Master Thesis Summary

June 2024, Aalborg University

Alexander Nesheim Casper Kjærhus Leth Jakob Faarbæk Gregersen

Association football, or soccer, is the world's most popular sport, with significant financial stakes, such as UEFA's €2.9bn prize money in 2022. Data-driven decision-making, often referred to as "Moneyball," is increasingly applied in football for performance optimization, injury prevention, and talent identification. In a low-scoring sport like football, where a single goal can dramatically change the outcome of a match or season, optimizing set pieces, particularly corner kicks, is crucial. For example, FC Midtjylland's success in the 2014-2015 Danish Superliga was partly due to their effectiveness in scoring from corner kicks, which accounted for 39% of their total goals that season. On average, set pieces contribute 20-26% of total goals across European teams.

As data collection in football has become more advanced, machine learning has emerged as a valuable tool for analyzing this data. This study focuses on automating the classification of corner kicks using machine learning to assist coaches in making more informed decisions. By classifying corner kicks, coaches can better understand the effectiveness of different types of corner kicks and improve both offensive and defensive strategies.

This study investigates the classification of corner kicks in association football using machine learning. Utilizing a dataset of 2132 corner kick situations recorded at 25 frames per second, the study extracts eight distinct features to evaluate their effectiveness in classification tasks. Six types of corner kicks were identified and annotated based on a coaching guide. The impact of upsampling the dataset was also examined. The goal is to provide valuable insights for analysts and coaches by identifying the most effective features and classification methods.

Key findings include:

- The Player Positions feature representation achieved a macro F1-score of 0.457 with the MLP classifier.
- Some labels, like *Arc* and *Skip header*, were challenging to classify reliably.
- Classifiers struggled to distinguish between the labels *Far post* vs. *Middle of goal* and *Skip header* vs. *Front post*.
- Upsampling improved the macro F1-scores for XGB and SVM classifiers but slightly worsened MLP and RF classifiers' performance.
- Post-upsampling, some labels like *Far post* and *Skip header* showed improved performance, while others like *Arc* and *Short* performed worse.

The study concluded that the current data labelling is inadequate and suggests re-labelling by set piece coaches. Future research should consider graph representations for graph classification of corner kick set pieces, as graph representations have shown promising results for node classification tasks.

Classifying Corner Kicks in Football

June 2024

Alexander Nesheim
Aalborg University
Aalborg, Denmark
aneshe19@student.aau.dk

Casper Kjærhus Leth
Aalborg University
Aalborg, Denmark
cleth19@student.aau.dk

Jakob Faarbæk Gregersen
Aalborg University
Aalborg, Denmark
jgrege16@student.aau.dk

***Abstract*— In association football, even a marginal advantage can translate to a significant scoring advantage, often determining the outcome of a match. This study investigates the classification of set pieces using machine learning. We specifically target corner kicks, to provide valuable insights for analysts and coaches. Our research utilizes a dataset capturing player and ball positions, recorded at a frequency of 25 frames per second, over 2132 corner kick situations. From this dataset, we extract eight distinct features to evaluate their effectiveness in classification tasks, either by themselves or in combination with other features. From a coaching guide, we identified six different types of corner kicks, which we have annotated the dataset with. The study also investigates the impact of upsampling on this dataset. Our analysis finds that our features and combination of features lays a good foundation for future research into corner kick classification.**

I. Introduction

Association football, also known as soccer, is the world's most popular sport [1]. In 2022 the prize money given by The Union of European Football Associations (UEFA), was €2.9bn. For the Danish Superliga, prize money from UEFA amounted to 34% of the total revenue reported by all UEFA member clubs [2]. It is then no surprise, that an increasing factor in association football is the idea of Moneyball—using data-driven decisions for performance optimization, injury prevention, and talent identification [3], [4], [5]. All of these applications seek to enhance the competitive edge, both on and off the pitch [6].

Association football is a sport characterized by its narrow margins, largely due to its low-scoring nature. A single goal can significantly sway the outcome of a match and consequently shape an entire season [7]. An illustrative example is Aalborg Boldklub's (AaB) relegation in the 2022–2023 Danish Superliga, where they fell short by just one point. Had they managed to score a goal in either of their final two games, they would have avoided relegation [8].

A key part of association football that can be optimised for goal scoring, is set pieces—A rehearsed tactic executed immediately following a stoppage in play. Stoppages in play occur in many ways, e.g. a corner kick is awarded to the attacking team when the ball crosses the goal line, having last been touched by a defending player, and without resulting in a goal. A stoppage in play, like the corner kick, allows for the teams to reset and execute a rehearsed tactic and generate high-quality goal-scoring chances. Set pieces provide an opportunity for control in an otherwise dynamic game.

The potential impact of mastering set pieces, and in particular corner kicks, is significant. In the 2014–2015 season of the Danish Superliga, FC Midtjylland won the league. Their success has been, in part, attributed to their effectiveness on corner kicks [9]. In total, they scored 25 goals, 39% of their total goals, from corners alone. While the teams who scored the second-most corners only scored 11 goals each. On average across Europe, teams' share of set piece goals is between 20% and 26% of their total goals [10]. Additionally, sources report that while the share of goals from set pieces increases, the total number of goals scored increases as well. Thus it is not a matter of redistributing the goals scored from open play to set pieces [9].

In recent years, interest in optimising set pieces has grown, with the top teams hiring set piece-specific coaches[11]. Set piece coaches make informed decisions about which set piece to execute and how to defend, largely thanks to the increasing amount of data being gathered in association football by using cameras at the stadium [12]. Player and ball locations on the pitch in the data are often annotated with events, e.g. when corner kicks are taken, either manually or automatically [13], [14]. As large amounts of data become available, this leads to comprehensive and insightful analyses that can provide an understanding of sports dynamics and performance[15]. Machine learning, in particular, plays a significant role in this field. For instance, the machine learning models Random Forest and Logistic Regression are being utilized to predict the likelihood of imminent injuries [16] and Graph Neural Networks are being used to forecast which player will receive the ball during a corner kick [17].

Automating the data analysis work of corner kick set piece coaches through machine learning allows coaches to focus more on coaching, and less on data analysis. By classifying similar corner kicks, the coach gets an overview of which types of corner kicks are performed by a particular team and it can be further used to identify which types of

corner kicks are effective against certain teams. This information can then be used to improve both offensive and defensive strategies on corner kicks.

To classify corner kick set pieces, we use tracking data from both players and the ball captured during corner kicks in the Danish Superliga. We bound the corner kick situation, to be one second before the kick itself, and either two seconds after the kick, or one second after a teammate touches the ball, whichever occurs first.

For the classification of corner kick set pieces, we identify six different types of corner kicks used in play. We label our dataset manually according to the six types. This labelled data is the ground truth and used for training our machine learning models. The tracking data is used to extract features that feed into four different classifiers: Multi-Layer Perceptrons, Support Vector Machines, Random Forest, and XGBoost. We compare the classifiers' performance for different feature representations.

In summary, our main contribution is in the analysis and evaluation of which features perform the best when used for the classification of types of corner kicks, which leads to a concrete problem formulation of:

> Using player and ball tracking data from corner kicks, what is the feature representation that leads to the best-performing model for classifying corner kick set pieces in association football with off-the-shelf machine learning classifiers, as evaluated by F1-scoring?

The rest of the paper is structured as follows. First, we introduce prerequisites for understanding the rest of the paper. Then we provide an account of the related works in the field of machine learning and set piece analysis. Hereafter, we detail our methodology and feature representations. This is followed by the experiments section, where we evaluate and discuss the performance of the proposed classifiers. Finally, we summarize our findings and touch on relevant future work.

II. Preliminary

A. Raw corner kick data

Our dataset comprises two separate sources, in total covering 451 matches from the Danish Superliga in the seasons 2021–2022, 2022–2023 and 2023–"2024" (up to September 2023). Aalborg Boldklub provided this dataset.

The primary data source is tracking data captured at 25 frames per second by six cameras positioned around the stadium. This data yields a coordinate `(x, y, z)` location on the pitch for each player and the ball. Additionally, the data contains information identifying each player and which team is controlling the ball, but we leave this data unused. Thus, a corner kick corresponds to the locations of all 22 players on the pitch, as well as the ball for a given interval of frames.

The secondary data source is how we identify when a corner kick occurs. It contains events annotated on the relevant frame of the tracking data. Thus an annotation relates to exactly one frame of a given match. These events include actions that influence the game's progress, such as being awarded a corner kick or events that the data provider uses for delimiting events, such as *end of corner kick*.

In summary, the data constituting a corner kick is all tracking frames in the interval from the frame where a *start of corner kick*-event is recorded to the frame where a *end of corner kick*-event is recorded.

B. Processing corner kicks

It occurs that corner kick events are reported erroneously. There are frames which are annotated as a *corner kick* when upon inspection the ball is still in open play, or, a free kick has been awarded during the corner kick. These corner kicks are considered invalid and will be disregarded.

The football pitch is symmetrical along its x- and y-axis, with the centre of the pitch as the origin. Leveraging this symmetry, we convert corner kicks from all corners of the pitch to a single designated corner (bottom-left corner). This simplifies the learning process for the model, as we posit that the specific corner of execution does not change the ground truth of a set piece.

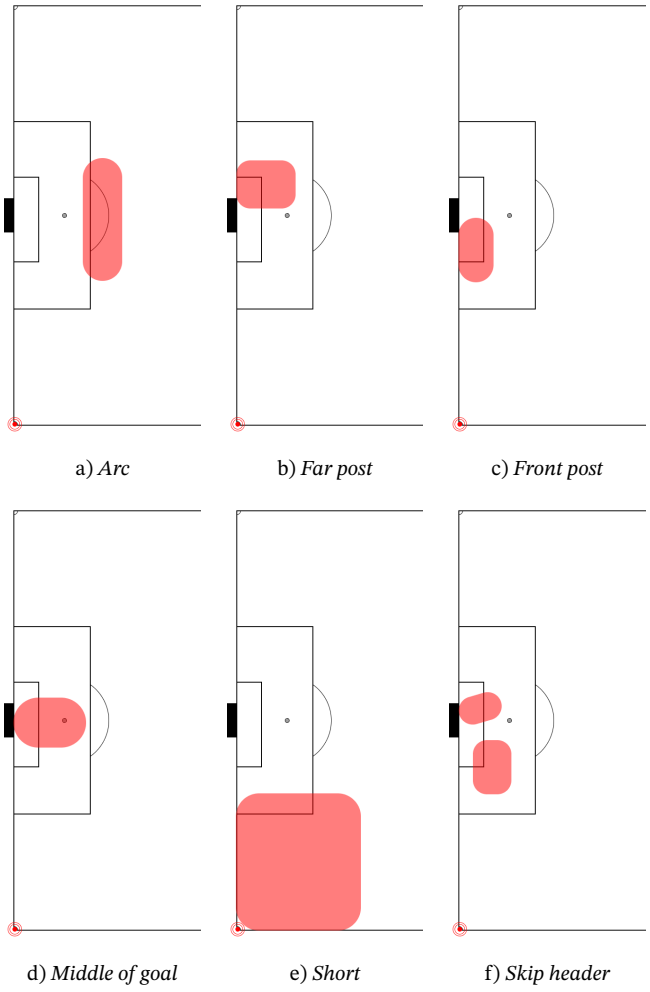
C. Labelled dataset

To enable a later classification task, we record the ground truth of a set piece, which corresponds to the type of corner kick routine. In total six distinct types of corner kicks have been identified in the problem domain through a corner kick coaching guide [18]. We label our dataset according to this coaching guide. Note that we decided not to include the *Direct Score Corner* since it is exceedingly rare. See Figure 1 for an illustration of the labels. In total, the labelled dataset consists of 2132 corner kicks.

Table 1 shows the distribution of labels in the dataset. The labelled dataset is imbalanced, as evidenced by the high variability between labels, with the least frequent label *Arc* having a share of 2.53% of the dataset, while the largest share of 32.08% is of the *Front post* label.

III. Related Works

Data-driven decision-making systems for association football focus on a specific subset of the game, ie. clustering of corner kicks, automated tagging of events and predicting the likelihood of scoring from a specific situation.



a) *Arc* b) *Far post* c) *Front post*

d) *Middle of goal* e) *Short* f) *Skip header*

Figure 1: Zones relevant for labelling a set piece
One or more players must make runs into or within the areas marked on the images for it to be labelled accordingly. If more than one area has some runs into them, the ball's position is used as a tiebreaker.

Early work in the field of association football corner kick analysis focused on notational analysis for characterising corner kicks. Including, the way the ball curves on the kick, the number of attacking players involved as well as the type of defensive strategy deployed by the defensive team has been studied [19]–[23].

Using notational analysis Power P. et al. [20] highlight that goals from corners are more often scored from the second ball (having been touched by a teammate) than from the ball delivery itself. Our work is situated in this vein, of attempting to gain an advantage by exploiting tracking data and implementing a data-driven system for the classification of set pieces.

Recently, Shaw L., Gopaladesikan S. [24] identified a key factor for classifying corner kick routines as being a set of rehearsed, simultaneous, coordinated runs that players execute to improve the scoring opportunity. These runs are called Co-occurring Player Runs.

Table 1: The distribution of labels in our dataset of 2132 corner kicks.

Label	Count	Share
Arc	54	2.53%
Far post	326	15.29%
Front post	684	32.08%
Middle of goal	529	24.81%
Short	242	11.35%
Skip header	297	13.93%

The authors deployed Gaussian Mixture Models (GMMs) to discover zones where players start and end their runs, which can characterize the attacking teams' behaviour during a set piece. These zones are used to represent Co-occurring Player Runs. Their method represents each set piece using the positions of the attacking players' locations at the start of the set piece and the end of the set piece.

Shaw L., Gopaladesikan S. down-sample each set piece to two frames. The first frame occurs two seconds before the corner kick is executed, and the second frame occurs either one second after the first teammate touches the ball or two seconds after the corner is taken, whichever occurs first. These are the frames from the set piece that is used in their feature representation. Their dataset consists of tracking- and event data comparable to our dataset. Our work differs on two points. Firstly, our method is designed for classification as opposed to clustering, and secondly, we explore other feature representations.

The latest developments in the field come from Wang Z. et al., which focuses on receiver prediction, shot attempt prediction, and tactical adjustment recommendations for corner kicks. They conceptualize a corner kick as a fully connected graph, where each node is a player, and edges indicate whether two nodes are teammates or opponents [17]. Each node is annotated with features like x- and y-coordinates, velocity, and the player's height and weight. Their model is a deep graph attention neural network employing geometric deep learning techniques. They augment each corner kick by generating symmetric transformations by reflecting each corner kick across the x-, y- and both axes, thus each corner kick is replicated four times in the data. In contrast, we reflect every corner to the bottom left corner (see Section II.B).

They show that the model can generate tactical adjustment recommendations which coaches are unable to distinguish from human recommendations, and in most instances, the coaches prefer the model recommendations to human-made alternatives. Our work differs on two points, in purpose and method; 1) we classify corner kick routines as a whole, while they focus on identifying particular situa-

tions within a corner kick (i.e. shot and receiver prediction), and, 2) we compare multiple different classifiers as well as utilise different feature representations.

Overall, our research aligns with Shaw L., Gopaladesikan S. as well as Wang Z. et al., as we employ a comparable dataset. Similarly, we apply machine learning classifiers to derive insights into corner kick routines. Our contribution diverges from these prior studies as we concentrate on classifying corner kick set pieces. To our knowledge, we are the first to attempt corner kick set piece classification. Specifically, we construct corner kick embeddings using diverse feature representations.

IV. Method

In this section, we explain our approach to classifying corner kick set pieces, according to the labels presented in Section II.C. In particular, we focus on feature representations for corner kick set pieces based on spatio-temporal data.

A. Corner kick labelling

As our dataset was not labelled initially, we had to obtain the labels by other means. For this, we implemented a simple web application, which allows a user to inspect a 2D animation of a randomly selected set piece. The animations showed the locations and movement of the offence, the defence and the ball. The user could label a set piece with one of the labels in Figure 1 or mark it *Invalid* i.e. the set piece is malformed or erroneous. An erroneous corner kick is when the ball is not recorded correctly or there is no stoppage in play. We also exclude all corner kicks that occur when there are less than 11 attacking players (e.g. when one or more players from the attacking team have been brandished a red card).

Furthermore, when in doubt, the user can skip a particular set piece. Each set piece is between 3 and 4 seconds, hence considerable effort has been spent on the labelling process. The labelling process has been carried out internally on a best-effort basis. We are aware of potential discrepancies between users due to the subjectivity of the labelling process. Additionally, labelling is challenging as some pairings of labels can be difficult to discriminate between. For instance, in Figure 1, the *Skip header* label (f) overlaps the *Front post* label (c) and *Middle of goal* label (d).

This is a clear limitation of our work. As we did not have access to professional coaches during the labelling process, we have not been able to rectify our labels. Nonetheless, we posit that our labelling effort is sufficiently accurate to study the effectiveness of feature representations.

B. Representing set pieces

Recall that the goal is to classify the type of a given set piece. Besides utilising the labels mentioned earlier, another important aspect is how the set piece under consideration is represented to the classifiers. In particular, each set piece is represented by a feature vector. A feature vector can encode a set piece to a numerical representation. We explore multiple feature representations and quantify their performance in the classification task via the F1-score. See Section IV.D for an explanation of F1-scores.

Both Wang Z. et al. and Shaw L., Gopaladesikan S. perform down-sampling on corner kicks, and calculate their features from one and two frames respectively. We argue that the timing of events is crucial for accurately classifying corner kicks, given that analysts rely on video footage to recognize corner kick patterns in their daily analysis. Furthermore, both Wang Z. et al. and Shaw L., Gopaladesikan S. demonstrate some level of success in extracting features from specific frames, indicating the presence of valuable information at specific moments during set pieces.

Therefore, the objective is to select a set of frames that are representative of a set piece. Shaw L., Gopaladesikan S. identify the start and the end of a corner kick, as mentioned in Section III.

Wang Z. et al. use a singular frame, i.e. the frame where the corner kick is taken. This leads us to choose three frames, merging the frames used by Wang Z. et al. and Shaw L., Gopaladesikan S. We use these frames for feature calculation, making all of our features employ down-sampling.

In the following, the engineered features are presented. Note that we also implement the CPR feature representation from Shaw L., Gopaladesikan S. for comparison. We split our feature representation into three categories: 1) Position-based, 2) Distance-based, and 3) Zone-based features. The first features are the most basic and naive features. They employ a minimum of feature engineering and use the player and ball positions directly in the representations. Second are the distance-based features, which encode the relations of attacking players and the ball using Euclidean distance. Finally, the zone-based features seek to encode spatial locality by considering the players' positions to emergent zones.

The features are presented in isolation, however, we hypothesize that some features encode specific aspects of a corner kick better than others. Therefore, we also explore combinations of features during the experiments.

1) *Position-based feature representations :*

Player Positions denote the feature representation, which consists of the player locations at a single given frame for each set piece. Thus the feature vector has 22 entries, corre-

sponding to the x- and y-coordinates of all 11 players of the attacking team.

$$\text{Player Positions} = [x_1, x_2, ..., x_{11}, y_1, y_2, ..., y_{11}] \quad (1)$$

Similarly, *Ball Position* denotes the feature representation, which consists of the ball location at a single given frame per set piece. Thus the feature vector has two entries, corresponding to the x- and y-coordinate of the ball.

$$\text{Ball Position} = [x, y] \quad (2)$$

The features mentioned above, are computed using a single frame of the set piece. This down-sampling means that the usefulness of the information encoded in the feature is highly dependent on how descriptive the selected frame is for the set piece. In the *Heatmap* feature representation, we ameliorate this concern of selecting a single frame, by using all frames of a set piece to compute a heatmap of their movements in the final third of the pitch.

A heatmap is usually depicted as a grid, where a colour gradient is used to indicate higher values, or heat, in a cell. Our heatmap comprises of cells, where the heat level reflects how many player positions have been recorded in that area, during the set piece.

$$\text{Heatmap} : \begin{bmatrix} H_{0,0} & H_{0,1} & \cdots & H_{0,n} \\ H_{1,0} & H_{1,1} & \cdots & H_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ H_{m,0} & H_{m,1} & \cdots & H_{m,n} \end{bmatrix} \quad (3)$$

Here, $H_{\text{m,n}}$ represents the heat level in the cell (m, n). When k players spend t frames in a cell, it accumulates a heat value of $k * t$ for that cell.

The heatmap is an image, which we pass to a convolutional autoencoder (CAE), for Figure 2, it is evident that most of the bins have a value of zero, which makes every representation of a heatmap more alike than they are different.

An autoencoder learns the features that are most significant for the task of reconstruction, as described in Appendix D. The latent layer of the autoencoder would therefore be the latent variables that best describe the heatmap, which we believe is a better foundation for classification.

The CAE is useful for obtaining a more compressed representation of the input (here the heatmap of a set piece). The CAE's learning objective is to reconstruct the input in the presence of an information bottleneck resulting in an approximated, reconstructed output [25]. The latent representation is the encoding of the input, which is used as the feature vector for the downstream classification task. See Appendix D for details on CAE in general, see Appendix E for the specific configuration of the CAE used.

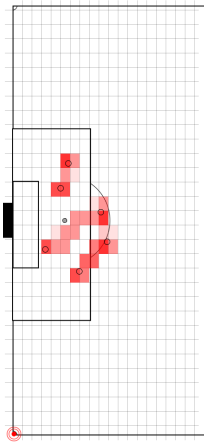


Figure 2: Heatmap
The heatmap consists of 42×30 bins, with a heat value between 0 and 1. The heat is higher where the red colour is less transparent.

See Appendix F for examples of the reconstructed output from our CAE.

Figure 2 showcases how the movement of three players would be recorded in the heatmap feature representation. The heatmap comprises 42×30 bins distributed evenly across the final third of the pitch. The squares where the red colour is less transparent are areas that players have occupied the most.

Note that the heatmap also results in downsampling, however, it is not dependent on how descriptive the selected frame of a set piece is, as opposed to Player Positions, Ball Positions and the features proposed by Wang Z. et al. and Shaw L., Gopaladesikan S..

2) *Distance-based feature representations :*

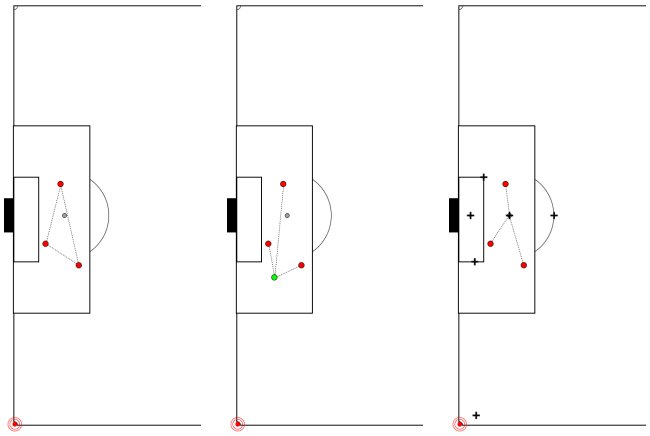
Similarly to Wang Z. et al., we posit that the dynamic relations between players are important for a set piece. To model these relations we employ three Euclidean distance-based features. The intuition is to model the relations by the distance between players, the ball or to specific points of interest in the final third.

Figure 3 shows examples of the features that are calculated using Euclidean distance.

Pairwise Players is intended to capture information on the players' locations relative to each other without relying on coordinate positions. This corresponds to the upper triangle of a distance matrix, based on all members of the offensive team. Pairwise Players calculation is shown in equation (4), where d is the Euclidian distance function between two players i and j.

$$\text{Pairwise Players} = \left[d_{i,j} \mid 1 \leq i < j \leq 11\right] \quad (4)$$

We order the elements in the feature vector from the lowest x-coordinate to the highest.



a) Pairwise Players b) Ball to players c) Informed Points

Figure 3: Distance-based features
a) consists of the pairwise distance between all players. b) consists of the distance from the ball to each of the players. c) consists of six points placed in the final third. Red dots indicate players, the green dot is the ball and the dotted lines are distances.

Intuitively, the goalkeeper of the attacking team is then the last element of the feature vector, regardless of the corner kick. The underlying assumption for this ordering is that the closer to the opposing team's goal a player is, the more important they are. This feature does not take into account where on the pitch the players are located, as it only considers the relations between the players themselves.

Ball to players is intended to capture the relation between players' positions and the ball's position. It is calculated as each player's distance to the ball. The Ball to players calculation is shown in equation (5), where d is the Euclidian distance between the ball position ball and player i.

$$\text{Ball to players} = \left[d_{\mathrm{ball},i} \mid 1 \leq i \leq 11 \right] \tag{5}$$

Informed Points is intended to capture where in the final third players are located in relation to points placed according to the areas significant in the coaching guide, which we used for labelling (see Figure 1).

For each point, we sum up all players' distances to that point. This produces six values, which make up the feature.

3) *Zone-based feature representations:* The classes depicted in Figure 1 indicate zones of importance per label. We posit that using the raw positional data to derive zones allows an efficient spatial encoding while providing a closer match to how humans identify set piece types.

While the distance-based features encode spatiality via the distance between points, the zone-based feature representations encode spatiality via frequency counts of players per zone. Thus the goal is to identify how many and which zones to count the frequencies for.

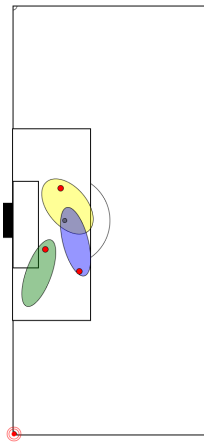


Figure 4: Players in Gaussians
This feature computes the probability that each player originates from a specific Gaussian. It does this for each Gaussian. For this example, the feature vector has three elements, corresponding to one for each Gaussian process in the model.

We use GMMs to identify zones. This achieves two things; 1) Allows us to discover emergent zones from the data, and, 2) instead of relying on simple frequencies, GMMs provide a probabilistic representation of each zone. This means we can determine the likelihood that a player belongs to each zone. This probabilistic approach accurately captures situations where a player might be near the edges of multiple zones, reflecting that players can simultaneously belong to or be influenced by more than one zone.

Figure 4 showcases an example with three players and three Gaussians. Each player has some probability of originating from each of the Gaussians. The probabilities are summed up to produce the feature vector, which consists of a sum of probabilities per Gaussian.

Additionally, we reimplement part of the feature representation proposed by Shaw L., Gopaladesikan S. for comparison. As briefly mentioned in Section III, they model a set piece using the location that players run from and the location they run to. To be able to represent the many possible runs, they want to find larger areas of the final third of the pitch, where players frequently start the run and other areas where players end their runs. To obtain these areas, or zones, they employ a GMM fitted on every player's location at the start of the set pieces and another GMM fitted on every players's end locations. They make use of the GMM's weights to calculate the probability that a player has made each of the available runs - from an initial zone to a target zone. For details on GMMs in gerneral see Appendix C.

The feature vector representing Co-occurring Player Runs denoted as CPR, is defined as:

$$\mathrm{CPR} = [x_0, x_1, ..., x_{\text{a-1}}, x_a] \tag{6}$$

Where a is the number of pairs in the cartesian product of the set of initial zones and the set of target zones. The value of any x is given by:

$$x_a = \sum_{p=1}^{N_p} P_p\Big(\text{InitialZone}_{R_a}\Big) P_p\Big(\text{TargetZone}_{R_a}\Big) \quad (7)$$

Upon reviewing the above Equation (7), which is from the appendix of L. Shaw and S. Gopaladesikan. We are not convinced that the probabilities are independent as the equation claims. It seems only logical that the probability of where you end up is heavily dependent on where you came from.

In Equation (7), N_p is the number of players, $P_p\Big(\text{InitialZone}_{R_a}\Big)$ represents the probability that player p started in the initial zone of run R_a, and $P_p\Big(\text{TargetZone}_{R_a}\Big)$ represents the probability that player p ended in the target zone of run R_a. The initial and target zones are defined from the centres and covariances of the Gaussians in the GMM. We determine the likelihood of a player's position, given the settings of each Gaussian distribution.

L. Shaw and S. Gopaladesikan further use Non-Negative Matrix Factorization to compose a set of runs, that used in combination, best explain the distinct types of corner kick set pieces. With this basic set of runs, they can easily find corner kick set pieces that are similar to each other.
Table 2 shows all the proposed feature representations for identifying set pieces.

C. Classifiers

We explore multiple classifiers to quantify the performance of the feature representations. We use Multi Layer Perceptron (MLP), Support Vector Machine (SVM), Random Forest (RF) and Extreme Gradient Boosting (XGB) classifiers. The RF and SVM classifiers are implemented in the Python library `scikit-learn` [26], while the MLP classifer is implemented using the `keras` library [27] and the XGB classifier is implemented using the XGBoost Python Library [28].

D. Evaluation

We evaluate the feature representations using the downstream classification task. For each of the representations, we record the performance using the *macro* and *micro* F1-score.

F1-score is a commonly used evaluation metric within machine learning. It considers two important metrics used in machine learning; Recall and Precision. Recall is the ratio of true positive predictions to the total actual positives. Precision is the ratio of true positive predictions to the total predicted positives.

These might seem very similar, but in fact, measure different things.

Table 2: Features for representing a set piece. In Appendix A, visual aids for each feature are provided.

Feature	Type	Explanation
Pairwise Players	Frame	Pairwise distances between attacking players
Ball to players	Frame	Distance from the ball to each player
Informed Points	Frame	Six points are placed in the final third at interesting locations. For each point we sum all the players' distances to it
Players in Gaussians	Frame	As explained in Section IV.B.3
Ball Positions	Frame	(x, y)-position of the ball
Player Positions	Frame	(x, y)-positions of the players on the offensive team
Co-occurring Player Runs [24]	Global	As explained in Section IV.B.3
Heatmap	Global	As explained in Section IV.B.3

Recall measures how many of the actual positive instances are correctly predicted while precision measures how many of the predicted positive instances are actually positive. Both are described in equation (8).

$$\left\{ \begin{array}{c} \text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}} \\ \text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}} \end{array} \right\} \quad (8)$$

The F1-score is calculated as the harmonic mean of the recall and precision as described in equation (9). *Micro* F1-score is based on the binary classification of a single label, while *macro* refers to the average of all *micro* F1-scores.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (9)$$

V. Experiments

We quantify our findings through experiments. Below we describe the experimental setting and present the tuned classifiers. Additionally, we present and discuss the experimental results.

A. Experimental Setting

The classifiers are trained individually, using an NVIDIA GeForce RTX 3060 GPU. The data is shuffled and split into an 80:20 ratio for training and evaluation. We explicitly seed the randomness of all experiments to ensure reproducibility. We employ a 5-fold cross-validation during hyperparameter tuning. We use Bayesian Optimization (BayesSearchCV) for hyperparameter tuning. All the classifiers have an equal tuning budget. Each model is tuned per experiment using the search spaces found in Appendix B.

Table 3: Macro F1-scores for all feature representations per classifier. **Bold** indicates highest mean value for the classifier.

Feature	SVM	RF	MLP	XGB
Pairwise Players	0.2343±0.02	0.2886±0.03	0.3338±0.04	0.2283±0.03
Ball to players	0.2502±0.04	0.3824±0.01	0.3753±0.1	0.2123±0.01
Informed Points	**0.4051±0.02**	0.3512±0.03	0.4286±0.09	0.3435±0.07
Players in Gaussians	0.3972±0.03	0.3971±0.01	0.4325±0.04	0.2952±0.04
Ball Positions	0.24±0.02	**0.4145±0.03**	0.4442±0.05	0.3879±0.03
Player Positions	0.3159±0.02	0.4028±0.06	**0.457±0.08**	**0.4166±0.03**
Co-occurring Player Runs	0.3471±0.05	0.3431±0.03	0.3729±0.02	0.3671±0.07
Heatmap	0.0799±0.01	0.0982±0.02	0.0799±0.01	0.1037±0.03

We report our results from five replications (each using a different seeded randomness), as $a \pm b$, where a is the mean observation of the evaluation scores, and b is the error bounds - calculated as the standard deviation.

B. Experimental Results

The individual results of the feature representations are presented in Table 3 via the macro F1-scores.

It is noteworthy that Player Positions performs the best in both the MLP and the XGB classifier. Additionally, the Informed Points and Ball Positions feature representations perform the best for the SVM and RF classifiers respectively. The Heatmap feature performs significantly worse than any other feature. No configuration achieves what we believe would be satisfying performance (F1-score > 0.7).

To investigate the cause for the lackluster macro F1-scores we inspect the micro F1-scores of the classifiers. In Table 4 the micro F1-scores of the highest scoring feature representation per classifier from Table 3 are shown.

It is evident that the classifiers struggle with the *Arc* and *Skip header* labels in particular. Meanwhile, *Short* and *Front post* achieve significantly better scores.

The configurations of classifier and feature representations that achieve the highest micro mean F1-scores, per label, are shown in Table 5.

Table 5 illustrates that no specific configuration can adequately capture all of the ground truth labels. However, it is still interesting Player Positions, Informed Points and Players in Gaussians - respectively a Position-, a Distance- and a Zone-based feature representation, can all compete in performance.

Table 4: Micro F1-scores from the best performing configurations of Table 3. **Bold** indicates highest score per label.

Label	SVM	RF	MLP	XGB
Arc	0±0	0±0	**0.2607±0.36**	0±0
Far post	0.5175±0.09	**0.5078±0.16**	0.478±0.19	0.4013±0.15
Front post	0.6092±0.05	0.6068±0.07	0.5676±0.11	**0.6106±0.06**
Middle of goal	**0.4854±0.06**	0.4709±0.11	0.4245±0.04	0.4756±0.06
Short	0.8186±0.04	0.8954±0.07	0.8748±0.1	**0.9236±0.05**
Skip header	0±0	0.0063±0.03	**0.1363±0.14**	0.0886±0.06

Table 5: Micro F1-scores of the combination of classifier and feature.

Label	Configuration	F1-score
Arc	MLP w. Player Positions	0.2607±0.36
Far post	SVM w. Informed Points	0.5175±0.09
Front post	XGB w. Player Positions	0.6106±0.06
Middle of goal	SVM w. Informed Points	0.4854±0.06
Short	XGB w. Player Positions	0.9236±0.05
Skip header	MLP w. Players in Gaussians	0.2483±0.19

C. Confusion matrices

Figure 5 shows the confusion matrix for predictions made by the best performing configuration (MLP + Player Positions, from Table 3). We note that *Short* is identified reliably. The confusion matrix indicates that when the classifier predicts *Far post* we are likely to be mistakenly identifying *Middle of goal*, while predictions for *Middle of goal* are confused with both *Far post* and *Front post*. Additionally, predictions for *Front post* are often confused with *Skip header*. Finally, *Skip header* is rarely predicted, and when it is, most of the time it is mistaken with *Front post*.

Overall, the classifiers are unable to distinguish *Far post*, *Front post*, and *Middle of goal* reliably. In particular, *Middle of goal* seems to confuse the classifier. As a final note, there is an argument to be made that the label *Skip header* is a variation of the label *Front post*, where the ball simply was delivered too short and therefore a player had to head it into the goal area.

We propose two avenues for ameliorating these shortcomings. 1) Combining the features that achieve the highest micro F1-scores could aid the classification task, and, 2) balancing the dataset by upsampling, such that each class is evenly represented during training.

More research into this is needed.

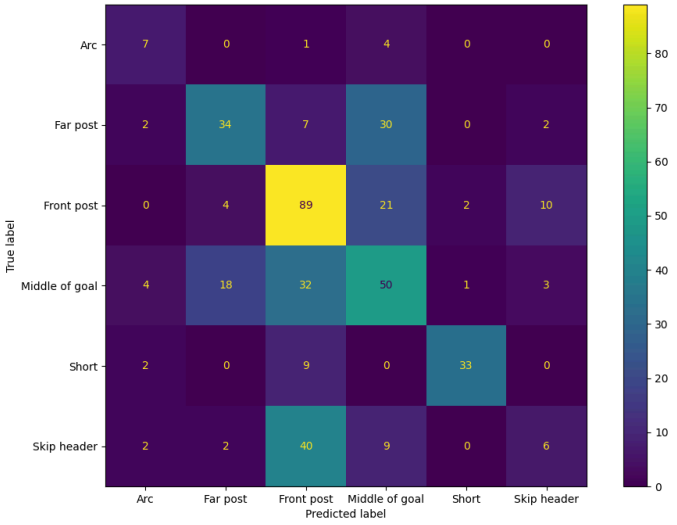


Figure 5: Confusion matrix of classifications made by the MLP using the Player Positions feature.

D. *Combinations of features*

The features we combine are chosen based on Table 5. The combination of features is done by concatenating the individual feature vectors per sample.

No classifier achieves better macro F1-scores using the combined features. Therefore we omit the micro F1-scores. We hypothesize that rather than aid the classifiers, the features making up the combination seem to somewhat cancel each other out. Intuitively, while a single feature achieves a high micro F1-score for *Arc*, it might achieve poor performance on all other labels, thus limiting the classifier, rather than enhancing it.

E. *Balanced dataset upsampling*

As shown in Table 1 the dataset is highly imbalanced. In this section, we examine the impact of balancing the dataset compared to the original, unbalanced dataset.

We balance the dataset through the SMOTE algorithm. SMOTE works by selecting a random data point in the dataset and generating new data that is situated along the line that goes through the chosen point and one of its nearest neighbours. This process is carried out iteratively until the classes are balanced [29].

The macro F1-scores of applying SMOTE on the dataset is shown in Table 7 alongside the original F1-score for the original dataset. We only consider the best performing classifier configurations from Table 3. Upsampling of our dataset generally improves performance across all classifiers except the MLP.

We take the micro F1-scores into account to further quantify our findings.

Table 6: Macro F1-scores for the combination of features that achieve the highest micro F1-scores (Player Positions, Informed Points and Players in Gaussians)

Feature	SVM	RF	MLP	XGB
Best feature per label	0.3471±0.05	0.3431±0.03	0.3729±0.02	0.3671±0.07

Table 7: Results of oversampling through SMOTE algorithm compared to the original results as reported in Table 3. Using the Player Positions feature representation

Dataset	SVM	RF	MLP	XGB
Original	0.3159±0.02	0.4028±0.06	0.457±0.08	0.4166±0.03
Upsample	0.3948±0.05	0.4529±0.02	0.4433±0.05	0.4667±0.03

Table 8: Per label F1-score results of upsampling through SMOTE algorithm compared to the original results as reported in Table 4

Ground Truth Class	Upsampled	Original
Arc	0.2121±0.11	0.2607±0.36
Far post	0.5092±0.16	0.478±0.19
Front post	0.4046±0.14	0.5676±0.11
Middle of goal	0.3357±0.13	0.4245±0.04
Short	0.8653±0.08	0.8748±0.1
Skip header	0.333±0.11	0.1363±0.14

The micro F1-scores of the best performing configuration, based on the original dataset (MLP + Player Positions) is shown in Table 8 comparing the original performance with the upsampled performance.

In Table 8, we see that *Arc*, *Front post*, *Middle of goal* and *Short* performance is lower, while *Far post* and *Skip header* are higher. Additionally, note that the standard deviation of *Arc* is significantly lower for the upsampled configuration, meanwhile, the reverse is true for *Middle of goal*.

In Figure 6 the confusion matrix for the MLP + Player Positions configuration is shown.

It is evident, that predicting *Arc* has become more prevalent compared to Figure 5, however, this is now confused with *Middle of goal*, which explains the drop in micro F1-score seen in Table 8. Previously, when predicting *Front post* it was often mistaken with *Skip header*, now the inverse is true; predicting *Skip header* is now often mistaken with *Front post*. On the other hand, *Far post* is now more reliably predicted, and only to a lesser extent confused with *Middle of goal*, and rarely with *Front post*.

In general, the results of upsampling the dataset is inconclusive. The upsampling boosts the XGB and SVM classifiers while impairing the MLP and RF classifiers.

Regardless of configuration, no classifier achieves satisfying results (F1-score >0.7).

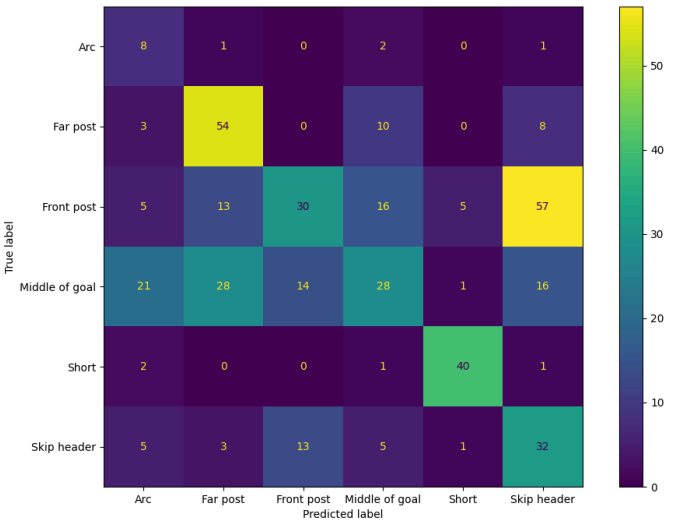


Figure 6: Confusion matrix of classifications made by the MLP using the Player Positions feature after upsampling.

VI. Recommendations and Conclusions

Based on tracking- and event data we engineered eight corner kick feature representations. In particular, we employ GMMs and CAEs in the feature engineering process. Additionally, we quantify the performance of the feature representations in the downstream corner kick classification task. We train four classifiers; Random Forest, eXtreme Gradient Boost, Support Vector Machine and Multi-Layer Perceptron for the classification task.

We study the impact on performance of balancing the dataset via SMOTE in relation to the four classifiers. Additionally, we explore the effects of combining feature representations based on their micro F1-score performance.

The Player Positions feature representation achieves a macro F1-score of 0.457 ± 0.08 for MLP classifier.

Additionally, we quantify the per-label performance by the micro F1-scores. The experiments show that the *Arc* and *Skip header* labels cannot reliably be classified with our feature representations(micro F1-score: 0.2607 ± 0.36 and 0.1363 ± 0.14, respectively). Generally, the classifiers are unable to clearly distinguish *Far post* from *Middle of goal*, *Middle of goal* from *Front post*, and *Skip header* from *Front post* as shown in Figure 5.

Through upsampling, we show that the XGB and SVM classifiers achieve significantly better macro F1-score compared to the original dataset, while the MLP and RF classifiers perform slightly worse.

Furthermore, the labels *Far post* and *Skip header* also show better performance after upsampling. Conversely, *Arc*, and *Short* perform slightly worse, while *Front post* and *Middle of goal* perform considerably worse after upsampling.

It is evident that our data labelling is lacking and if possible should be redone by set piece coaches, in the future. Whether the labels themselves are inherently ambiguous or the labelling process itself is the source of error is unclear.

Furthermore, due to time constraints, we have not attempted to replicate or build upon the graph representations proposed in Wang Z. et al.. However, future work studying graph representations of corner kick set pieces for graph classification seems promising.

References

[1] Wikipedia contributors, "Association football — Wikipedia, The Free Encyclopedia." [Online]. Available: https://en.wikipedia.org/w/index.php?title=Association_football&oldid=1219315307

[2] UEFA, "The European Club Finance and Investment Landscape." [Online]. Available: https://ecfil.uefa.com/2023

[3] H. Van Eetvelde, L. D. Mendonça, C. Ley, R. Seil, and T. Tischer, "Machine learning methods in sport injury prediction and prevention: a systematic review," *Journal of experimental orthopaedics*, vol. 8, pp. 1–15, 2021.

[4] A. Majumdar, R. Bakirov, D. Hodges, S. Scott, and T. Rees, "Machine learning for understanding and predicting injuries in football," *Sports Medicine-Open*, vol. 8, no. 1, p. 73–74, 2022.

[5] J. G. Claudino, D. d. O. Capanema, T. V. de Souza, J. C. Serrão, A. C. Machado Pereira, and G. P. Nassis, "Current approaches to the use of artificial intelligence for injury risk assessment and performance prediction in team sports: a systematic review," *Sports medicine-open*, vol. 5, pp. 1–12, 2019.

[6] N. Chmait and H. Westerbeek, "Artificial intelligence and machine learning in sport research: An introduction for non-data scientists," *Frontiers in sports and active living*, vol. 3, p. 363–364, 2021.

[7] C. Anderson and D. Sally, *The Numbers Game: Why Everything You Know About Soccer Is Wrong*. Penguin Publishing Group, 2013.

[8] Wikipedia contributors, "2022–23 Danish Superliga — Wikipedia, The Free Encyclopedia." [Online]. Available: https://en.wikipedia.org/w/index.php?title=2022%E2%80%9323_Danish_Superliga&oldid=1214647845

[9] T. Knutson, "I Think We Broke Denmark." [Online]. Available: https://statsbomb.com/articles/soccer/i-think-we-broke-denmark/

[10] J. Campbell, "Set-piece kings Midtjylland – and their former Celtic star – have left top teams admiring from afar." [Online]. Available: https://theathletic.com/1103112/2019/08/07/how-set-piece-kings-midtjylland-and-their-former-celtic-star-have-left-even-the-likes-of-manchester-city-behind/

[11] [Online]. Available: https://www.theguardian.com/football/article/2024/may/03/identify-weaknesses-and-exploit-them-the-rise-of-the-set-piece-coach?CMP=share_btn_url

[12] K. Carney, "Set pieces have gone from ugly duckling to Gamechanger as clubs focus on details | Karen Carney." [Online]. Available: https://www.theguardian.com/football/2024/feb/29/set-pieces-have-gone-from-ugly-duckling-to-game-changer-as-clubs-focus-on-details

[13] R. Theagarajan, F. Pala, X. Zhang, and B. Bhanu, "Soccer: Who has the ball? Generating visual analytics and player statistics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1749–1757.

[14] S. A. Pettersen *et al.*, "Soccer video and player position dataset," in *Proceedings of the 5th ACM multimedia systems conference*, 2014, pp. 18–23.

[15] A. Rossi, L. Pappalardo, and P. Cintia, "A narrative review for a machine learning application in sports: an example based on injury forecasting in soccer," *Sports*, vol. 10, no. 1, p. 5–6, 2021.

[16] A. Rossi, L. Pappalardo, P. Cintia, F. M. Iaia, J. Fernández, and D. Medina, "Effective injury forecasting in soccer with GPS training data and machine learning," *PloS one*, vol. 13, no. 7, p. e201264, 2018.

[17] Z. Wang *et al.*, "TacticAI: an AI assistant for football tactics," *CoRR*, 2023, doi: 10.48550/ARXIV.2310.10553.

[18] CoachingAmericanSoccer.com, "Types of corner kicks | Coaching American Soccer." [Online]. Available: https://coachingamericansoccer.com/tactics-and-teamwork/types-of-corner-kicks/

[19] B. Hannah and S. J. Antony, "Analysis of attacking corner kick strategies in the FA women's super league 2017/2018," *International Journal of Performance Analysis in Sport*, vol. 19, no. 6, pp. 893–903, 2019.

[20] P. Power, J. Hobbs, H. Ruiz, X. Wei, and P. Lucey, "Mythbusting set-pieces in soccer," in *Proceedings of the 12th annual MIT Sloan Sports Analytics Conference*, 2018.

[21] C. A. Casal, R. Maneiro, T. Ardá, J. L. Losada, and A. Rial, "Analysis of corner kick success in elite football," *International Journal of Performance Analysis in Sport*, vol. 15, no. 2, pp. 430–451, 2015.

[22] C. Pulling, "Long corner kicks in the English Premier League: Deliveries into the goal area and critical area," *Kinesiology*, vol. 47, no. 2., pp. 193–201, 2015.

[23] C. Pulling and J. Newton, "Defending corner kicks in the English Premier League: near-post guard systems," *International Journal of Performance Analysis in Sport*, vol. 17, no. 3, pp. 283–292, 2017.

[24] L. Shaw and S. Gopaladesikan, "Routine inspection: A playbook for corner kicks," in *Machine Learning and Data Mining for Sports Analytics: 7th International Workshop, MLSA 2020, Co-located with ECML/PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings 7*, 2020, pp. 3–16.

[25] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*, 2011, pp. 52–59.

[26] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[27] F. Chollet and others, "Keras." [Online]. Available: https://keras.io/

[28] T. Chen and others, "XGBoost." [Online]. Available: https://xgboost.readthedocs.io/en/stable/index.html

[29] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[30] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *2007 IEEE conference on computer vision and pattern recognition*, 2007, pp. 1–8.

[31] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[32] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*, 2011, pp. 52–59.

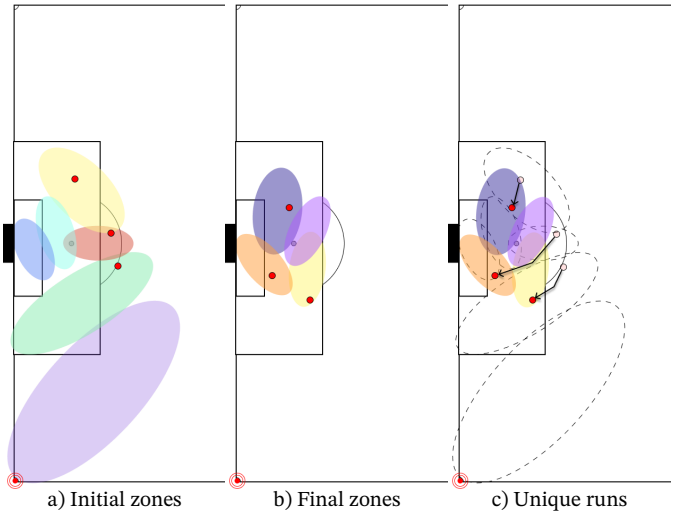
[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

Appendix

A. Feature illustrations

This section contains an illustration of the feature Co-occurring Player Runs on a football pitch. The illustration is a top-down view of the final third, in which the set piece is executed. The ball is kicked from the bottom-left corner, which is marked with a red circle for clarity.

Figure 7 showcases the CPR feature re-implemented from L. Shaw and S. Gopaladesikan [24].



a) Initial zones b) Final zones c) Unique runs

Figure 7: Co-occurring Player Runs
All the runs that players execute simultaneously, that originate from one initial zone to a final zone. Red dots indicate players and coloured ellipses indicate Gaussian processes. In c) the dotted ellipses are initial zones and the coloured ellipses are final zones. The arrows indicate the run that a player has made.

B. Hyperparameter search spaces

Below, the search spaces for the hyper-parameter tuning are shown per Table 9, Table 10, Table 11, and Table 12 respectively RF, SVM, MLP, and XGB.

Table 9: RF Hyper-parameter search space

Parameter	Values
n_estimators	10, 110, ..., ..., 910, 1010
criterion	'gini', 'entropy', 'log_loss'
max_depth	10, 20, ..., 90, 100, None
min_samples_split	2, 5, 10
min_samples_leaf	1, 2, 4
max_features	'log2', 'sqrt'
bootstrap	True, False
class_weight	'balanced', 'balanced_subsample', None

Table 10: SVM Hyper-parameter search space

Parameter	Values
C	0.05, 0.10, ..., 0.95, 1
kernel	'linear', 'poly', 'rbf', 'sigmoid'
gamma	'scale', 'auto'
class_weight	'balanced', None

Table 11: MLP Hyper-parameter search space

Parameter	Values
learning_rate	0.0001, 0.0002, ..., 0.049,0.50
batch_size	64, 128, 256
optimizer	SGD, Adam, RMSprop, Nadam
n_layers	1, ..., 5
n_units	4, 8, ..., 20, 24
kernel_init	'glorot_normal', 'he_normal', 'he_uniform', 'glorot_uniform'
activation	'relu', 'selu'
dropout	0.0,0.1, ..., 0.6,0.7
normalizer	True, False

Table 12: XGBoost Hyper-parameter search space

Parameter	Values
n_estimators	10, 110, ..., 1010
max_depth	3, 4, ..., 9, 10
grow_policy	'lossguide', 'depthwise'
learning_rate	0.1, 0.2, 0.3,0.4 0.5
tree_method	'approx'
gamma	0.0, 0.1, ..., 0.9, 1.0
min_child_weight	0.0, 0.1, ..., 0.9, 1.0
subsample	0.0, 0.2, ..., 0.8, 1.0
colsample_bytree	0.0, 0.2, ..., 0.8, 1.0
colsample_bylevel	0.0,0.2, ..., 0.8, 1.0
colsample_bynode	0.0,0.2, ..., 0.8, 1.0
objective	'multi:softmax'
num_class	6

C. Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a model that describes some set of data as if it were to be calculated from some set of Gaussian processes. GMMs try to maximise $P(X \mid \pi, \mu, \Sigma)$, where x is the data to be fitted, π is the probabilities of originating from each Gaussian process, μ are the means (centre) of the Gaussians, Σ is the covariance matrix for the Gaussians to be fitted. To put it in simpler terms: Maximise the likelihood of generating dataset X from K Gaussian processes given some parameters. The aforementioned formula is expanded in equation (10)

$$P(X \mid \pi, \mu, \Sigma) = \prod_{n=1}^{N} \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(X_n \mid \mu_k, \Sigma_k) \right] \quad (10)$$

To put equation (10) into words, the probability of generating the dataset given some parameters is the product of the sums of the likelihood of the data being generated by each Gaussian, where N is the number of points of data and K is the amount of gaussian components.

This maximisation of probability can be found by using an algorithm called Expectation-Maximization, or EM for short. EM works by first guessing some initial parameters for the k Gaussian processes. These initial parameters are then used for the E-step where the posterior probability is calculated. Then the just calculated posterior is used to update the Gaussian process parameters in the M-step. The E and M steps are then repeated until convergence.

The E-step for Gaussian mixture models is shown in Equation (12), where a probability matrix is calculated based on some prior probability, resulting in a posterior probability, that can then be used to update the parameters of the Gaussian processes in the M-step.

$$Z_{nk} = \left\{ \begin{matrix} 1 \text{ if } X_n \text{ in class } k \\ 0 \text{ if not} \end{matrix} \right\} \quad (11)$$

$$\gamma(Z_{nk}) = P(Z_{nk} = 1 \mid X_n) \quad (12)$$

In the M-step of the EM algorithm, the posterior probability calculated in the E-step is used to update the parameters shown in equation (13), which are all derived from equation (10).

$$\left\{ \begin{matrix} \mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(Z_{nk}) x_n \\ \Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(Z_{nk}) (x_n - \mu_k)^2 \\ \pi_k = \frac{1}{N} \sum_{n=1}^{N} 1 \cdot Z_{nk} \end{matrix} \right\} \quad (13)$$

I) *Deriving gamma from bayes formula:*

Bayes formula is denoted as such: $P(B|A) = \frac{P(A|B)P(B)}{P(A)}$. If we expand the Bayesian formula with respect to the Gaussian Mixture Model goal of finding the posterior probability of whether an observation X_n is in class k (this being denoted as $P(Z_{nk} = 1 \mid X_n)$), we find that the bayesian formula is denoted in equation (14)

$$\frac{P(X_n \mid Z_{nk} = 1) P(Z_k = 1)}{\sum_{j=1}^{K} P(Z_{nj} = 1) P(X_n \mid Z_{nj} = 1)} \quad (14)$$

As shown in equation (14), the posterior probability of an observation being in class k is $P(X_n \mid Z_{nk} = 1)$, or the probability of X_n being generated given that it was generated from the gaussian process belonging to k, times $P(Z_k =$

1) being the probability that any observation is in class k, that term is then divided by the sum of the probability of being generated by all gaussian processes, being the divisor in equation (14). By using Bayes rule, we can surmise that equation (14) results in $P(Z_{nk} = 1 \mid X_n)$.

D. Convolutional Autoencoder

We employ a convolutional autoencoder for non-linear dimensionality reduction. It is evident that using the raw heatmaps as input for the classifiers, risk provoking *the curse of dimensionality*, given the heatmap resolution (42×30) gives rise to a feature vector of length 1260. Considering we have 2132 labelled corner kicks, it seems excessive to have half as many features per set piece as the total amount of set pieces. The Convolutional Autoencoder (CAE) is a deep learning model based on an autoencoder architecture [30].

An autoencoder is a type of neural network, and the objective is to learn a latent representation (called Z or Code) which allows for the accurate reconstruction (decoding) of the original input. The two main parts of an autoencoder are the encoder and decoder. The encoder maps the input data to a lower-dimensional latent space, and the decoder reconstructs the original data from this latent representation [31].

The primary objective of an autoencoder is to minimize the reconstruction loss, which measures the difference between the input data and its reconstruction. This is typically achieved by minimizing the binary cross entropy (BCE) between the original and reconstructed data. In general, an autoencoder introduces an *information bottleneck* to enforce dimensionality reduction. The information bottleneck is achieved by setting the dimension of the hidden layer to be strictly lower than the dimensions of the input layer. An autoencoder with a lower dimension in the hidden layers than the input layer is said to be *undercomplete* [31].

While traditional autoencoders work well with flat, vectorized data, they struggle with image data where spatial hierarchies are crucial. Convolutional autoencoders (CAEs) address this by incorporating convolutional layers into the encoder and decoder networks [30], [32].

In a CAE, the encoder consists of convolutional layers that apply a series of filters (kernels) to the input image, producing feature maps. These convolutional layers are typically followed by pooling layers that reduce the spatial dimensions of the feature maps, promoting translational invariance and reducing computational complexity [33].

For instance, consider an input image of size $N \times M$. A convolutional layer with K filters, each of size $k \times k$, slides these filters over the input image, generating K feature maps. If $N > k$ and $M > k$, the convolution operation results in feature maps that retain the spatial hierarchy of the input image while reducing its dimensionality.

Table 13: CAE hyperparameter configuration

Parameter	**Values**
optimizer	Adam
learning_rate	0.001
kernel	(3,3)
pooling size	(2,2)
pooling_type	max
epochs	1000
loss	binary_crossentropy
batch_size	128
n_layers	1
n_units	8
activation	relu
output_activation	sigmoid

The latent representation of a CAE captures the most salient features of the input image. This compressed latent representation is then passed to the decoder for reconstruction.

The decoder in a CAE performs the inverse operations of the encoder. It uses transpose convolutions (also known as deconvolutions) to upsample the latent representation back to the original image size. The goal is to reconstruct the input image from the compressed feature maps by minimizing the BCE loss [30].

The heatmap lends itself nicely to CAEs as the representation, since in essence, it is an image.

E. CAE configuration

The CAE employed as part of the heatmap feature representation is configured as seen in Table 13.

F. CAE Reconstruction

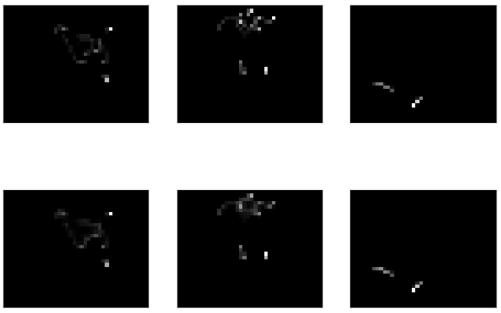


Figure 8: Final (1000 epochs) CAE reconstruction
Top is the input image, bottom is the reconstructed output.