data to automatically detect passes. This leads to the following research question:

> *How can we use tracking data to automatically detect passes in soccer matches?*

In order to answer this question, we define the following sub-questions:

> *What rules can we define to detect passes in the tracking data?*
>
> *How can we evaluate the performance of the pass detection?*
>
> *How can we improve the performance upon the rule-based approach by using machine learning?*

1.2 Outline

In Section 2 we discuss the existing work in the field of event detection in soccer, where we can build upon. Next, in Section 3 we compare the tracking data and the event data that is used in this thesis, give an overview of the distribution of the event data set and explain the coordinate system used in the

tracking data. The methods used to detect the passes automatically are then described in Section 4. The results of the experiments are presented in Section 5 and discussed in Section 6. Finally, in Section 7 we draw the conclusions of this thesis.

Chapter 2

Related Work

This is a fairly new field of research, so there is not a lot of literature available regarding automatic event detection on tracking data in soccer. However, there has been some research on using video data to detect events. These events are limited to goals, shots, corners, free kicks, yellow cards, fouls and offside as they rely on object detection on video data before the classification task (Fakhar, Rashidy Kanan, & Behrad, 2019). In this chapter we will discuss the research that has been done in similar tasks and how it relates to our research.

Most of the research available deals with event recognition based on video, whether this is broadcast footage or from dedicated cameras in stadia. Yu, Lei, and Hu (2019) developed a Convolutional Neural Network (CNN) model for event and replay detection. The authors annotated the broadcast video when a goal, shot, corner, free kick, yellow card, foul or offside event occurred. They achieved a precision of 0.340 and a recall of 0.409 on detecting goal events using their CNN with replay detection. Fakhar et al. (2019) also employs highlight detection, but opted for a Long Short-Term Memory (LSTM) network to classify events. Both these papers did not include passes in their events.

Automatic event detection. One paper in particular that made advancements in automatic event detection is from Vidal-Codina, Evans, El Fakir, and Billingham (2022). FIFA started researching this topic and called it the first attempt that has been made at automatically detecting events from tracking data. They used a deterministic decision tree-based algorithm where no learning is involved. They begun with determining the state of ball control (*dead ball*, *no possession*, *possession*, or *duel*) for each frame by counting the players in a radius around the ball, e.g. if there are at least two opponents within the radius, the ball is in a state of *duel*. A switch to a different state is what they refer to as a *control frame*. The authors then used a set of rules to determine the event type, e.g. a shot on goal, a cross, or a pass if none of the other

events apply. In the supplementary material they mention that an event is correctly detected if the event is within a ± 10 s window of the manually annotated event, which are tagged by Sportec Solutions (STS). They report a supra 90% precision and recall on the automatically detected passes, but do not make a distinction between the different providers for this metric. These events are predicted on tracking data of 31 matches, supplied by the three different providers; namely Track160 (9), Tracab (19) and Hawk-Eye (3). These providers all have camera installations in the stadia that are used in the competitions and leagues they provide in, thus no broadcast footage is used to generate the tracking data from.

PassNet. PassNet is a solution for automatic pass detection (Sorano, Carrara, Cintia, Falchi, & Pappalardo, 2020). It combines three tasks; feature extraction, object detection and sequence classification. A sequence of video frames is used as input for ResNet18, which reduces the dimensionality while retaining the most important information. In the next task the authors train a Convolutional Neural Network to assign bounding boxes based on a data set containing images of balls and persons, and labelled as such. In the final task the output of both the ResNet18 and the CNN are combined and fed into a Long Short-Term Memory (LSTM) network. The LSTM network is trained to classify the sequence as a pass or not. The authors report an F_1-score of 0.59, a precision of 0.48 and a recall of 0.77 in a scenario where different matches are used for training and testing.

Event and tracking synchronization. There are two potential issues when synchronizing the event data and tracking data. Firstly, manually collecting event time stamps is prone to human error, e.g. reaction time, distractions and decision times delay the time stamp. Secondly, the event data is collected on a different clock than the tracking data, which can be off by a few seconds (Anzer & Bauer, 2021).

The authors developed a synchronization algorithm while working on a goal scoring probability model to combat these two issues. By calculating the distance between the ball and the player, the ball and the event location, and the player and the event location, they could minimize a weighted sum and find the frame where the pass has most likely been given. The algorithm is described in more detail in Section 4.

Individual Ball Possession. Tracking data makes it possible to detect when players are in control of the ball (Link & Hoernig, 2017). The authors developed a machine learning model that is able to determine how long the ball is under influence by a player based on their distance to the ball combined with the direction of motion, speed and acceleration of the ball. The result of their model is a list of timespans where the player is on the ball. The authors suggest their work could be used to help detect match events automatically.

Hyperparameter optimization. While using high quality data is important for the performance of a model, a significant performance boost can be gained by hyperparameter optimization. It is possible to tune these parameters manually, but this is time consuming and prone to human error. A more efficient way is to use a hyperparameter optimization algorithm, such as grid search (Bergstra & Bengio, 2012) or a Tree-structured Parzen Estimator (TPE) (Ozaki, Tanigaki, Watanabe, & Onishi, 2020). These algorithms can be used to find the optimal parameters for a model.

Popular multi-objective optimization algorithms that require thousands of evaluations to converge are not suitable for real-world applications. They suggest using a TPE algorithm, which is a single-objective Bayesian optimization algorithm altered to utilize tree-structured parzen estimators, which makes it more suitable for real-world applications with limited budget (Ozaki et al., 2020).

Chapter 3

Data

In this chapter, we explore the data sets used to automatically detect passes during a soccer game. We begin by describing the event data set, which contains manually tagged events. Then, we delve into the tracking data set, which contains the positions of all the players and the ball during the match. We also examine the dimensions of the pitch and how the coordinates are defined in the data sets, as well as how the data sets are synchronized. Finally, we take a look at examples of passes in the data set and the features that we use to detect them.

3.1 Event Data

The event data that we are using in this thesis, is gathered by analysts at Sportradar. They watch the matches and tag all the events. Each event covers a ball action of a player, a call by the referee, or other moments in the match. For each event the analysts collects; the type of action, the player that performs the action, the match time of the event, and the location of the event. Additionally, certain event types have several extra properties to specify more details of the event. All the events are collected live during a soccer match and are directly stored in a database, which is then made available to end user products.

The tagging procedure is executed by at least four analysts, which consist of one speaker and one typist per team in the soccer match. This is shown in Figure 3.1. The speaker has their eyes on the television all the time and the typists enters the corresponding key combination in the input tool. This tool maps a keypress to a specific event type. The types of events that are tagged are listed in Appendix A. Note that not all events are on the ball, e.g. `VARChecking` or `Offside`.

There are two approaches to detecting pass events in a soccer match. You could either detect all other

Figure 3.1: *Analysts watching the match and tagging events.*

ball events and categorize the remaining events as passes as Vidal-Codina et al. (2022) did, or you could detect passes directly. In this thesis we will focus on the latter, as it could provide a short-term solution without the need to understand all the other events.

In the future we would like to recognise these other events from tracking data. However, some events are more complex to detect, and simply not feasible for this thesis. For example, a foul is not always clear when it is committed and depends on the referee's interpretation. We could use the fact that a foul is often followed by a free kick, but then we would need to detect the free kick as well.

Figure 3.2 shows the distribution of the most occurring events and shows that the majority of events are passes, which is in line with the findings of Pappalardo et al. (2019), who found that roughly 50% of the events are passes in their Wyscout data set. This makes sense in the context of soccer, as the ball is mostly progressed by passing it to a teammate that is in a better position, or to keep possession of the ball. Without the ball, it is impossible to score a goal, which is reflected in Figure 3.2 where `GoalAttempt` and `Goal` are two of the least occurring events. This is also the reason we focus on the automatic detection of passes in this thesis.

3.2 Tracking Data

Tracking data covers the location of all 22 players, referee, and ball during a soccer match. Depending on the resolution of the tracking data, the sample rate can vary. Most common sample rates are 10 Hz and 25 Hz for tracking data. In this thesis, we will be using tracking data provided by Sportlogiq with a sample rate of 25 Hz, this means that we have a sample every 40 ms, containing the positions of the players, referee, and ball. For a match played over two halves, this sums up to around $140{,}000$ frames.

As our tracking data is generated from broadcast footage of the match, it is likely that not all players are

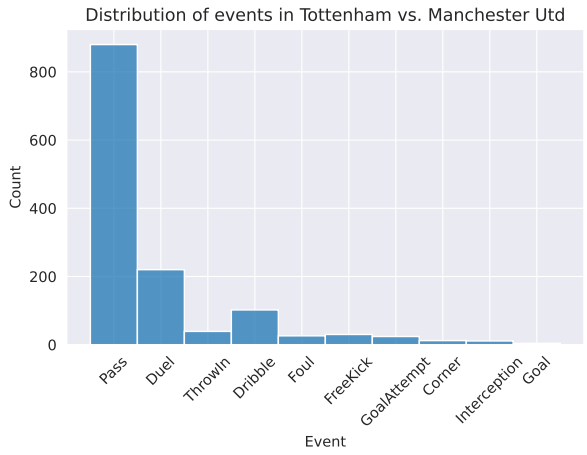

Figure 3.2: *The distribution of events in Tottenham Hotspur–Manchester United (2021–04–11).*

visible in frame. This is especially true for the goalkeeper, as the camera is often focused on the ball and the players near it. In the case this happens, the position is either interpolated based on their previous and next known position, or the location is set to an inferred value based on the player's position (e.g. around the penalty area for a goalkeeper). This preprocessing is already done by Sportlogiq and therefore not part of this thesis.

The tracking data provided consists of roughly $140{,}000$ frames for each match. A single frame, which is taken every 40 ms, contains the positions of the ball, referee, and all the players. The tracking data is stored in a JSON file, which contains a list of frames. Each frame contains the following information:

1. `time` — The time since the phase started in seconds.
2. `ball` — The position of the ball in the frame.
3. `referee` — The position of the referee in the frame.
4. `home_team` — The positions of the players of the home team in the frame.
5. `away_team` — The positions of the players of the away team in the frame.

An example of a single frame is shown in Figure 3.3 where the tracking data is plotted onto a soccer pitch using the standard pitch dimensions of 105 m long and 68 m wide. The home team plays from left to right and is displayed in red, the away team is coloured blue, and the ball black.

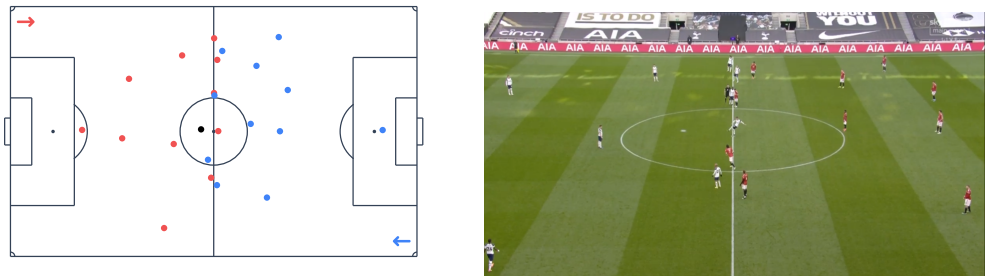

Figure 3.3: *An example of a single frame in the tracking data (left) from Tottenham Hotspur–Manchester compared to broadcast footage (right) (2021–04–11, four frames after kickoff, second half).*

3.3 Data Preprocessing

In order to use both data sets, we need to preprocess them. As the players in the event data set use different IDs than the players in the tracking data, we need to find the matching Sportradar IDs for the latter. The squad numbers of the players within a team are unique and locked once registered during the season. This allows us to match the players in both data sets by their squad number. We also need to synchronize the event and tracking data, as they are collected independently. First, we need to transform the coordinates of the tracking data so that they are in the same coordinate system as the event data.

3.3.1 Coordinate system and pitch dimensions

Sportradar uses a coordinate system in the event data that is agnostic of the pitch dimensions, where the origin is in the bottom left corner of the pitch. This means that the coordinates of the pitch are $(0, 0)$ in the bottom left corner and $(100, 100)$ in the top right corner. However, the tracking data uses a different coordinate system, where $(0, 0)$ is in the center of the pitch and the coordinates are in meters. When the broadcast video is transformed to tracking data, pitch data is supplied to the tracking provider. These pitch sizes vary per stadium, but are usually around 105 m long and 68 m wide.

In order to compare the data sets, we need to transform the coordinates of the tracking data to the same coordinate system as the event data. When plotting the data, we use a standard pitch dimension of 105 m long and 68 m, and convert the coordinates back to meters.

3.3.2 Synchronization of event and tracking data

The time of the event data set could have a systematic offset with the tracking data set as they are collected independently. In addition, there is a delay of human interaction between seeing the event on live on television and tagging it. As the positions added to the event data come from a tablet where the position is selected on a map of the pitch by the analyst, there could be imperfections in the data due to human

error. In order to accurately compare both datasets we need to synchronize them (Anzer & Bauer, 2021). We find the frame where the event occurred in the tracking data by minimizing a sum of features:

1. The distance between the player and the event position.
2. The distance between the ball and the event position.
3. The distance between the player and the ball.

In order to get the best results we could run hyperparameter optimization to get the optimal weights for these features. We can validate all pass detection methods against this corrected event data set. This corrected event data set will be used as the ground truth for all pass detection methods.

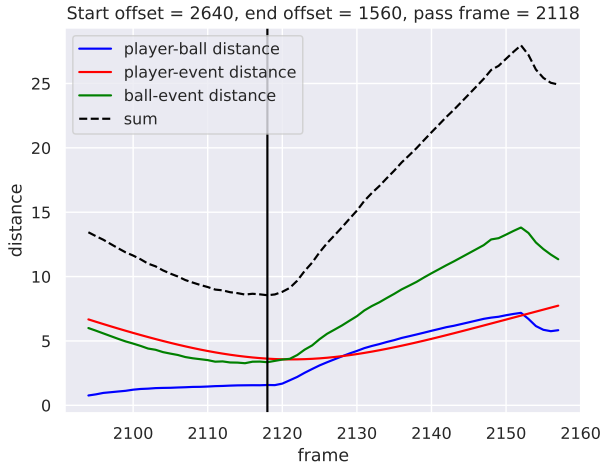

Figure 3.4: *The frame where the pass has most likely been given.*

3.4 Definition of a pass

A pass is defined as an event where a player has control over the ball and attempts to kick the ball to another player on the same team. This definition does not state that the pass has to be successful, but only that the player attempts to pass the ball. It does not matter whether the ball is intercepted by an opponent or if the ball goes out of play. The completion of the pass is determined if the next ball event is from a different player on the same team.

3.5 Matches

The matches used in this thesis are listed in Table 3.1. These matches are selected based on both the availability of the tracking and event data, where the former is more difficult as it is provided by an external party. This is also the reason we use the 2020/2021 season of the English Premier League, as this is the only season where we have access to both data sets.

Match ID	Home Team	Away Team	Date
113989	Brighton	Everton	2021–04–12
113995	Sheffield United	Arsenal	2021–04–11
113996	Tottenham Hotspur	Manchester United	2021–04–11
113999	Arsenal	Fulham	2021–04–18
114005	Manchester United	Burnley	2021–04–18
114008	Wolverhampton Wanderers	Sheffield United	2021–04–17
114015	Manchester City	Southampton	2021–03–10

Table 3.1: *The matches used in this thesis.*

Chapter 4

Methodology

In this chapter we will describe the methodology used to automatically detect passes in soccer tracking data. We will be using two main methods in order to achieve this goal. We start by crafting a model based on rules, which will act as our baseline in the subsequent experiments. In the second part we use neural networks and experiment with different architectures and hyperparameters.

4.1 Rule-based approach

As stated in Section 3.4, a pass is defined as a player controlling the ball and passing it to another player. In order to detect passes, we will use a rule-based approach to capture the behaviour of a pass. The tracking data can tell us which player is closest to the ball and from there we can determine if a pass has been made.

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} \tag{4.1}$$

We start by calculating the distance for each player to the ball for all the frames. For this the Euclidean distance is used as defined in Equation 4.1. All consecutive frames where the same player is closest to the ball are grouped together with their starting and ending frame index. From now on we refer to these as a *sequence*. The rules a sequence must satisfy for it to be considered a pass are:

1. The distance between the player and the ball must be smaller than a certain threshold.
2. The ball must change direction by a certain delta.
3. The ball must accelerate by a certain delta.

4. The sequence must be longer than a certain length in frames.

These rules should capture an interaction with the ball. The first rule is necessary to ensure that the player is within reach of the ball. If a player is not close to the ball, they are unable to interact with it. The exact value will be determined by hyperparameter optimization later in this thesis.

The second rule, whether the ball changes in direction, would imply that the ball has been touched. This is not always the case, as a curved ball changes direction without being touched by a player. As such, we use the delta between the entry and exit direction of the ball. The entry direction is the average direction from the first frame of the sequence to the frame where the distance between the player and the ball is the smallest. The exit direction is calculated from that frame to the last frame of the sequence. The delta between the entry and exit direction must be larger than a certain threshold. This threshold will also be determined by hyperparameter optimization.

The third rule, whether the ball accelerates, is also necessary to ensure that the ball has been touched. The calculation of this rule is in line with previous rule. The entry and exit speed are compared and the resulting delta must be higher than a threshold that is to be determined by means of hyperparameter optimization.

For the last rule, the sequence must be longer than a certain length in frames. This is necessary to filter out noise in the tracking data. The exact value will be, again, determined by hyperparameter optimization. If all these rules apply we consider that a pass had been made within the sequence.

The next step is determining at which frame the pass has been made within the sequence. The most trivial approach would be to minimize the player's distance to the ball and take that frame. However, we noticed that this not always results in the most accurate frames. When a player *pings* a football, which is the art of rapidly passing the ball over a long distance, the time that the player makes contact with the ball is very short. Players also tend to take a touch before passing the ball, which means that the distance increases a bit before the actual pass is made. To combat these issues, we look for the latest occurrence in the sequence where the distance to the ball is underneath a certain threshold. We then walk through the frames backwards from that point until we find a local minimum. An example is shown in Figure 4.1.

If no local minimum can be found, we assume that the player has not touched the ball. These sequences and the ones that do not meet the other criteria are removed from the dataset. This leaves us with a data set of sequences that only contain passes and their predicted frame where the pass has been given.

To get the most out of this model, we optimize the hyperparameters and select the best values for the parameters based on their performance on the validation set.

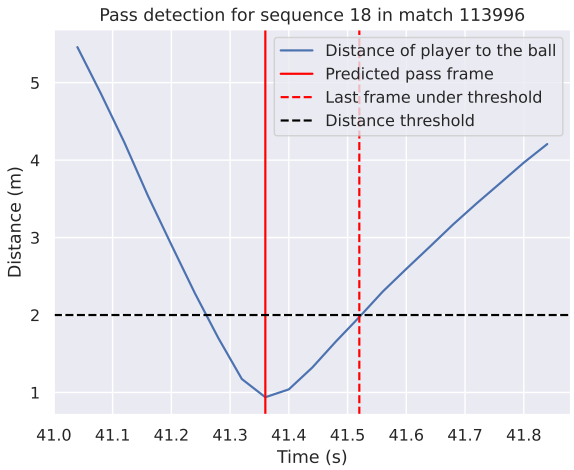

Figure 4.1: *The frame found after walking backwards from the latest occurrence where the distance to the ball is underneath a certain threshold.*

4.1.1 Evaluation

In order to determine if the pass has been detected correctly, we compare it to the event data set. After the events have been corrected and their frame in the tracking data has been found we compare it to the predicted frames. If the predicted frame is within a certain threshold of the corrected frame, we consider it a true positive. Conversely, if the predicted frame is not within the threshold, we consider it a false positive. If there is no predicted frame for a corrected event, we consider it a false negative.

In addition to the threshold, we also add the parameters of the event correction algorithm to the model for hyperparameter optimization.

4.1.2 Hyperparameter optimization

In order to get the most out of our model we use a Tree-structured Parzen Estimator (TPE) algorithm to optimize the hyperparameters. This algorithm is implemented and provided by the `optuna` package (Akiba, Sano, Yanase, Ohta, & Koyama, 2019). The hyperparameters and their search space are shown in Table 4.1.

4.2 Deep learning

In this section, we will discuss the deep learning models that we have used for pass detection. We will start by discussing the Convolutional Neural Network (CNN), followed by the Recurrent Neural Network (RNN).

Hyperparameter	Search space
Rule-based model	
Minimum distance	$[0, 20]$
Minimum length	$[0, 100]$
Minimum direction change	$[0, \pi]$
Minimum speed increase	$[0, 10]$
Prediction evaluation	
Corrected event threshold	$[0, 200]$
Event correction	
Event window start	$[-3000, 0]$
Event window end	$[0, 3000]$
Player/ball weight	$[0, 1]$
Ball/event weight	$[0, 1]$
Player/event weight	$[0, 1]$

Table 4.1: *Hyperparameters and their search space. Split into three categories; rule-based model, prediction evaluation, and event correction.*

We will treat the problem of pass detection as a binary classification problem, as there are two outcomes which the network has to classify; either a pass occurs or it does not. The data set has been created in a sliding window fashion with a window size of 50 frames and a stride of 25 frames. This means the networks see 2 seconds at a time and moves 1 second forward for every sample. The label for the window is 1 if a pass occurs in the window and 0 if it does not. The data set is split into a training, validation and test set.

4.2.1 Features

In order to use the tracking data for pass detection we need to extract features from the data. The features listed in Table 4.2 are used throughout the deep learning experiments. As the deep learning experiments all use a sliding window data set, the features will be extracted for every frame in a window.

The displacement of the ball d, displayed in Equation 4.2, between the current frame t and the next frame $t + 1$ is calculated using the Euclidean distance between the ball in the two frames.

$$d = \sqrt{(x_{t+1} - x_t)^2 + (y_{t+1} - y_t)^2} \tag{4.2}$$

The direction of the ball between the current and the next frame is calculated with the two argument arctangent function. The function takes the difference in y and x coordinates as arguments and considers both signs, which results in the correct quadrant. The function is shown in Equation 4.3, where x_t and y_t

Feature	Description
Ball displacement	The distance between the ball in the current frame and the next frame.
Ball direction	The direction the ball was travelling in between the current frame and the next frame. Calculated using the two argument arctangent function.
Distance to nearest home player	The distance between the ball and the nearest home player in the current frame.
Distance to nearest away player	The distance between the ball and the nearest away player in the current frame.

Table 4.2: *Features used for the recurrent neural network.*

are the x and y coordinates of the ball in the current frame and x_{t+1} and y_{t+1} are the x and y coordinates of the ball in the next frame. The result is an angle (θ) in radians between $-\pi$ and π.

$$\theta = atan2(y_{t+1} - y_t, x_{t+1} - x_t) \tag{4.3}$$

The remaining features are the distances to the nearest player of each team. Like the displacement of the ball, these are calculated using the Euclidean distance between the ball and the player.

4.2.2 Convolutional Neural Network

A convolutional neural network is a type of neural network that is commonly used for image classification. It is able to extract features from an image by applying a convolutional filter. The filter is moved over the image and the dot product is taken between the filter and the image. This results in a feature map and enables the network to extract features from the image.

Spatio-temporal input

We settled on two methods for pass detection using a CNN. In both methods, we use the positions of the home players, away players and the ball as input, but differ in the representation of the temporal input. The first approach is to use a single image with Kernel Density Estimation (KDE) applied to three channels for the positions of the home players, away players and the ball. In the second approach we use a sequence of images as input.

Single image In order to use a convolutional neural network for pass detection, we need to convert the tracking data to an image. KDE is applied to the positions of the home and away players, plus the ball.

These positions are weighted towards the end of the window (see Equation 4.4).

$$weight = \frac{window\_length - frame\_index}{window\_length} \tag{4.4}$$

The resulting KDE is fitted onto a 100×100 grid. This results in an image with three channels; home players, away players and the ball. A sample of the data set is shown in Figure 4.2.

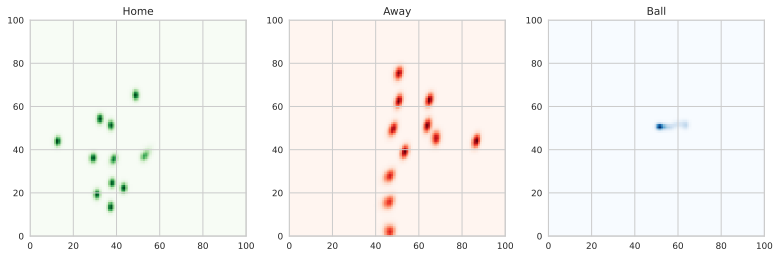

Figure 4.2: *Sample of a heatmap created by KDE.*

Sequence of images In the second approach we will use a sequence of images as input. We set our initial window size to 50, which means we will have a sequence of 50 images. Each frame contains the positions of the home players, away players and the ball fitted onto a 100×100 grid. An example of a single frame from the sequence is shown in Figure 4.3.

The difference with the heatmap approach is that we do not fit a KDE onto the data and there is no weighting applied to the positions. The data we input into the network is the raw positions of the home players, away players and the ball. Which results in more data for the network to learn from.

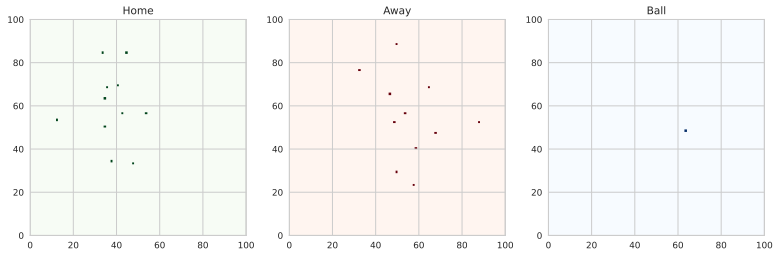

Figure 4.3: *Sample of a single frame from a sequence.*

Additional features In addition to the sequence of images, we will also add the following features with a shape of (50, 4) to the input. These features are also used for the Recurrent Neural Network and are further discussed in Section 4.2.1. The features are:

- The ball displacement between frames.

- The ball direction change between frames.
- The distance between the ball and the nearest home player.
- The distance between the ball and the nearest away player.

Architecture

The architecture we use for the CNN is shown in Figure 4.4. This shows the architecture for the sequence of images. The architecture for the single image is the same, except for the input layer. The input layer for the single image has a shape of (100, 100, 3).

4.2.3 Recurrent Neural Network

A recurrent neural network is commonly used for time series data. It is able to memorize information from previous time steps. However, a simple recurrent neural network might have trouble with long term dependencies. This is where the Long Short-Term Memory (LSTM) network comes in. In a bidirectional LSTM network, the input is processed in two directions in separate recurrent networks. Those networks are connected to the same output layer. Bidirectional LSTM networks show a significant improvement over unidirectional LSTM networks. These networks are also much faster to train than standard RNNs (Graves & Schmidhuber, 2005)

In addition to the LSTM network, we will also experiment with a Gated Recurrent Unit (GRU) network. A GRU network is comparable to an LSTM network in terms of performance. However, which network performs better depends on the data set and the task. Both networks are able to learn long term dependencies and are superior to a standard RNN (Chung, Gulcehre, Cho, & Bengio, 2014).

These recurrent neural networks are trained on the same data set as the other experiments using only the features described in Section 4.2.1. The input shape for the recurrent neural network is (50, 4) as a window size of 50 is used with a stride of 25.

Architecture

The architecture we use for the RNN is shown in Figure 4.5. The two LSTM layers are bidirectional. We also experiment with a GRU layer and a simple RNN layer, which replaces the LSTM layer.

Hyperparameter Optimization

As with the rule-based model, we will optimize the hyperparameters of the LSTM model. The hyperparameters we will optimize are shown in Table 4.3.

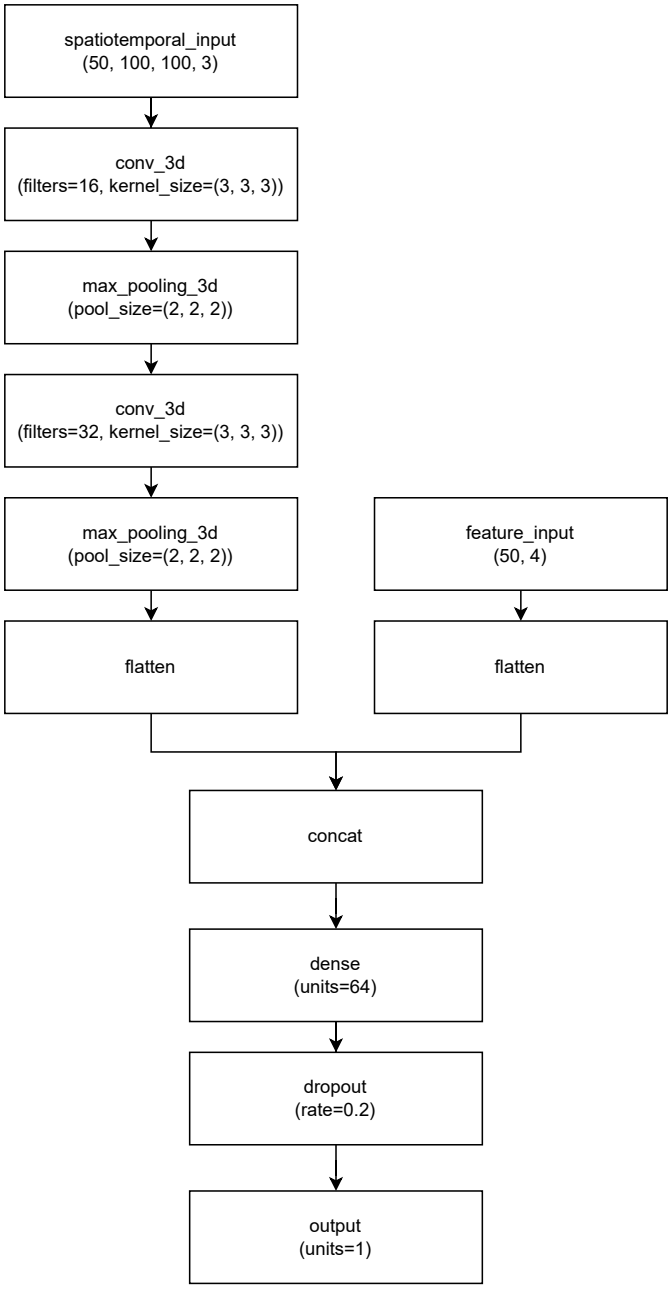

Figure 4.4: *Architecture of the convolutional neural network.*

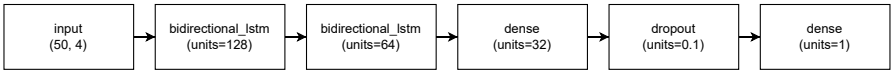

Figure 4.5: *Architecture of the recurrent neural network.*

Hyperparameter	**Search space**
Number of layers	$[1, 2]$
Number of units	$\{128, 256\}$
RRN type	{LSTM, GRU, SIMPLE RNN}
Dropout	$[0, 0.5]$
Recurrent dropout	$[0, 0.6]$
Bidirectional	{True, False}

Table 4.3: *Hyperparameters and their search space for the recurrent neural network.*

Chapter 5

Experiments & Results

The results of the experiments are presented in this chapter. As described in the Chapter 4 we developed a rule-based system, and trained multiple deep neural networks, which we will compare to each other. First, the results of the event and tracking data synchronization are presented, then the results of the rule-based system for automatic pass detection are shown. Finally, the results of the neural networks, including the CNN and the RNN, are presented.

5.1 Evaluation

For each sequence we look for the frame where the pass has been given, but it also possible that there is no pass in the sequence. In order to evaluate the model, we create a confusion matrix and calculate the precision, recall, and F_1-score. Precision is the fraction of correctly predicted passes over all predicted passes, and is defined as follows:

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

Recall is the fraction of correctly predicted passes over all actual passes, and is calculated as follows:

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

The F_1-score is a harmonic mean of the precision and recall. As it takes both the false negatives and false positives into account, it is a good measure for a imbalanced data set. The F_1-score is defined as follows:

$$F_1 = 2 \times \frac{\mathit{precision} \times \mathit{recall}}{\mathit{precision} + \mathit{recall}}$$

A simple classification accuracy is often a poor metric for measuring performance (Fawcett, 2006), which is why we will use a receiver operating characteristic (ROC) curve to evaluate the model. The ROC curve is a two-dimensional plot in which the true positive rate (TPR) is plotted on the y-axis against the false positive rate (FPR) on the x-axis. The TPR is the same as recall, while the FPR is the fraction of incorrectly predicted passes over all negative samples.

The area under the curve (AUC) is a measure of how well the model is able to distinguish between the classes. A perfect classifier would have an AUC of 1.0, while a random model would have an AUC of 0.5. This random classifier is represented by the diagonal dashed line in our ROC curves.

$$\mathit{TPR} = \frac{\mathit{true\ positives}}{\mathit{true\ positives} + \mathit{false\ negatives}}$$
$$\mathit{FPR} = \frac{\mathit{false\ positives}}{\mathit{false\ positives} + \mathit{true\ negatives}}$$

5.2 Event time correction

Due to the fact that the tracking data is collected with a different clock than the event data, the event times have a systematic offset from the tracking data. We correct these times by finding the frame where the sum of the distances between the ball and the player, the player and the event location, and the ball and the event location is smallest. An example where these features are plotted, is shown in Figure 3.4.

Multiplying the time of an event by the framerate results in a naive frame number. However, the event time is not necessarily the time of the pass, but the time of the pass event. This means that the ball could have been passed before the event was tagged. In order to find the frame where the ball was passed, we combine the event time with the tracking data by summing the previously mentioned features. The error between the event time before and after the correction is calculated as follows:

$$\mathit{error} = (\mathit{event\ time} \times \mathit{frame\ rate}) - \mathit{pass\ frame}$$

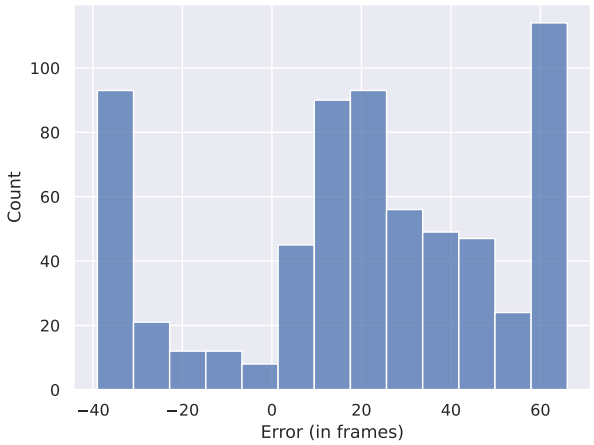

Figure 5.1: *The error between the event times and the found pass frames.*

There were 664 passes—in the Tottenham Hotspur vs. Manchester United match—that could be found in the tracking data, with a mean error of 19.97 frames and a standard deviation of 32.55 frames. The distribution of the errors is shown in Figure 5.1. Note that there are peaks on the edges of the distribution. This could be due to the fact that the same player had an event before or after the pass event, which could have caused the algorithm to match the wrong event. The mean error of 19.97 frames is under a second, which is acceptable for our purposes.

5.3 Rule-based model

The rule-based model has been developed to automatically detect passes in the tracking data. In this method we find sequences where the same player is nearest to the ball and determine based on the rules described in Section 4.1 if a pass has been given.

As there is no learning involved in the algorithm itself, but only during hyperparameter optimization (HPO), we could get away with a smaller training set on which HPO is performed. The smaller training set leads to multiple benefits, first of all, due to only having the two matches running HPO is faster, and secondly, we can find out if the model is able to generalize to unseen data, as the test set is not used during HPO. The split of the matches into training and test sets is shown in Table 5.1.

Table 5.2 shows the results of the classification in a confusion matrix on the entire test data set. The model has an F_1-score of 0.75 and is able to detect $3{,}759$ passes from the Sportradar event data set, but fails to find the remaining $1{,}128$ passes. The mean absolute error (MAE) on the predicted frame for a sequence is 21.81 frames. This means that the events tagged by Sportradar are on average 0.87 s away from the

Match ID	Home Team	Away Team	Date
	Training set		
113989	Brighton	Everton	2021–04–12
113995	Sheffield United	Arsenal	2021–04–11
	Test set		
113996	Tottenham Hotspur	Manchester United	2021–04–11
113999	Arsenal	Fulham	2021–04–18
114005	Manchester United	Burnley	2021–04–18
114008	Wolverhampton Wanderers	Sheffield United	2021–04–17
114015	Manchester City	Southampton	2021–03–10

Table 5.1: *The split of the matches into training and test sets for the rule-based model.*

predicted pass frame.

		True	
		Pass	No Pass
Predicted	Pass	3,759	1,418
	No Pass	1,128	0

	Score
MAE	21.81
Precision	0.73
Recall	0.77
F_1-score	0.75

Table 5.2: *Confusion matrix on the test data set*

5.3.1 Hyperparameter optimization

The F_1-score is used as the objective function during hyperparameter optimization. After running more than 500 iterations, we receive the hyperparameters listed below. These hyperparameters have been used for the previously reported results of the rule-based model. The hyperparameters used for the event correction are frozen to the values listed in Table 5.3, and will be used for labelling the data set for the deep learning model.

The optimization history of the hyperparameter optimization is shown in Figure 5.2. Within a few trials a passable model is found, however there is slight improvement over the next 500 iterations. The best parameters were found at trial number 375, which has an F_1-score of 0.75. The entire optimization took over 5 hours and 40 minutes, in which 528 iterations have been run.

The hyperparameter importance is estimated using a functional ANOVA framework (Hutter, Hoos, & Leyton-Brown, 2014), which is included in the `optuna` package (Akiba et al., 2019). It fits a random forest model on the results of the hyperparameter optimization and estimates the importance of both the interaction between the hyperparameters and the hyperparameters individually. The results are shown in

Hyperparameter	Value
Rule-based model	
Minimum distance	2.03
Minimum length	10
Minimum direction change	0.09
Minimum speed increase	9.58
Prediction evaluation	
Corrected event threshold	200
Event correction	
Event window start	−1,880 ms
Event window end	140 ms
Ball/event weight	0.44
Player/event weight	0.33
Player/ball weight	0.23

Table 5.3: *Hyperparameters and their found values.*

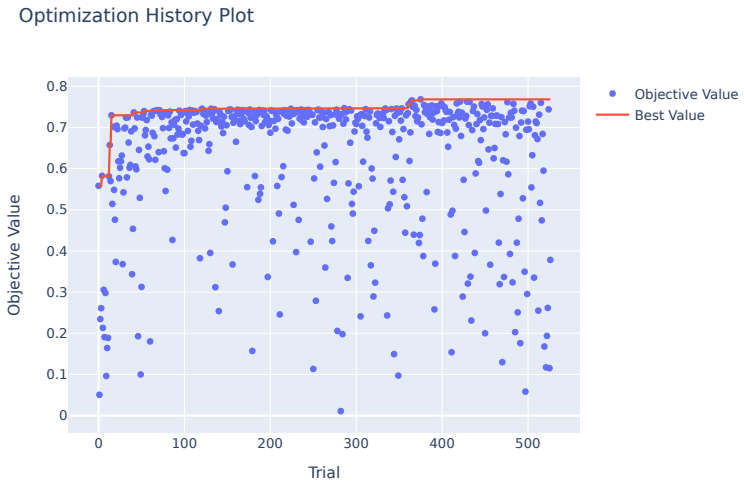

Figure 5.2: *The optimization history of the hyperparameters.*

Figure 5.3, which shows that the most important hyperparameters are the minimum direction change and the minimum length of a sequence.

The relation between the most important hyperparameters is shown in a parallel coordinate plot in Figure 5.4. The figure shows that while the minimum direction threshold is important, it is important to be close to zero in order to get a good result. The exit speed increase was not important at all as shown in Figure 5.3, this is also reflected in the parallel coordinate plot where a wide range of values is used which

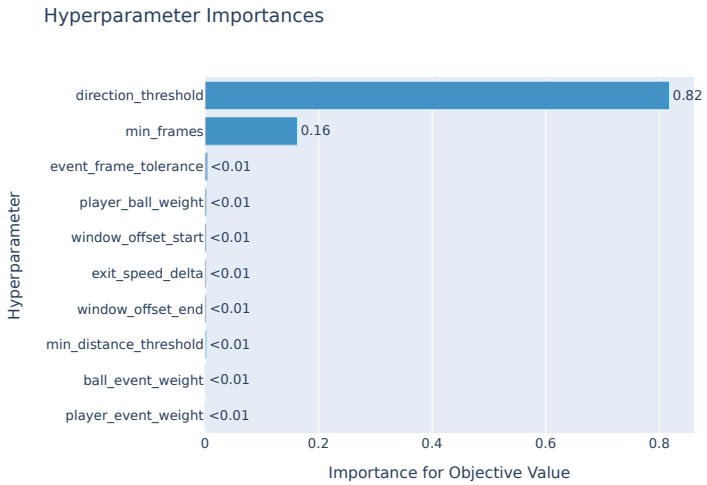

Figure 5.3: *The importance of the hyperparameters.*

all result in a similar objective value. That is not the case for the minimum frame length, which seems to be important for values between 10 and 30.

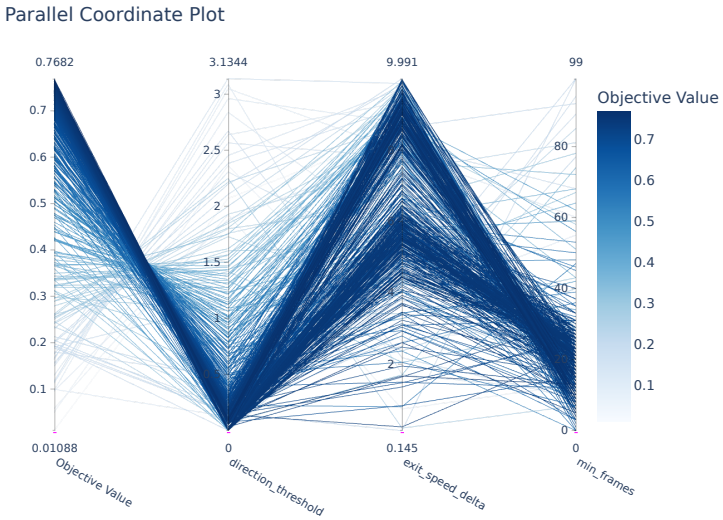

Figure 5.4: *The relation between the most important hyperparameters.*

5.4 Deep Learning

In this section we will discuss the results of the deep learning models. We will start with the Convolutional Neural Network (CNN) and then move on to the Recurrent Neural Network (RNN). All the results shown are produced using the same test set, which consists of one match. The validation set contains two matches and the training set contains the remaining four matches. The split that has been made is shown in Table 5.4.

For the experiments we will be going through the tracking data in a sliding window fashion. The window size is set to 50 frames and the stride is set to 25 frames. These parameters are chosen based on the results of the hyperparameter optimization of the rule-based system.

Match ID	Home Team	Away Team	Date
	Training set		
113989	Brighton	Everton	2021–04–12
113995	Sheffield United	Arsenal	2021–04–11
113996	Tottenham Hotspur	Manchester United	2021–04–11
113999	Arsenal	Fulham	2021–04–18
	Validation set		
114005	Manchester United	Burnley	2021–04–18
114008	Wolverhampton Wanderers	Sheffield United	2021–04–17
	Test set		
114015	Manchester City	Southampton	2021–03–10

Table 5.4: *The split of the matches into training, validation and test set.*

5.4.1 Convolutional Neural Network

As described in the previous chapter, we experiment with two different types of input. In the first experiment we use the heatmaps as input, and in the second we use the raw positions to generate an image for each frame. In the third experiment we apply an ablation study to the second experiment to see if the image representation is necessary. For these experiment we will be going over the tracking data in a sliding window fashion, with a window size of 50 and a stride of 25.

Heatmaps

In the first experiment a heatmap is used as input, where the positions are plotted onto different channels of the image. Temporality is added by weighting their positions by the time they occur in the window.

Figure 5.5 shows the learning curve of the CNN over 20 epochs. The loss jumps around a lot on the val-

idation data, which is not the case for the training loss. The training loss shows little movement and decreases over time to 0. If we take a look at the accuracy, we see that we reach 1.0 when training, but validation hovers around 0.5. These are all signs over overfitting. The model does not generalize well to newly seen data.

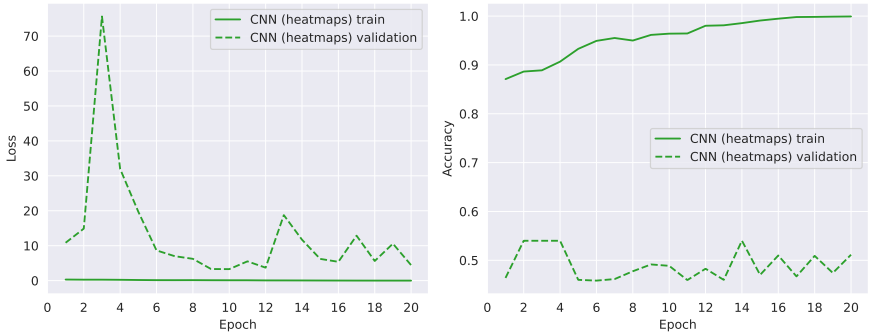

Figure 5.5: *The learning curve of the CNN trained on heatmaps.*

		True	
		Pass	No Pass
Predicted	Pass	1,215	1,019
	No Pass	630	697

	Score
Precision	0.54
Recall	0.66
F_1-score	0.60
Accuracy	0.54

Table 5.5: *Confusion matrix and scores of the CNN (heatmap) predictions*

Table 5.5 shows the classification report of the CNN trained on heatmaps. The model has a precision of 0.54 and a recall of 0.66. The F_1-score is 0.60 and the accuracy is 0.54. These results are only slightly better than a random classifier.

Window of Frames

In the second experiment we change the input for the model. The previous experiment used heatmaps to train the model on, these are substituted with an image per sequence in the frame. They are fairly similar to the heatmaps, however they only contain data for a single frame and are not fitted with Kernel Density Estimation. In addition to the window of images we also add four features to the model as described in Section 4.2.2.

Figure 5.6 shows the learning curve of the CNN trained on sequences of images. After two epochs the model has converged, after which learning has almost no impact on the validation loss anymore. This is reflected in the accuracy, for which the accuracy on the training set keeps improving, while the accuracy on the validation set stagnates.

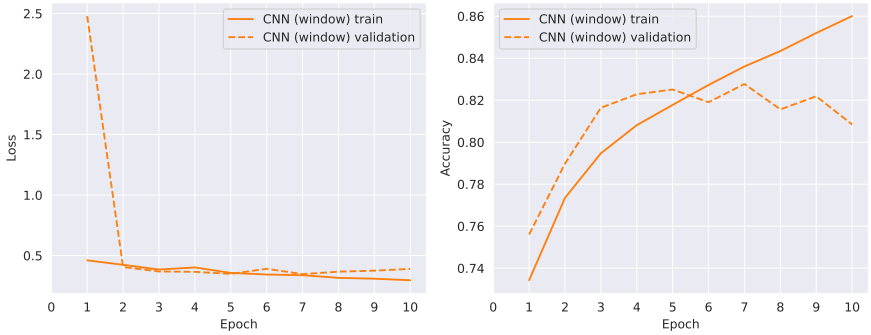

Figure 5.6: *The learning curve of the CNN trained on windows.*

The confusion matrix of the predictions on the test set for the CNN trained on sequences of images is shown in Table 5.6. The model has a precision of 0.68 and a recall of 0.67, the F1 score is 0.68 and the accuracy is 0.80. The F_1-score is better than the one of the CNN trained on heatmaps, but is this due to the fact that the image representation is better or do the additional features carry the model?

		True	
		Pass	No Pass
Predicted	Pass	1,047	486
	No Pass	518	3,058

	Score
Precision	0.68
Recall	0.67
F_1-score	0.68
Accuracy	0.80

Table 5.6: *Confusion matrix and scores of the CNN (window) predictions*

Ablation Study

In the third experiment we perform an ablation study on the CNN trained on the sequences of images. We remove the convolutional layers from the network and train only on the feature input, this results in a Dense Neural Network (DNN). The network is then trained for 50 epochs on the same data set, without the convolutional input. The learning curve for this network is shown in Figure 5.7.

The results on the test set are shown in Table 5.7. The model has a precision of 0.72 and a recall of 0.59, the F1 score is 0.65 and the accuracy is 0.80. The DNN has similar results to the CNN trained on sequences of images. This means that the image representation does not add much to the model. In fact, the DNN does not look fully converged yet after 50 epochs, so it might be able to improve further. Interesting to note is that the validation accuracy is higher than the training accuracy. However, the accuracy of the predictions on the test set is lower than the training accuracy.

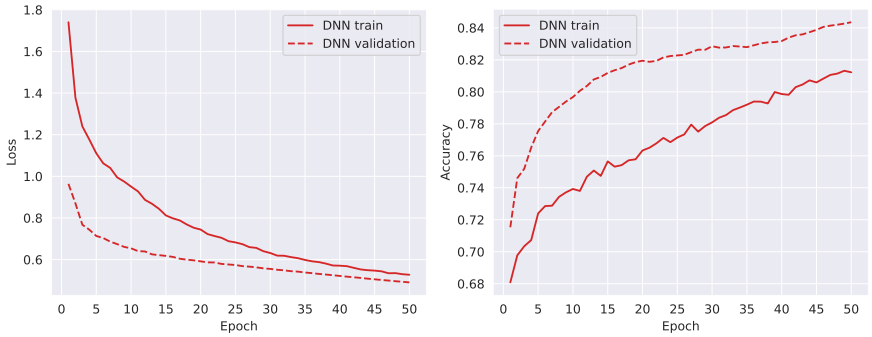

Figure 5.7: *The learning curve of the DNN trained on the features.*

		True	
		Pass	No Pass
Predicted	Pass	923	358
	No Pass	642	3,186

	Score
Precision	0.72
Recall	0.59
F_1-score	0.65
Accuracy	0.80

Table 5.7: *Confusion matrix and scores of the DNN predictions*

5.4.2 Long Short-Term Memory

At last, we experiment with a Recurrent Neural Network (RNN). The RNN is trained on the same features that were used for the CNN (windows). The network consists of two LSTM layers containing 256 and 128 units respectively, both with a recurrent dropout of 0.4. The network has been trained for 30 epochs, after which early stopping is applied.

The results on the test set are shown in Table 5.8. The model is able to predict 87% of the samples in the test set correctly. The precision on the passes is 0.84, while the recall is fairly lower at 0.72. The F_1-score is 0.78.

		True	
		Pass	No Pass
Predicted	Pass	1,132	214
	No Pass	433	3,330

	Score
Precision	0.84
Recall	0.72
F_1-score	0.78
Accuracy	0.87

Table 5.8: *Confusion matrix of the LSTM predictions*

The learning curve of the model is shown in Figure 5.8. After 10 epochs the learning speed declines, but the model keeps slightly improving. The validation loss is lowest at epoch 25, after which it starts increasing again. The network has been trained for 5 more epochs after this point, but the validation loss does not

decrease anymore. The accuracy seems to reflect this as well, as it does not improve after epoch 25.

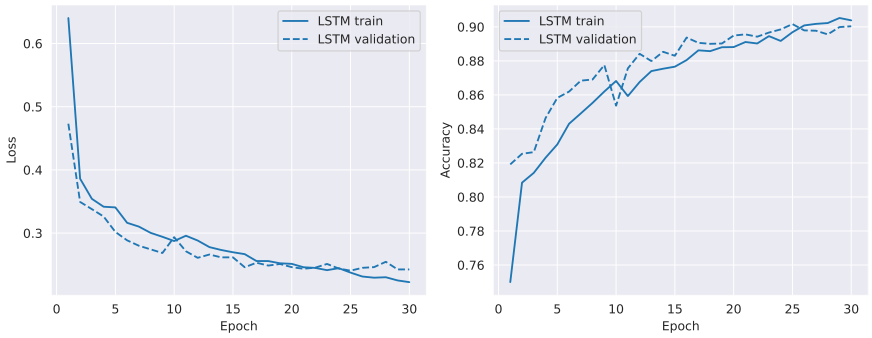

Figure 5.8: *The learning curve of the LSTM.*

Hyperparameter optimization

The results shown above are achieved after hyperparameter optimization. The parameters and their objective value are shown in Appendix B. In the first experiment we look for the differences in performance between the LSTM, GRU and a simple fully-connected RNN layer. The latter performs significantly worse than the other two.

We also find that the difference between using one and two layers for the LSTM and GRU is negligible. After running another experiment with bidirectional layers, we find that the performance of the LSTM is slightly better. The final parameters we use for the LSTM are shown in Table 5.9.

Hyperparameter	Value
Number of layers	2
Number of units	128 in the first layer, 64 in the second layer
RNN type	LSTM
Dropout	0.1
Recurrent dropout	0.4
Bidirectional	True

Table 5.9: *Hyperparameters for the recurrent neural network.*

5.5 Comparison

In this section we will compare the results of the different models; the CNN, LSTM and the rule-based system. Figure 5.9 shows the ROC curve of the models. The LSTM performs best with an AUC of 0.94, followed by the CNN trained on windows of images with 0.88. After running the DNN multiple times,

we find that the performance is consistent, but always has a horizontal line in the ROC curve, which means that at that certain threshold a lot of false positives are introduced. The CNN trained on heatmaps performs the worst of all the deep learning methods with an AUC of 0.54.

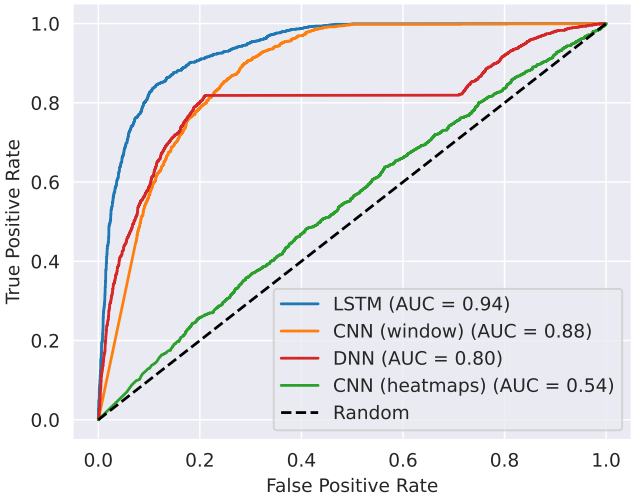

Figure 5.9: *ROC Curve of the Neural Networks*

The results of all the models are visible in Table 5.10. The LSTM performs the best, with the rule-based system coming in second. The LSTM has a better precision (0.84) compared to the rule-based system (0.71), but a lower recall (0.72) compared to the rule-based system (0.80). In the end, the F_1-score of the LSTM is slightly higher (0.78) than the rule-based system (0.77).

	Precision	**Recall**	F_1**-score**
Rule-based system	0.73	0.77	0.75
CNN (heatmaps)	0.54	0.66	0.60
CNN (windows)	0.68	0.67	0.68
DNN	0.72	0.59	0.65
LSTM	0.84	0.72	0.78

Table 5.10: *Precision, recall and F1 score of the different models.*

Chapter 6

Discussion

In this section we discuss the results found during the experiments in the previous chapter. We ran experiments on both a rule-based system and a variety of neural networks. These methods will be compared to each other and we will discuss which neural network performed best. We will also discuss the quality of the data set and how this affects the results.

6.1 Rule-based system

We present a rule-based system that is able to automatically recognize most of the passes in a soccer match after hyperparameter optimization with an F_1-score of 0.75. Most of the hyperparameters do not seem to influence the objective value, the F_1-score, however the minimum direction change and the minimum length of sequences do. The best value for the minimum direction change was 0.09, which is likely due to the fact that there is no directional change when the ball is not touched by the player. The ball must be touched by a player in order to pass it, whether this pass is made direct or the ball controlled first, therefore there is a directional change before or when a pass is made. This results in a value that filters out the sequences where the nearest player did not touch the ball.

The speed increase when a ball is passed does not seem important for the performance of the model, as a wide range of values give good objective values, thus can be concluded that the minimum direction threshold is a better indicator of when the ball has been passed. The minimum number of frames in a sequence is best around 10. This is likely necessary for it to filter out noise, while being short enough to capture the majority of passes.

6.2 CNN

The representation used in the CNN has not been proven to work. Using the heatmaps, we condensed the spatio-temporal data into one image with KDE applied to three channels; both the teams, and the ball. However, the network did not seem to converge and the loss on the validation set grew large quickly. The accuracy on the validation set hovered around 0.5 without any significant improvements during the training. This might be due to the fact that the representation is not suitable for this task. The same was visible to an extent in the CNN trained on windows and the additional features. This time, the accuracy on the validation set was around 0.8 without any signs of learning during the training process.

In the next experiment we removed the image representation and only used the features extracted from the tracking data; the ball displacement, direction of the ball, and the distance to the nearest players of both the teams. This time, the network showed it was able to learn from the data, but achieved similar results to the CNN with the windows and the additional features. This confirms that the image representation is not suitable for this task and unnecessary.

6.3 LSTM

The Long Short-Term Memory network gives the best results out all of our experiments. The data set is easier to handle than the CNN as it does not contains any images, thus is also easier to train. While the accuracy is 89%, it is not directly comparable to the rule-based system. The rule-based system only looks at moments where a pass could potentially occur, and the LSTM evaluates all the windows in the data set for a window size and stride. This means that the accuracy of the LSTM will be higher as it contains the more common negative samples as well. The LSTM is also able to capture passes that are not captured by the rule-based system. This is because the rule-based system is limited by the rules defined. If we compare the F_1-score of the LSTM with the rule-based system, we see that the LSTM scores 0.78 while the rule-based system scores 0.75. As it also could learn from the negative cases, it is able to capture slightly more passes than the rule-based system.

The features of a few predictions of the LSTM are plotted in Figure 6.1. In the first example the home player made at pass at around 20 frames into the window, which it predicted correctly. The second example also shows a pass, but this time it is right at the end of the window, of which it has trouble predicting indicated by the lower confidence. The third example shows a negative sample, where no pass was made.

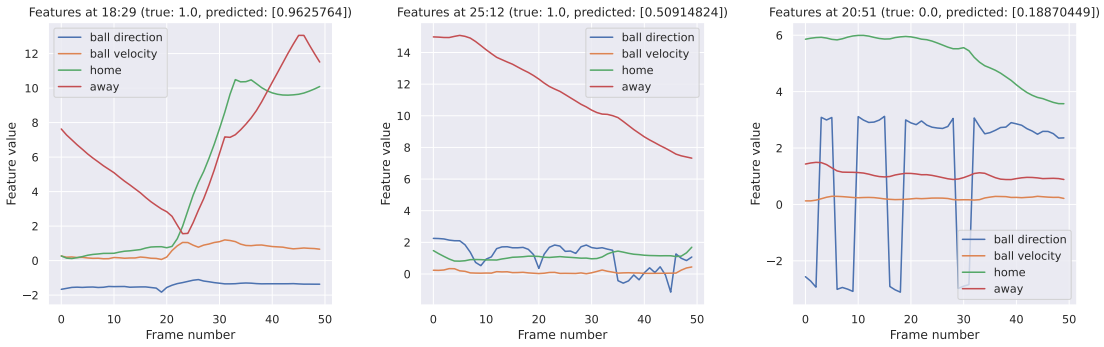

Figure 6.1: *Three samples and their predictions of the LSTM.*

6.4 Data quality

The results found in the experiments depend on the quality of the data used. This might be the most important factor in succeeding in this task. If the event data set is not complete, we will not be able to find all passes in the tracking data. However, the other way around is also true, if the tracking data lacks precision or contains inconsistencies, the rules defined will not be able to capture the passes. This happens when the ball is not tracked correctly, or when the players are not tracked correctly. This can be caused by occlusion, or if some players are swapped in the tracking data.

Due to the fact that we use tracking data generated from broadcast video, there are instances where a replay is shown or the camera is zoomed in on a player while the ball is brought back into play via a throw-in or goal kick. As the ball or players are not visible on screen, assumptions are made where these could be. The interpolation of the tracking data is done by Sportlogiq, and is currently out of scope for this thesis. However, increasing the quality of the tracking data will likely increase the performance of the methods presented in this thesis.

Chapter 7

Conclusion

In this thesis, we developed two distinct methods for automatic pass detection, a rule-based system and a neural network. We found that the rule-based system was able to recognize most of the passes in a soccer match with an F_1-score of 0.75. The passes that were not recognized were either passes that did not show up on the broadcast video, or due to lack of quality in the tracking data. The LSTM was able to classify 87% correctly on a data set with a window size of 50 and a stride of 25. It had an F_1-score of 0.78 on the test data set.

The other neural networks that were experimented with did not perform as well as the LSTM. The CNN trained on heatmaps did seem to converge, and had a huge validation loss throughout the learning process. The CNN trained on sequences had the same problem, although less severe. In the ablation study where we removed the image representation from the CNN, we found that the network performed better. This suggests that the image representation is not necessary and only hindered the network. This finally led to the LSTM, which performed the best out of all the neural networks, and was able to capture passes that were not captured by the rule-based system.

We set out to answer if it is possible to automatically detect passes in soccer using tracking data, and if so, which method performs best. We found the rules that make it possible to detect passes in tracking data, most notably the change in direction when a pass is given, and the speed boost given to the ball. We also found that the LSTM was able to predict more passes correctly than the rule-based system, which suggests that there is room for improvement in the latter. In the end, we were not able to match the figures from Vidal-Codina et al. (2022), however it is likely that the difference in performance is due to the difference in tracking data quality. The tracking data used in this thesis was collected from broadcast video, while the tracking data used by Vidal-Codina et al. (2022) was collected from dedicated cameras

in the stadia.

In future work we like to explore different tracking data sets. The quality of the tracking data is important for the performance of these methods and advancements already have been made during the course of this thesis. As the tracking data used was collected from broadcast video it would be interesting to see how the methods perform on tracking data collected from wearables or multiple in-stadium cameras. This would resolve an issue where there is no tracking data available during replays or where the live feed is zoomed in on a single player.

Another direction for future work is to explore different events. While there has been some work on different events using a decision-tree algorithm (Vidal-Codina et al., 2022), it would be interesting to see if a neural network can capture more complex and less obvious events. For example, the detection of a foul is not always clear and depends on the referee's interpretation. However, a neural network might be able to learn the distinction between a foul and a normal tackle with additional video data where it might for example learn the referee's gestures, the outcome and the reaction of the players.

Finally, we could leverage the tracking data for more complex statistics. For example, we could use the tracking data to detect players offering themselves to receive a pass, and use this to calculate the space created by a player—or use it to calculate the statistics envisioned by Arsène Wenger in the FIFA Football Language—such as opposition line breaks, the pressure applied to a ball carrier, and when a team loses the ball due to pressure (Wenger, 2021).

Appendix A

Event Types

Name	Description
StartPhase	The start of a new phase in the match.
EndPhase	The end of a phase.
Goal	A goal has been scored.
OwnGoal	An own goal has been scored.
YellowCard	A yellow card has been given.
RedCard	A red card has been given.
SubIn	A player has been substituted in. This event is always paired with SubOut.
SubOut	A player has been substituted out. This event is always paired with SubIn.
OutOfPlay	The ball went out of play.
Foul	A player has been fouled.
Pass	The ball has been passed. Pass completions depends on the next event.
Reception	The ball has been received.
GoalAttempt	An attempt to score has been made.
SaveOnGoalAttempt	The goal attempt has been saved.
Interception	The ball has been intercepted.
GoalKick	A corner kick has been taken.
ThrowIn	The ball has been thrown back into play.
Penalty	A penalty has been taken.
FreeKick	A free kick has been taken.
Touch	A touch has been taken.
Dribble	A dribble is annotated when the player runs more than 5m with the ball.
Clearance	An action is labeled as clearance if the player clears the ball without having the intention to reach a team mate.
DefensiveBlock	The ball has been blocked.
Duel	The ball is in contest between two players.
Offside	Offside has been given.
FreeKickOnGoal	Free kick has been taken and was on target.
DefensivePositioning	A defensive action without touching the ball.
PlayerOutOfPlay	A player is out of play.
PlayerBackInPlay	A player has come back into play.
VARChecking	The VAR is checking.
VARRefereeChecking	The referees themselves checks the VAR images.
VARDecision	A decision has been made by the VAR.

Table A.1: *The different types of events that are tagged in the Sportradar data set.*

Appendix B

RNN Hyperparameter Optimization

num_units	dropout	optimizer	num_rnn_layers	rnn_type	recurrent_dropout	Accuracy
256	0.1	adam	2	lstm	0.2	0.792
256	0.1	adam	2	lstm	0.4	0.787
256	0.1	adam	1	lstm	0.4	0.781
128	0.1	adam	2	lstm	0.4	0.781
256	0.1	adam	2	gru	0.2	0.78
128	0.1	adam	2	lstm	0.3	0.779
128	0.1	adam	2	gru	0.2	0.779
256	0.1	adam	1	gru	0.2	0.775
256	0.1	adam	1	lstm	0.3	0.774
256	0.1	adam	1	lstm	0.2	0.773
256	0.1	adam	2	gru	0.3	0.773
128	0.1	adam	2	gru	0.4	0.773
128	0.1	adam	1	gru	0.4	0.772
128	0.1	adam	2	lstm	0.2	0.772
128	0.1	adam	1	gru	0.3	0.769
128	0.1	adam	1	gru	0.2	0.769
128	0.1	adam	1	rnn	0.3	0.768
256	0.1	adam	2	gru	0.4	0.768
128	0.1	adam	2	gru	0.3	0.764
256	0.1	adam	1	rnn	0.2	0.764
128	0.1	adam	1	lstm	0.3	0.762
256	0.1	adam	1	rnn	0.3	0.761
128	0.1	adam	1	lstm	0.2	0.761
128	0.1	adam	1	rnn	0.2	0.76
256	0.1	adam	2	lstm	0.3	0.757
128	0.1	adam	1	rnn	0.4	0.755
128	0.1	adam	2	rnn	0.4	0.749
256	0.1	adam	2	rnn	0.2	0.749
256	0.1	adam	1	gru	0.4	0.748
256	0.1	adam	1	rnn	0.4	0.745
128	0.1	adam	2	rnn	0.2	0.744
128	0.1	adam	2	rnn	0.3	0.744
128	0.1	adam	1	lstm	0.4	0.742
256	0.1	adam	2	rnn	0.4	0.737
256	0.1	adam	1	gru	0.3	0.718
256	0.1	adam	2	rnn	0.3	0.686

Table B.1: *Hyperparameter optimization results.*

rnn_type	recurrent_dropout	bidirectional	Accuracy
lstm	0.4	1	0.821
gru	0.4	1	0.799
gru	0.4	0	0.76
lstm	0.4	0	0.755

Table B.2: *Hyperparameter optimization results for directionality on a network with two layers of* 256 *and* 128 *units.*

Appendix C

Dashboard

During the course of this thesis a platform has been developed in to order to get a better understanding of the rule-based system. The dashboard is built using React, and the soccer pitch is plotted using the d3.js library. This made it possible to make it fully interactive. Hovering over a player shows their ID, name and squad number (see Figure C.1).

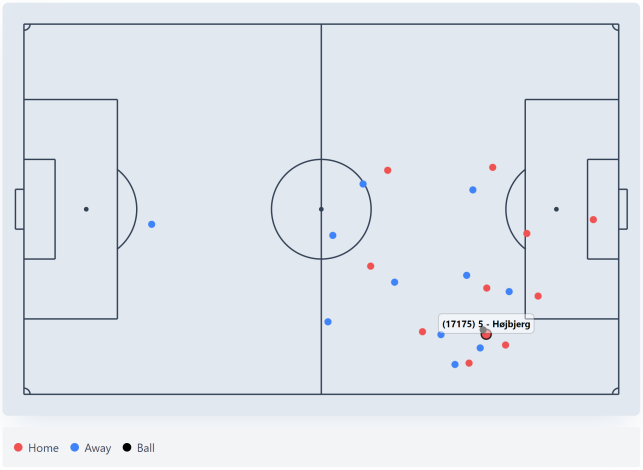

Figure C.1: *Hovering over a player shows their ID, name and squad number*

The full dashboard is shown in Figure C.2. After selecting a match, the sequences found by the rule-based system are displayed on the right. These are interactable, and clicking on a sequence will show the corresponding pass in the soccer pitch. The sequences are colour coded based on their occurrence in the rule-based predictions and the ground truth. The sequences that are correctly predicted have a green check mark. If the rule-based system predicts a pass that is not in the event data set, a yellow question

mark is shown. The sequences that are in the event data set, but are not found with the rule-based system are marked with a red cross.

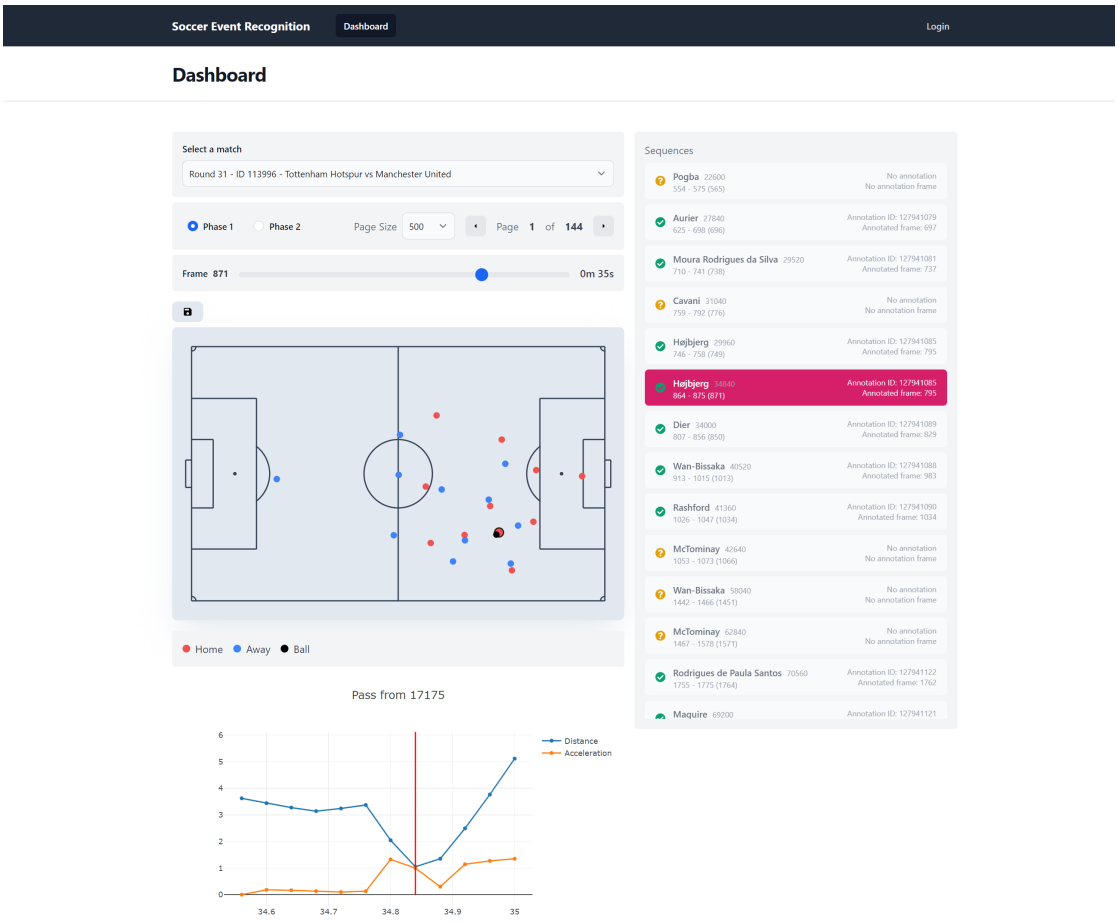

Figure C.2: *Dashboard*

By moving the selected frame slider around and keeping an eye on the soccer pitch, we can see the ball and players move around the pitch. This is useful when trying to understand why a certain pass was not detected by the rule-based system. In our findings we see that the ball sporadically disappears from the tracking data, comparing this with the video data of the match shows that this either is due to the ball being out of frames, or it disappears in the empty stands during COVID-19 matches.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019, July). Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2623–2631). Anchorage AK USA: ACM. Retrieved 2023-08-23, from https://dl.acm.org/doi/10.1145/3292500.3330701 doi: 10.1145/3292500.3330701

Anzer, G., & Bauer, P. (2021). A Goal Scoring Probability Model for Shots Based on Synchronized Positional and Event Data in Football (Soccer). *Frontiers in Sports and Active Living*, *3*. Retrieved 2023-03-10, from https://www.frontiersin.org/articles/10.3389/fspor.2021.624475

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014, December). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. arXiv. Retrieved 2023-06-15, from http://arxiv.org/abs/1412.3555 (arXiv:1412.3555 [cs])

Fakhar, B., Rashidy Kanan, H., & Behrad, A. (2019, June). Event detection in soccer videos using unsupervised learning of Spatio-temporal features based on pooled spatial pyramid model. *Multimedia Tools and Applications*, *78*(12), 16995–17025. Retrieved 2022-10-01, from https://doi.org/10.1007/s11042-018-7083-1 doi: 10.1007/s11042-018-7083-1

Fawcett, T. (2006, June). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. Retrieved 2023-05-30, from https://www.sciencedirect.com/science/article/pii/S016786550500303X doi: 10.1016/j.patrec.2005.10.010

Graves, A., & Schmidhuber, J. (2005, July). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5), 602–610. Retrieved 2023-05-30, from https://www.sciencedirect.com/science/article/pii/S0893608005001206 doi: 10.1016/j.neunet.2005.06.042

Hutter, F., Hoos, H., & Leyton-Brown, K. (2014, January). An Efficient Approach for Assessing Hyperparameter Importance. In *Proceedings of the 31st International Conference on Machine Learning* (pp. 754–762). PMLR. Retrieved 2023-08-23, from https://proceedings.mlr.press/v32/hutter14.html (ISSN: 1938-7228)

Link, D., & Hoernig, M. (2017, July). Individual ball possession in soccer. *PLOS ONE*, *12*(7), e0179953. Retrieved 2023-02-22, from https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0179953 (Publisher: Public Library of Science) doi: 10.1371/journal.pone.0179953

Ozaki, Y., Tanigaki, Y., Watanabe, S., & Onishi, M. (2020, June). Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (pp. 533–541). Cancún Mexico: ACM. Retrieved 2023-03-31, from https://dl.acm.org/doi/10.1145/3377930.3389817 doi: 10.1145/3377930.3389817

Pappalardo, L., Cintia, P., Rossi, A., Massucco, E., Ferragina, P., Pedreschi, D., & Giannotti, F. (2019, October). A public data set of spatio-temporal match events in soccer competitions. *Scientific Data*, *6*(1), 236. Retrieved 2023-07-25, from https://www.nature.com/articles/s41597-019-0247-7 (Number: 1 Publisher: Nature Publishing Group) doi: 10.1038/s41597-019-0247-7

Sorano, D., Carrara, F., Cintia, P., Falchi, F., & Pappalardo, L. (2020, July). *Automatic Pass Annotation from Soccer VideoStreams Based on Object Detection and LSTM*. arXiv. Retrieved 2023-06-27, from http://arxiv.org/abs/2007.06475 (arXiv:2007.06475 [cs])

Vidal-Codina, F., Evans, N., El Fakir, B., & Billingham, J. (2022, September). Automatic event detection in football using tracking data. *Sports Engineering*, *25*(1), 18. Retrieved 2023-03-07, from https://doi.org/10.1007/s12283-022-00381-6 doi: 10.1007/s12283-022-00381-6

Wenger, A. (2021, August). *Football Language.* Retrieved 2023-06-27, from https://www.fifatrainingcentre.comwww.fifatrainingcentre.com/en/resources-tools/football-language/index.php

Yu, J., Lei, A., & Hu, Y. (2019). Soccer Video Event Detection Based on Deep Learning. In I. Kompatsiaris, B. Huet, V. Mezaris, C. Gurrin, W.-H. Cheng, & S. Vrochidis (Eds.), *MultiMedia Modeling* (Vol. 11296, pp. 377–389). Cham: Springer International Publishing. Retrieved 2022-11-07, from http://link.springer.com/10.1007/978-3-030-05716-9_31 (Series Title: Lecture Notes in Computer Science) doi: 10.1007/978-3-030-05716-9_31