Machine Learning (2024) 113:8687–8709
https://doi.org/10.1007/s10994-024-06606-y

Towards a foundation large events model for soccer

Tiago Mendes-Neves[1,2] · **Luís Meireles**[3] · **João Mendes-Moreira**[1,2]

Received: 7 July 2023 / Revised: 30 April 2024 / Accepted: 4 August 2024 /
Published online: 13 September 2024

Abstract
This paper introduces the Large Events Model (LEM) for soccer, a novel deep learning framework for generating and analyzing soccer matches. The framework can simulate games from a given game state, with its primary output being the ensuing probabilities and events from multiple simulations. These can provide insights into match dynamics and underlying mechanisms. We discuss the framework's design, features, and methodologies, including model optimization, data processing, and evaluation techniques. The models within this framework are developed to predict specific aspects of soccer events, such as event type, success likelihood, and further details. In an applied context, we showcase the estimation of xP+, a metric estimating a player's contribution to the team's points earned. This work ultimately enhances the field of sports event prediction and practical applications and emphasizes the potential for this kind of method.

Keywords Large events model · Event data · Event prediction · Soccer analytics · Deep learning

1 Introduction

A foundation model is a machine learning model that is pre-trained on massive amounts of data and can be fine-tuned to perform various downstream tasks effectively. Large Language Models (LLMs), such as GPT-3 (Radford et al., 2018; Brown et al., 2020),

Editors: Philippe Lopes, Werner Dubitzky, Daniel Berrar, Jesse Davis.

✉ Tiago Mendes-Neves
tiago.neves@fe.up.pt

Luís Meireles
luiscunhameireles1@gmail.com

João Mendes-Moreira
jmoreira@fe.up.pt

[1] Faculdade de Engenharia, Universidade do Porto, Porto, Portugal

[2] LIAAD, INESC TEC, Porto, Portugal

[3] Nordensa, Cluj, Romania

are prime examples of foundation models, showcasing their versatility through diverse applications in natural language processing, code generation, translation, and more.

However, LLMs may not excel in specific specialized tasks (Huang and Chang, 2023; Valmeekam et al., 2022), particularly in soccer. For instance, predicting the outcome of a soccer match remains challenging for these models due to the complex interplay of factors that influence game results.

The limitation arises from LLMs being designed to learn human communication rather than the specific "language" of soccer: soccer data typically consists of event or tracking data, which are not natural language constructs. Event data encompasses discrete in-game actions such as passes, shots, and fouls, while tracking data captures continuous player and ball positions throughout a match. By developing a model that understands the language of soccer events, we can create a generative events model for soccer, unlocking a wide array of powerful downstream applications in analysis and prediction.

Drawing inspiration from LLMs, we can adapt their approach to the soccer language. LLMs leverage the context of previous words to predict the next word in a sequence. Similarly, Large Events Models (LEMs) predict the subsequent event given the current game state.

To formalize this idea, let's represent a game as a sequence of events, $E_1, E_2, ..., E_n$. Our objective is to design a model that, given a partial sequence $E_1, E_2, ..., E_k$ (where $k < n$), predicts the next event E_{k+1}. This approach differs from Markov models (Rudd, 2011) since our approach does not assume independence from states previous to E_k.

Event forecasting in soccer matches holds great importance but is riddled with challenges. Soccer matches are influenced by numerous variables, both on and off the field, including player and team attributes, weather conditions, and psychological factors. Data availability and quality can significantly impact the accuracy of event predictions, making it crucial to obtain reliable and comprehensive datasets.

This paper proposes a LEM, composed of a set of models for generating subsequent events in soccer matches. In addition, we have developed a framework that utilizes this set of models capable of forecasting events in soccer matches. This framework is designed to simulate games starting from a user-defined game state, simulating the game multiple times, with all simulations beginning at the specified game state. The framework outputs the resulting probabilities from the simulations and the events generated for every game. This allows for inspecting and interpreting the patterns that lead to the different outcomes, providing valuable insights into the game dynamics and underlying processes.

We consider the proposed LEM framework a foundation model due to its applicability to a wide range of practical scenarios and its ability to support various downstream tasks. Furthermore, the models support fine-tuning, allowing the framework to provide insights into specific contexts (Radford et al., 2018). Applications, such as action value estimation (i.e., how much value an event brings to a team), game forecasting, and anticipating in-game patterns, can provide a deeper understanding of the game, improve decision-making, and offer strategic advantages for players, teams, and analysts. LEMs contrast with approaches in sports analytics in the sense that currently, for each new prediction task, a new specialized model is required. With LEMs, the same model provides multiple insights. For example, when building a game forecasting model, LEMs provide the probabilities for each outcome but also event data that can be used to build analytics, such as which team is likely to dominate possession, how many shots we expect for each team, where can we expect the opposition to create a threat to our defense, among others.

The research questions we aim to address in this study are:

- Can we accurately generate events from a given game state?
- How do we use the generated events to compute metrics that summarize player performance?

The main contributions of our work are as follows:

- *Deep learning models* We developed a set of PyTorch-based deep learning models tailored for event prediction. The models are designed to capture the complex relationships between various in-match variables and the likelihood of specific events occurring, enhancing both their predictive capabilities and inference speed compared to alternatives, with the latter being crucial for practical applications. The combination of these sequential models is referred to as LEM.
- *Novel framework* We propose a unique and innovative framework for predicting the next events in soccer matches, integrating game state data and deep-learning models. This framework offers a comprehensive, adaptable, and efficient solution for applying event forecasting for many applications in soccer.
- *Practical applications* Our novel framework focuses on its applicability across various domains, such as action value estimation, match prediction, and many other applications related to the use of data and statistical methods to measure performance and make informed decisions in soccer. In this context, we introduce the Expected Points Added (xP+) metric, which utilizes the framework potential to estimate a player's contribution to the team's performance. This emphasis on real-world impact bridges the gap between academic research and industry needs, promoting the adoption of advanced predictive techniques in sports-related domains.

The structure of the paper is organized as follows:

- Section 2 examines existing research on both methodologies related to soccer analytics and relevant deep learning techniques. It also discusses how the proposed approach impacts the state of the art.
- Section 3 formalizes the proposed event forecasting models.
- Section 4 provides a high-level introduction to our proposed framework, detailing its features, approach, and implementation details.
- Section 5 explains the methodology behind our framework's deep learning models, data processing, model optimization, and evaluation techniques.
- Section 6 presents the results of our models and framework, including a practical application of estimating the xP+ for each player.
- Section 7 provides a critical evaluation of our framework and its practical applications, addressing its limitations, potential improvements, and use cases.
- Section 8 summarizes the main contributions and findings, highlighting the significance of our work in advancing sports event prediction and its practical applications while highlighting venues for future research.

2 Literature review

Soccer analytics has evolved rapidly over the past few years, fueled by the increasing availability of data on player and team performances and advances in machine learning and artificial intelligence techniques (Tuyls et al., 2021). This literature review aims to provide an overview of the existing research and methodologies in soccer analytics and event prediction while highlighting the impact and improvements offered by our proposed framework.

Researchers have explored various machine learning techniques for soccer analytics, including decision trees (Decroos et al., 2019; Garnica-Caparrós and Memmert, 2021) and neural networks (Garnica-Caparrós and Memmert, 2021; Fernández and Bornn, 2021; Wang et al., 2024). These approaches have shown promise in predicting match outcomes and player performance. Still, they cannot predict individual events within a match, i.e., the techniques can forecast how many goals a player will score during a season but do not provide any information on which games the player is likely to score and how many.

The emergence of event and player tracking data has spurred interest in modeling the interactions between players and the ball during soccer matches. Event data records discrete in-game actions such as passes, shots, and fouls, while player tracking data provides continuous information on player and ball positions throughout the match. A variety of techniques has been applied to analyze these data types, including Markov models (Cervone et al., 2014), network analysis (Clemente et al., 2015), and deep learning (Merhej et al., 2021; Hubáček et al., 2019; Vaswani et al., 2020; Mendes-Neves and Mendes-Moreira, 2020).

Deep learning, in particular, has gained traction due to its ability to learn complex patterns in large datasets. Convolutional neural networks (CNNs) have been employed to analyze player tracking data and predict player movements (Fernández et al., 2021), while recurrent neural networks (RNNs) and their variants, such as long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), gated recurrent units (GRUs) (Cho et al., 2014), and transformers (Vaswani et al., 2017) have been utilized to model sequences of events (Yeung et al., 2023; Simpson et al., 2022).

Our proposed framework addresses these limitations by borrowing the same concept as LLMs: leveraging previous words' context to forecast the next word's probabilities. The non-deterministic approach of LLMs is responsible for the creativity attributed to these models. For example, it enables ChatGPT to generate different answers to the same prompt. By applying similar concepts to soccer event prediction, we aim to develop a generative model capable of accurately predicting the next event and including variability in its outputs to simulate diverse scenarios.

The works most similar to our proposal are Yeung et al. (2023) and Simpson et al. (2022). Focusing on Simpson et al. (2022), our proposal differs in some key aspects: (1) we forecast all 33 event types available in our dataset, while Simpson et al. (2022) forecasts only 4, (2) their game state is defined by the last 40 events, while our game state is defined using only the last event. Yeung et al. (2023) extends the work of Simpson et al. (2022), keeping the restriction of 4 offensive event types and, in addition, the author groups action coordinates to zones on the pitch, reducing the complexity of the event forecasting problem. Our approach uses raw coordinates instead of grouping since they contain key information for accurate event forecasts. These differences have several consequences: due to different methodologies, we cannot compare model accuracy, but we expect our model to be less accurate, primarily attributed to forecasting more complex variables and due to a

much simpler network architecture. However, our approach gains a lot regarding inference speed and the number of applications. For our goals, the benefits outweigh the accuracy loss.

The potential of our approach is tremendous. Through simulation, we can forecast the value of each game state in terms of points or likeliness to score in the same possession, providing an alternative to frameworks such as VAEP (Decroos et al., 2019) and xT (Singh and Karun, 2019). Furthermore, LEMs can estimate the value of specific game situations, like set pieces (Shaw and Gopaladesikan, 2020). Other metrics, such as expected goals and expected assists, can be estimated through simulation, as well as tactics and strategic insights that can be obtained through the analysis of fine-tuned models (Mendes-Neves et al., 2021).

The literature on soccer analytics has made significant strides in recent years. Still, there remains a gap in developing generative models that can forecast a wide range of events and scenarios within a match. Our proposed framework aims to bridge this gap by adopting basic principles from LLMs and applying them to soccer event prediction. By offering a more flexible approach compared to specialized models, our framework opens the door to numerous practical applications and deeper insights into soccer dynamics. As the field of soccer analytics continues to evolve, our work stands as a promising step towards more comprehensive, nuanced, and actionable event prediction that can benefit players, teams, analysts, and various stakeholders in the soccer ecosystem.

3 Problem formulation

In this section, we delve into the specifics of our approach to predicting the outcomes of a soccer match. We define the Game State, detailing the input data and explaining how we employ event data to anticipate the subsequent soccer event.

3.1 Defining game state

We start by describing the input data used for our problem formulation. We utilize event data to forecast the next soccer event based on the description of the most recent event. This setup allows our model to learn from the temporal patterns in the events, which is crucial for making accurate predictions in the dynamic context of a soccer match. The game state, our input data, is formally defined in Eq. 1, where, G_i is the game state before the event i and F is a subset of features describing the game state. Note that each subset F can contain more than one variable.

$$G_i = (F_1, F_2, ..., F_8) \tag{1}$$

The description of each subset F is as follows:

1. *Event type* A categorical feature representing the type of event that occurred. There are 33 possible types of events, including passes, shots, fouls, and many others. We did not group similar event types (e.g., passes and smart passes) because their underlying characteristics have different distributions, e.g., the relationship to location and current score. For learning purposes, this variable is one-hot encoded to 33 binary variables.
2. *Period* Represents the period of the match, either the first half (0) or the second half (1). It does not account for extra time or penalty shootouts.

3. *Minute* A numerical feature recording the minute of the game during which the event occurred. It ranges from 0 to the maximum duration of a match (i.e., the length of a match plus any additional time). Note that this variable also contains the second in the decimal part. For learning purposes, this variable is normalized by dividing by 60 to ensure that nearly all instances fall in the 0–1 interval.
4. *X* and *Y* Continuous features representing the spatial coordinates where the event took place on the pitch. They are values ranging from 0 to 100, with the origin (0,0) at the bottom left corner of the pitch and (100,100) at the opposite corner. The location of an event is always made according to the perspective of the team performing it. This means that teams attack from x=0 to x=100. For learning purposes, these variables are normalized by dividing by 100, ensuring they are in the interval 0–1.
5. *IsHomeTeam* A binary feature indicating whether the event was performed by a player from the home team (1) or the away team (0).
6. *IsAccurate* A binary feature denoting whether the event was executed accurately (1) or not (0). For example, in the case of a pass, it would be marked as accurate if the pass reached its intended target.
7. *IsGoal* A binary feature is set to 1 if the event resulted in a goal and 0 otherwise.
8. *HomeScore* and *AwayScore* Numerical features representing the current score of the home team and the away team, respectively, at the time of the event. These variables are normalized for learning purposes by dividing them by 10, ensuring that they fall (mostly) in the interval [0,1].

Note that we do not include information about the end coordinates or the identity of players. The reason for not including end coordinates comes from the fact that, in most situations, the end coordinates of an event are the starting coordinates of the following event. Therefore, we would add a substantially higher computational cost to our model with little practical use. Regarding the inclusion of player identity in the LEMs, it remains an open problem. However, for most applications, the player identity is not required. For example, the VAEP model (Decroos et al., 2019) does not include player identity in the model, but is able to rank soccer players according to their skill level.

3.2 Desired output

The objective of our problem is to predict the next event in the soccer match, given a sequence of game state instances. For this, we forecast the state-change array $\hat{e}$. The $\hat{e}$ is an array containing the forecasted probabilities for each variable, formally defined in Eq. 2, where n is the number of previous game states used to forecast. In this proposal, we use $n = 1$, enabling smaller networks that increase inference speed and reduce the risk of overfitting the models. Disadvantages to using a single game state for forecasting include the models being less accurate and some event forecasts not being coherent with the previous chain of events. Increasing n is a path for future improvement but requires larger datasets and more computational power.

$$\hat{e}_i = f(G_i, G_{i-1}, ..., G_{i-n}) \tag{2}$$

The $\hat{e}$ contains all the information required to estimate the next event. The probabilities related to variables *EventType*, *IsAccuracte*, *IsGoal*, *X*, *Y* and *IsHomeTeam* are directly forecasted by the models. Instead of forecasting *Period* and *Minute*, we forecast the *TimeElapsed* variable, which is the temporal distance between two events. We do not

forecast *HomeScore* and *AwayScore* since they are a consequence of other predicted variables.

The non-binary variables are one-hot encoded. This allows us to obtain the individual probabilities for each category. For example, for each forecasted event, we know the probability for each event type. The same happens with *TimeElapsed* (we know the probability of the next action being 0, 1,..., 60 s after) and with *X* and *Y* (we know the probability of the next location being 0, 1,..., 100).

We need to make three sequential inferences to build $\hat{e}$. First, we infer the next action *EventType* using the game states (Eq. 3). Then, we infer the accuracy of the events, *IsAccurate* and *IsGoal*, by using the game states and the forecasted event type (Eq. 4). At last, we forecast the remaining variables, *TimeElapsed*, *X*, *Y* and *IsHomeTeam*, using all information available about the event (Eq. 5). From a practical standpoint, we present the data flow in Fig. 1.

$$\hat{e}_{i(type)} = f(G_i, G_{i-1}, ..., G_{i-n}) \tag{3}$$

$$\hat{e}_{i(accuracy)} = f(G_i, G_{i-1}, ..., G_{i-n}, \hat{e}_{i(type)}) \tag{4}$$

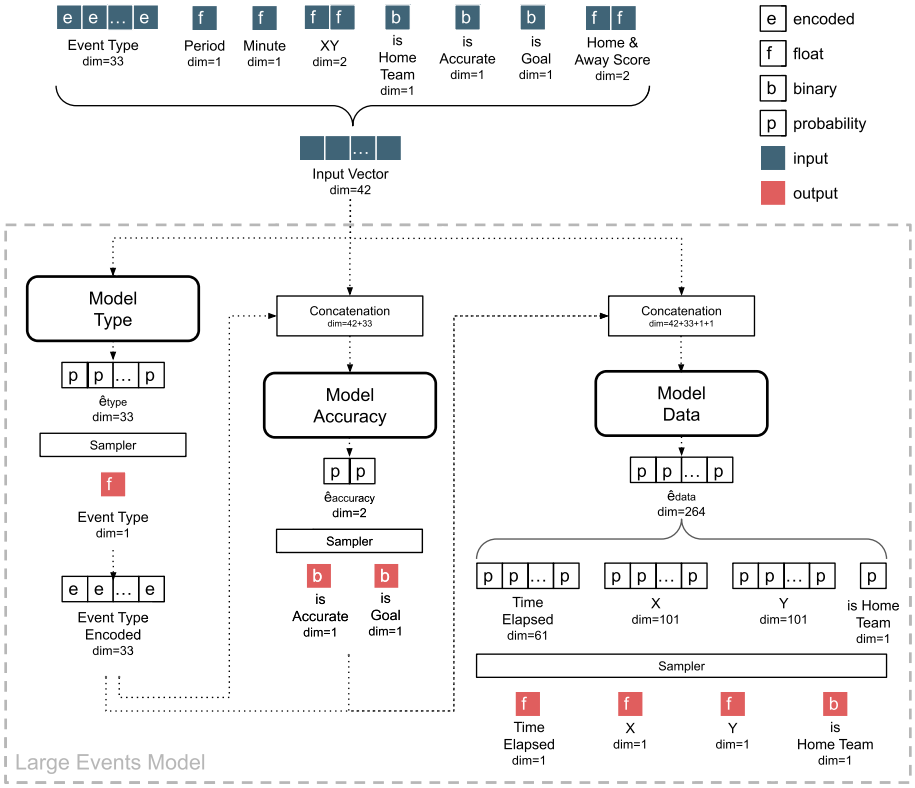

Fig. 1 A schematic representation of the data flow of our LEM. In blue, we present the composition of the input vector that contains the context for our models. The data is fed sequentially through the three models, with each model requiring the outputs of previous models to make an inference. After the data flows through all the models, we extract the outputs highlighted in red to generate the vector $\hat{e}$

$$\hat{e}_{i(data)} = f(G_i, G_{i-1}, ..., G_{i-n}, \hat{e}_{i(type)}, \hat{e}_{i(accuracy)}) \tag{5}$$

The reason to forecast the variables separately is to take advantage of their dependence. For example, we know that, on average, a shot is much less likely to be successful than a pass. Therefore, forecasting *EventType* first and then forecasting *IsAccurate* with this information is a much easier problem to learn than forecasting all components with the same model. This approach allows us to condition the probabilities, making it easier for events to follow the expected behaviors. This means that event coordinates are forecasted with knowledge of the current event. If the forecasted type is a corner kick, the forecasted distribution of *XY* coordinates will be significantly different compared to forecasting a pass.

Another important aspect is that the variables are **probabilistically sampled**, with the probability forecasted by the model. Therefore, our approach is **not deterministic**. This is crucial to increase the spectrum of applications that our model covers.

Given these steps, we can now forecast the next event $\hat{E}$ by changing the previous game state using the state-change array (Eq. 6).

$$\hat{E}_i = f(G_i, \hat{e}_i) \tag{6}$$

4 Framework overview

This section provides an overview of how the framework operates. Figure 2 provides the framework's architecture.

Upon receiving the user's input data, which can be either a single event or an array of events, the first step in our framework is to ensure that this data is in a format compatible with our models. The data preprocessing component handles this task.

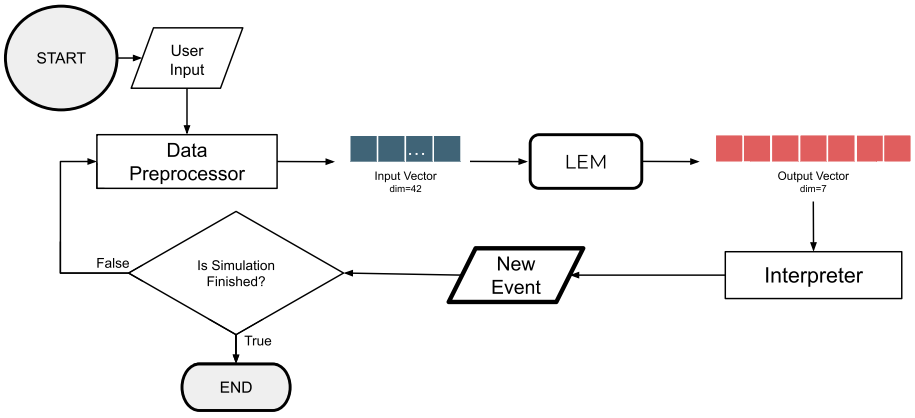

Fig. 2 Fluxogram illustrating the workflow of the proposed framework. The process starts with user input data that undergoes data preprocessing, ensuring it is in the correct format for the models. This data is then fed to the LEM, generating predictions $\hat{e}$ for the state-change array. These predictions are passed through an interpretation function which converts the state-change array into a human-readable event standard. From there, either the process is repeated by converting the new event to the input format or the simulation ends. The simulation is stopped only when the user-defined criteria are met or when the match ends

If the input data is not already in the required format, the data preprocessing component performs the necessary transformations. This involves encoding categorical variables, normalizing numerical features, and structuring the data so that each instance represents a game state as defined in Sect. 3. The preprocessing stage is crucial to the functioning of the models, as it ensures they receive data in a consistent format that they can correctly interpret.

After data preprocessing, the cleaned data is fed sequentially through the three models contained in the LEM, as visible in Fig. 1. The output from the models is a probability distribution over all possible next events. Given the current game state, this distribution represents the model's predictions about the next event. We pass each distribution's probabilities through a sampler that chooses an action according to the probabilities. For binary variables, we sample from a Bernoulli distribution. For multicategory variables, we sample from a Multinomial distribution.

Once the model predictions are obtained, they are passed through an interpretation function. This function transforms the model's output back into the format that can be used as input for the next iteration. The steps required are to update the time-related variables and update the score. Time-related variables are updated by increasing the *Minute* variable by the number of seconds from *TimeElapsed*. If the number of minutes is over 45, we assume the game will either change status to the second half or end in case we were already in the second half. This is an oversimplification of the game. Ideally, including more previous game states would allow us to forecast injury time. However, this would result in a substantial increase in computational and data requirements that are out of the scope of this proposal. To update the *HomeScore* and *AwayScore*, we check if the game is still running, if there is an event that can be a goal, and if the *IsGoal* variable is true. If all these checks are passed, a goal is added to the team performing this action.

When we get a new event, we can use this event to update the game state. This process can be repeated to simulate the sequence of events for the rest of the game. After the process is finished, there is an option to transform the output into the human-readable event standard. This format can benefit users who wish to interpret the results without understanding the underlying model representation.

To increase the efficiency of our framework, we perform these operations in parallel. For example, if we wish to simulate a specific game state 1000 times, we can create a matrix that contains 1000 copies of the game state. This procedure considerably speeds up the inference process due to its efficiency in handling the data in tensor-based data structures. The tensor-based implementation allows the model to scale for GPU usage, which becomes relevant when simulating on a large scale. For example, calculating the Expected Points Added (xP+) metric requires valuing every event during a season. In early-stage experiments, we made inferences faster in magnitude 1:1.000.000 compared to Scikit-learn's (Pedregosa et al., 2011) Random Forest (Breiman, 2001). By simulating the model's output in this way, the framework can generate a variety of potential game states, which can be interpreted using different methods to generate valuable insights.

5 Methodology

This section details the methodology employed to train the models in our framework, highlighting the data, hyperparameter optimization, and training steps.

5.1 Datasets and data splits

The data used for training and testing the models come from the Wyscout open dataset (Pappalardo et al., 2019). This dataset contains detailed event data for several major European soccer leagues for the 2017/18 season. For our training set, we used data from the French Ligue 1, German Bundesliga, and the Italian Serie A, containing a total of 1.718.640 events. The test set consists of data from the English Premier League and Spanish La Liga (1.214.838 events). We opted for a competition-level split to facilitate potential future work on LEMs by leaving full-season data out of the training set. Furthermore, it provides a full season for testing, which is crucial for many applications and use cases, including benchmarking metrics. However, we acknowledge that this approach is not the best regarding model accuracy since we force our LEM to forecast data for a league that is out of the train data.

Figure 3 illustrates the distribution of events across the soccer pitch. Due to the dynamic nature of the game, the events are not evenly distributed. Most events are located in a team's defensive half. Furthermore, we can observe errors in the annotations. As human annotators collect data, we observe less frequent actions near the pitch lines. These non-systematic errors are often a result of the annotation methodology (Biermann et al., 2023). Figure 4 shows data for event types, which is also not uniformly distributed. There are still known biases in type annotations, such as the example of

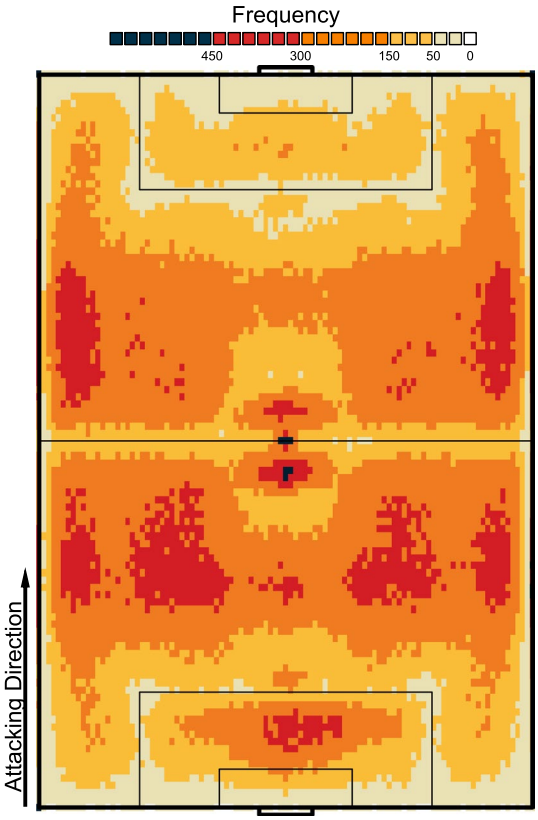

Fig. 3 A heatmap of the distribution of events in our training set on a soccer pitch

crosses that go toward the goal: in the rare occurrence where the ball leads directly to a goal it is considered a shot, while if it does not lead to a goal it is considered a cross (Mendes-Neves et al., 2021).

Figure 5 shows the remaining variables we forecast in LEMs. These two variables display a more uniform distribution of values. However, the number of events drops slightly across the game, with a decay between the two halves, highlighting the importance of including the current period of the game in the game state. Furthermore, the number of events sharply drops after 45 min as soccer matches have additional time that is not consistent across games. The only variable we do not provide a visualization of is the *isGoal* variable. This variable is highly imbalanced (0.3% of the events). The *isGoal* variable is only true for events directly related to goals, like shots or save attempts.

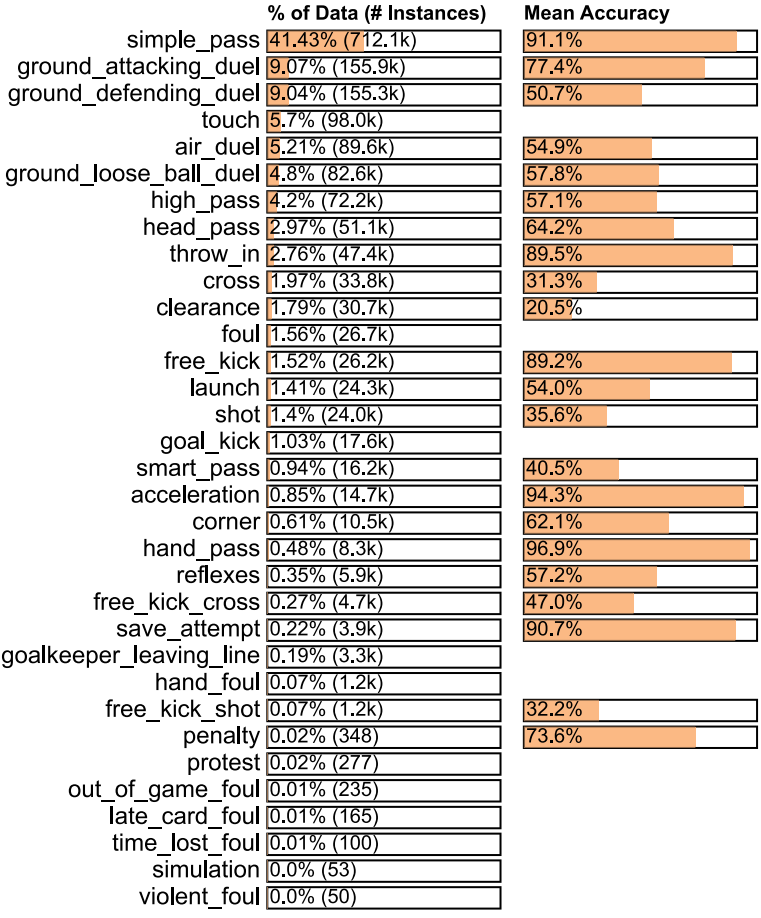

Fig. 4 The decomposition of the events in our dataset across the different types and their respective average accuracy

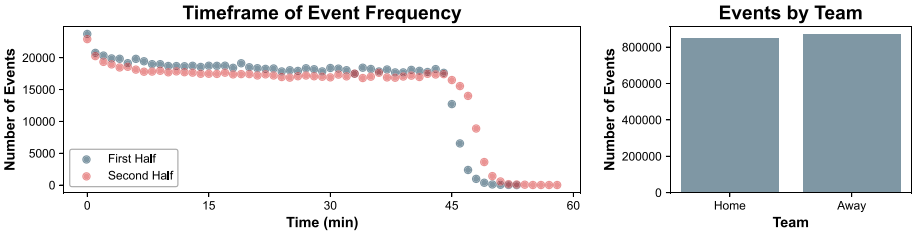

Fig. 5 On the left, we present the number of events in our dataset across the time frame of a match. On the right, we show that the dataset is balanced regarding the number of events for home and away teams

5.2 Model training

We used PyTorch (Paszke et al., 2019), a deep-learning library that provides efficient tensor operations to implement our models. To train our models, we defined a custom class in PyTorch, the *MultiLayerBinaryClassifier*. This class allows us to specify the size of hidden layers and to construct the corresponding model architecture automatically. We used the Xavier uniform initialization for the network weights to ensure that all neurons initially have approximately the same output distribution and improve the convergence rate.

The loss function used for training is Binary Cross Entropy Loss (BCELoss), which is suitable for binary classification problems and provides a measure of the error between the model's predictions and the actual data. As for the optimizer, we used Adam (Kingma and Ba, 2017) due to its efficiency, which is well-suited for big data problems.

Our training regime includes a maximum of 100 epochs with an early stopping patience of 3, meaning that the training will cease if there is no improvement in validation loss for three consecutive epochs.

Even though PyTorch can leverage GPUs for accelerated computation, we chose to train our models on a CPU. At the scale of our problem, using GPU results in performance degradation due to the overhead of transferring data between the CPU and GPU. However, GPUs substantially accelerate inference through parallelization, which is helpful for large-scale operations.

5.3 Hyperparameter optimization

For hyperparameter optimization, we employed Optuna (Akiba et al., 2019), a framework for automated hyperparameter tuning. We selected the Tree-structured Parzen Estimator (TPE) as the optimization algorithm (Bergstra et al., 2011). TPE is a Bayesian optimization algorithm that supports both categorical and numeric parameters. We searched over 40 trials, examining architectures with 1 to 3 layers. Each layer could have a size of 16, 32, 64, 128, or 256 neurons. The learning rate was sampled from the interval 1e-5 to 1e-1, and batch sizes were 32, 64, 128, 256, 512, and 1024. The activation function for each layer could be either ReLU, sigmoid, or tanh, with the final layer always being sigmoid.

The size of layers and batch were limited to powers of 2 to narrow the search space, thereby promoting faster convergence. The hyperparameter search was conducted on the Italian Serie A data, with a 70-30 data split for training and validation. To avoid overfitting, we introduced a penalty for model complexity by adding 0.05% to the error per layer.

Table 1 Specifications of the final models after hyperparameter optimization

Model	In	Out	Layers	L. Rate	Batch	Activ.	Loss	Loss/Out
Type	42	33	[256]	0.0010	32	sigmoid	1.5009	0.0455
Accuracy	75	2	[128]	0.0410	1024	sigmoid	0.0038	0.0019
Data	77	264	[64, 256, 256]	0.0063	1024	relu	12.755	0.0483

Table 2 This table presents the performance metrics accuracy and F1 score (average weighted) for each forecasted categorical variable and the respective baselines

Model	Accuracy (%)				F1-score (%)			
	Type	Accurate	Goal	Home	Type	Accurate	Goal	Home
Majority Class	40.8	67.8	99.7	50.9	23.6	80.8	0	67.5
Decision Tree	36.2	74	99.7	92.6	36.8	80.65	59	92.6
Random Forest	49.9	80	**99.8**	–	46.6	85.7	68.4	–
Default MLP	55.5	**81.7**	**99.8**	93.6	49.8	87.2	**68.8**	93.7
Our Model	**55.7**	**81.7**	**99.8**	**93.8**	**49.9**	**87.3**	68.5	**93.8**

The results shown in bold represent the best performing model for the respective metrics

Missing values on Random Forest due to memory overflow errors

5.4 Final model specifications

After hyperparameter optimization, the best-performing models were selected for each task. The specifications of these models are summarized in Table 1.

In the following sections, we will discuss the results obtained from these models.

6 Results

In this section, we present our machine learning model's results, which include a variety of metrics, transition maps, and simulation results that help understand the model's performance and the relationships it has learned. We also developed an xP+ model to show our approach's flexibility and potential impact.

6.1 Model test performance

Tables 2 and 3 present the variable-specific metrics used to evaluate the performance of our model on the test set. We computed different metrics for each variable, such as accuracy and F1-score for classification and R2-score and Mean Absolute Error for regression. These metrics provide a comprehensive understanding of how well our model predicts each variable. For each variable, we provide several baselines: (1) forecasting the majority class (for classification problems) or the average value of the variable (for regression problems), (2) decision tree, (3) Random Forest (Breiman, 2001), and (4) a multi-layer perception. We

Table 3 This table presents the performance metrics R2 score and mean absolute error (MAE) for each forecasted continuous variable and the respective baselines

Model	R2-score (%)			Mean Absolute Error		
	X	Y	TimeElapsed	X	Y	TimeElapsed
Average Value	0	0	0	0.212	0.265	0.051
Decision Tree	47.1	17.7	51.3	0.112	0.179	0.033
Random Forest	–	–	–	–	-	- -
Default MLP	59	27.5	52.4	0.097	0.165	0.027
Our Model	**63.6**	**29.2**	**55.2**	**0.085**	**0.156**	**0.026**

The results shown in bold represent the best performing model for the respective metrics

Missing values on Random Forest due to memory overflow errors

selected these machine learning algorithms as baselines due to their support of multi-label classification. All baseline models were configured with the default parameters provided by Scikit-learn (Pedregosa et al., 2011). The Random Forest model has memory overflow issues while training the model Data, caused by the high amount of data and dimensionality of the problem. The requirements to run a 100 estimator model would surpass 300GB. Therefore, the respective metrics are missing from Tables 2 and 3.

As expected, our model outperforms the baselines in nearly every metric. The performance difference is more visible in the continuous variables, where the necessary complexity of the model increases substantially. Note that performance is only a part of the decision about which model to use, with inference speed being a key factor. The amount of inferences required for the work described in Sect. 6.4 required a day using our model and would need upwards of one month for any other option.

6.2 Transition maps

To validate the model, we visualize the transition maps that the model has produced. Figures 6, 7, 8, and 9 showcase different transition aspects of our model, from understanding what event types are likely to be generated knowing the current event type, changes in location coordinates, how much time elapses between actions and how our model interprets the expected goals metric.

6.3 Simulation results

Figure 10 shows how well the results of simulated games match the distribution of the actual outcomes. The closer these two distributions are, the more accurate our model is.

6.4 Building an xP+ metric with simulation

To showcase the possibilities of our framework, we will present a use case where we estimate how many points each player individually added to his team in the season.

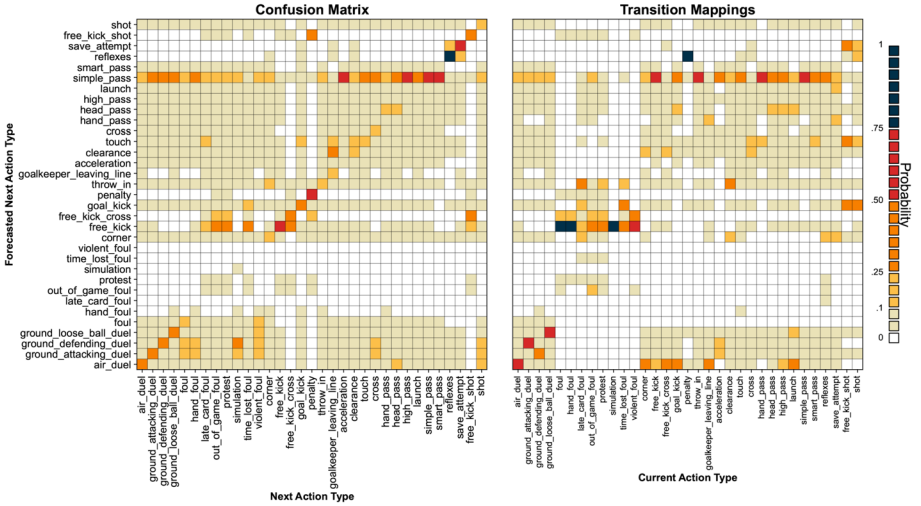

Fig. 6 On the left is the confusion matrix between the forecasted event type and the actual event type of the next action. On the right are the transition mappings between the current event type and the forecasted probabilities

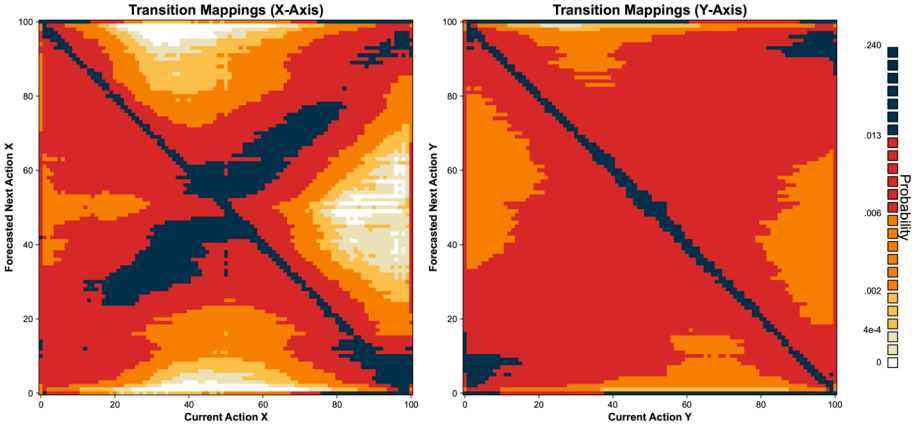

Fig. 7 The probability of transitioning from current location x,y to the next location x,y. The pattern contains two behaviors: (1) the positive correlation between the current coordinates and the next coordinate, as the next event performed by the same team is expected to be close to the current event, and (2) a negative correlation caused by when the next event is performed by the opposite team, as the coordinate axis changes to the opposition's perspective. Note: the probability color scale is on a logarithmic scale and ranges from 0 to 0.24

The first step to calculate this metric was to estimate how many points a team was expected to obtain for every given state. We computed the game state for every event in the test set and simulated the game until the end. We ran the simulations for 24 h on an NVIDIA RTX3060, resulting in 253 total simulations per game state, a total

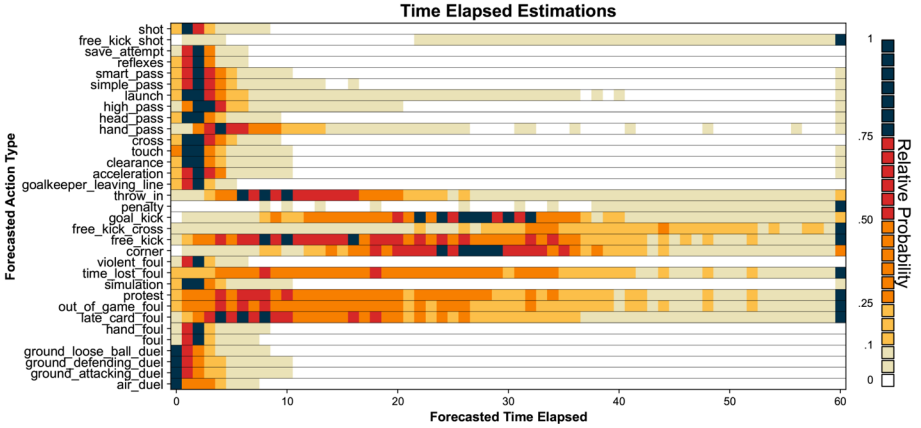

Fig. 8 The distribution of the forecast for *TimeElapsed* for the events in the test set. The relative probability corresponds to the forecasted probability divided by the maximum probability of the forecasted action type

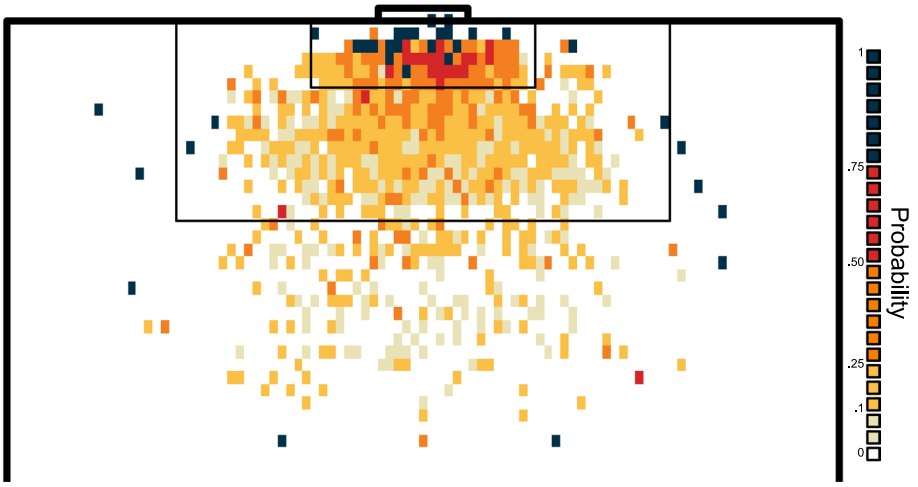

Fig. 9 A representation of the average probability of scoring for each location in the pitch, estimated by forecasting the test dataset. Note that the points with unrealistic high probability are due to a known bias in our dataset: when a cross goes in the direction of the goal, it is labeled as a shot if it ends in a goal but is still labeled as a cross if it gets claimed by the keeper (Mendes-Neves et al., 2021)

of 307.354.014 game simulations. Then, we estimate the probability for each match outcome using Eq. 7.

$$\begin{cases} P(HomeWin) = \frac{\#HomeWins}{\#Simulations} \\ P(Draw) = \frac{\#Draws}{\#Simulations} \\ P(AwayWin) = \frac{\#AwayWins}{\#Simulations} \end{cases} \tag{7}$$

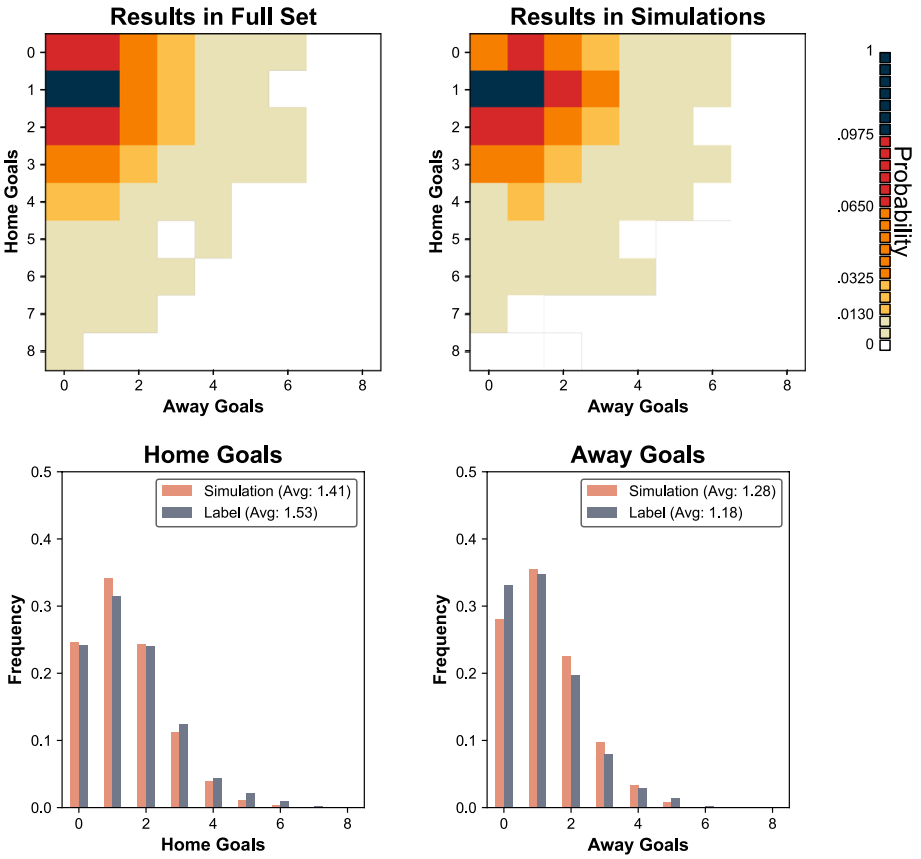

Fig. 10 Comparing the distribution of actual results from our dataset against the simulated games (10,000 simulations). The visible differences are for the following scores: (0–0) and (1–0) are less frequent, while (1–2) and (1–3) are more frequent than in reality. Other differences occur in relatively rare results (e.g., 7–0). Note: the probability color scale is on a logarithmic scale

For the given probabilities, we can estimate how many points each team is expected to earn in the game, given the current game state, as formalized in Eq. 8. Note that in soccer, the match-winning team earns 3 points, while in a draw both teams earn 1 point.

$$\begin{cases} E(HomePoints) = 3 * P(HomeWin) + 1 * P(Draw) \\ E(AwayPoints) = 3 * P(AwayWin) + 1 * P(Draw) \end{cases} \tag{8}$$

Finally, we compute each action's xP+ metric. We compute this metric by estimating the difference between the expected points for the player's team before he intervenes in the

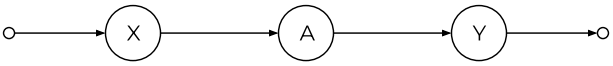

Fig. 11 A sequence of events in the game. To evaluate event A, we simulate the game from state X and state Y, and then assume that the increase in expected points is due to event A

Table 4 The top soccer players based on their xP+ values. The left part of the table corresponds to players in the 2017/18 Spanish La Liga. The table's middle part presents the 2017/18 English Premier League results. The last part of the table presents the VAEP ratings of the 2017/18 English Premier League reported by Decroos et al. (Decroos et al., 2019)

Player	Team	xP+	Player	Team	xP+	Player	Team	VAEP
M Gómez	Celta	18.27	Salah	Liverpool	30.86	Coutinho	Two Clubs	0.899
Messi	Barcelona	17.43	Mahrez	Leicester	30.33	Salah	Liverpool	0.817
Griezmann	Atlético	16.35	De Bruyne	Man City	16.70	De Bruyne	Man City	0.641
Suárez	Barcelona	14.36	Kane	Tottenham	15.78	Hazard	Chelsea	0.636
Bacca	Villarreal	14.04	Arnautović	West Ham	14.37	Mahrez	Leicester	0.635
Isco	R Madrid	12.76	Richarlison	Watford	14.11	Martial	Man United	0.607
Fornals	Villarreal	12.74	Sterling	Man City	13.89	Sterling	Man City	0.579
Rodrigo	Valencia	12.13	Hazard	Chelsea	13.41	Pogba	Man United	0.549
Asensio	R Madrid	11.97	Lanzini	West Ham	11.58	Kane	Tottenham	0.545
S León	Real Betis	11.22	Zaha	C Palace	11.44	Son	Tottenham	0.539

Table 5 The top soccer players ranked by the xP+ value in three different categories: shot, pass, and duel

Player	Shot	Player	Pass	Player	Duel
Salah	15.60	De Bruyne	14.09	Salah	12.66
Kane	14.65	Albrighton	12.55	Stuani	7.43
Messi	14.09	Mahrez	10.69	Richarlison	6.43
Moreno	13.85	Parejo	10.34	Mahrez	6.19
Suárez	13.58	Fornals	9.99	Lennon	5.94
C Ronaldo	13.35	Alba	9.67	Portillo	5.89
M Gómez	12.67	Barragán	9.13	Nolito	5.88
Mahrez	11.21	Pogba	8.55	Benteke	5.33
Sterling	10.67	S Francis	8.37	Casemiro	5.31
ángel	10.41	Lanzini	7.66	U Núñez	5.13

match and after his intervention ends, as formalized in Eq. 9. This means we calculate the expected points for the action occurring before (X) and the expected points for the action occurring after the player action, as visualized in Fig. 11.

$$\begin{cases} xP+_A = E(HomePoints)_Y - E(HomePoints)_X \text{ if action from home team} \\ xP+_A = E(AwayPoints)_Y - E(AwayPoints)_X \text{ if action from away team} \end{cases} \tag{9}$$

In Tables 4, 5, and 6, we present the aggregated sum of all player actions for the 2017/18 season. Note that the xP+ metric is not supposed to give a "best player in the world" ranking: an excellent player in an excellent team will find it challenging to obtain a big xP+ score because there is a limited amount of points being distributed by many high-quality players. This metric is best used with knowledge of team context, as we present in Table 6. Our approach is similar to Kharrat et al. (2020), but we use the LEMs to estimate the current probability of winning for each team.

Table 6 The xP+ values for the top players in the top 4 teams of each test league. ∗P. Coutinho played the first half of the season in Liverpool and the second half in Barcelona. The xP+ value corresponds to the half-season in the club

La Liga	Barcelona		Real Madrid		Atlético Madrid		Valencia	
	Messi	17.43	Isco	12.76	Griezmann	16.35	Rodrigo	12.13
	Suárez	14.36	Asensio	11.97	F Luis	7.81	Neto	8.94
	Alba	11.15	C Ronaldo	10.69	Correa	7.35	Parejo	7.47
	Vermaelen	5.13	S Ramos	9.11	Carrasco	6.26	C Soler	4.84
	Coutinho*	4.93	Bale	8.82	D Costa	4.03	S Mina	4.69
	Squad Total	63.29	Squad Total	73.50	Squad Total	30.45	Squad Total	30.31
Premier League	Manchester City		Manchester United		Tottenham		Liverpool	
	De Bruyne	16.70	Matić	10.74	Kane	15.78	Salah	30.86
	Sterling	13.89	Lukaku	10.34	Alli	9.96	Mané	9.70
	D Silva	10.78	Smalling	8.94	Trippier	9.71	Coutinho*	8.06
	G Jesus	7.02	Lingard	8.26	Eriksen	8.57	Firmino	6.92
	Agüero	6.93	Pogba	6.98	Son	6.95	Gomez	4.83
	Squad Total	80.56	Squad Total	64.86	Squad Total	56.74	Squad Total	49.29

7 Discussion

The results presented in this paper demonstrate the efficacy of our generative LEM in various aspects of soccer analytics. The model's variable-specific performance accurately predicts both categorical and continuous variables, considering that we opted for a single-game state approach. The metrics are substantially above the baseline, which positively impacts our confidence in the results obtained by the framework.

A key limitation is the available data. Specifically, we needed to use cross-league train/test split. Otherwise, we would not have results for an entire season. The fact that the trained models did not have any data regarding the leagues we are trying to use the model in makes it impossible for the models to learn specific play patterns that occur in the league. Furthermore, the architecture of the models is quite simplistic. This results from the lack of data to train a more complex model: we penalized more complex architectures in the training process to avoid the model learning league-specific patterns that would not generate well for the test leagues. We do not see evidence that these limitations affect the quality of the insights. However, we still recommend building models with larger datasets and deeper game states to increase confidence in the insights obtained.

Transition maps provide an interesting perspective into the model's understanding of the game. In the action transition map, we can observe the dominance of the passing action in the game over other actions. Furthermore, the transitions seem to follow the flow of a soccer game. The coordinate transition matrix displays the patterns of the underlying data. The diagonal lines that cut the plots show the effect of changing from one team's perspective to another. The coordinates are always given from the team's attacking perspective, which means that when possession changes, the coordinates change to the symmetric point in the middle of the pitch. The other diagonal shows the transitions when possession changes within the team, where we can observe that the coordinates follow a distribution that resembles a normal distribution, centered slightly in front of the current coordinate.

Regarding event types, our model learned correctly to model the frequency of events, even for events with small sample sizes. For example, penalty kicks (n=193) have an expected number of occurrences of 174.8 in the test set. Furthermore, the model estimates a 71.2% chance of scoring each penalty in the test set, showing that the model is also able to correctly model the outcomes of rare events. The penalty conversion is 73.5% in the train set. While there is a 10% difference between actual and expected occurrences, it is mostly related to the limitations of our training process regarding cross-league training and the inherent variation within soccer.

The time elapsed estimation map provides a good overview of actions resulting in different time distances. We hard-capped the forecast of time elapsed at 1 min, which is a reasonable assumption since only two actions consistently take more than that time to be executed: the penalty and direct free kicks. Note that the introduction of Video Assistant Referees (VAR), which did not exist in the games that composed our dataset, might change the pacing and frequency of certain in-game events. Therefore, special attention should be paid to the time elapsed variable when training models with more recent datasets, which include VAR-related disruptions.

The simulation results corroborate the model's prediction power. The similarity between the distributions of simulated and actual outcomes indicates that the model has a firm grasp on the overall dynamics of the soccer matches. Note that the model simulates the average home team versus the average away team, which generally results in more balanced outcomes.

We provide one of the significant outcomes of our research: the xP+ metric. This metric, calculated through simulations, provides an estimate of the individual contribution of each player to their team's expected points, offering an insightful perspective into the player's contribution relative to their team.

When we analyze the top players based on xP+, it is noteworthy to see some well-known names in soccer, like the top performer Mohamed Salah. However, it's also interesting to observe that some players do not have a high xP+, not because they are bad players but because their impact is lower within the context of their specific team. For example, Mohamed Salah's teammate Sadio Mané could have contributed substantially more if Salah did not collect as many points. Teams can only collect a limited amount of points, which may negatively impact the xP+ of players playing in better teams.

Another interesting aspect is that xP+ identifies several players from lower-rated teams in the top 10. This highlights the value of the xP+ metric in identifying impactful players in more challenging contexts, which traditional metrics might overlook. Players such as Mahrez from Leicester, Arnautović from West Ham, and Richarlison from Watford were all transferred by significant sums in the following seasons. Zaha from Crystal Palace is now considered a legend at the club, having peaked in his market value at 55 M€ as per Transfermarkt. Of the surprises in the top 10, only Lanzini from West Ham did not surpass the 2017/18 level, but for an external reason: the player tore his cruciate ligament at the end of the 2017/18 season, right after joining the Argentinian team for the 2018 World Cup. This is one of the most devastating injuries for a football player, which left him out of competition for eight months, significantly impacting his career. Of these players, only Mahrez is highlighted by Decroos et al. (Decroos et al., 2019) as one of the top players in the Premier League. With LEMs, we have the power to better understand the player context with different metrics.

The xP+ metric we propose can be calculated using multiple approaches. Within our framework, fine-tuning our base models to focus on the player's team data will change the metric to understand better the context where the player is executing the actions: for

example, if a wide player has a fast and smaller striker in his team, he should perform different actions than if he has a slower and taller player. Furthermore, one could combine the player and target team data to study the impact of the player coming into a new team: (1) fine-tune the model with the team data, (2) further fine-tune the model with the player's data, (3) compare the simulated results between the model fine-tuned with the player's data and the model without player fine-tuning. Also, how metrics are computed can differ depending on the use case. For example, we account for the current score of the match. But, for other use cases, the user might want to set a constant result for all actions and evaluate actions independently of the current result.

In conclusion, our framework can serve multiple purposes, understanding and predicting different dynamics in a soccer match. By combining machine learning with extensive simulation, we provided examples of how to quantify individual players' contributions in a novel way. Furthermore, despite the lack of experience from the authors in other sports' data, we believe these methods could be extended to more sports.

8 Conclusion

This research presented a novel framework for predicting various aspects of soccer match events using generative models. The framework contains three deep learning models tailored to predict a specific aspect of a soccer match event, from the event type to its success and detailed information. Our results indicate that the framework can efficiently and accurately simulate soccer games, enabling many applications to be built upon these models.

However, this work is not without limitations. The models' effectiveness is inherently tied to the quantity of available data. Specifically, the available open datasets are insufficient to evaluate tasks such as fine-tuning models to replicate a team's behavior. Despite these challenges, our models provide a solid foundation for further research and development.

Future work will focus on further exploring the applications of LEMs, specifically, how to fine-tune the model to learn specific team patterns without overfitting. We showed that the framework can measure the xP+ metric, but we believe that the framework's impact is much broader. Given the fine-tuned model, we can understand how players would perform in different contexts, analyze how a game between two teams will develop tactically, and forecast soccer matches. We believe these tools and insights could revolutionize how we understand and interact with soccer data.

Author contributions Tiago Mendes-Neves led data preparation, conceptualizing the framework and its applications, and implementation. Also, contributions to data interpretation, discussions, and writing the manuscript.

Luís Meireles focused on aligning the model with practical soccer applications while participating in data interpretation, discussions, and manuscript revisions.

João Mendes-Moreira supervised the project, provided guidance, and played a critical role in data interpretation, discussions, manuscript revisions, and refining the manuscript.

Funding Open access funding provided by FCT|FCCN (b-on). This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020.

Data availability The data is publicly available to reproduce this work (Pappalardo et al., 2019).

Code availability The code and the models are available at https://github.com/nvsclub/LargeEventsModel.

Declarations

Conflict of interest The authors have no Conflict of interest to declare that are relevant to the content of this article.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Akiba, T., Sano, S., Yanase, T., et al. (2019). Optuna: a next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. ACM, Anchorage AK USA, (pp 2623–2631), https://doi.org/10.1145/3292500.3330701,

Bergstra, J., Bardenet, R., Bengio, Y., et al. (2011). Algorithms for hyper-parameter optimization. In Proceedings of the 24th international conference on neural information processing systems. Curran Associates Inc., Red Hook, NY, USA, NIPS'11, (pp. 2546-2554)

Biermann, H., Komitova, R., Raabe, D., et al. (2023). Synchronization of passes in event and spatiotemporal soccer data. *Scientific Reports*. https://doi.org/10.1038/s41598-023-39616-2

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Brown, TB., Mann, B., Ryder, N., et al. (2020). Language models are few-shot learners.http://arxiv.org/abs/2005.14165, arXiv:2005.14165 [cs]

Cervone, D., D'Amour, A., Bornn, L., et al. (2014). Predicting points and valuing decisions in real time with NBA optical tracking data. In Proceedings of the 2014 MIT sloan sports analytics conference (p. 9)

Cho, K., van Merrienboer, B., Gulcehre, C., et al. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. http://arxiv.org/abs/1406.1078

Clemente, F. M., Martins, F. M. L., Kalamaras, D., et al. (2015). General network analysis of national soccer teams in FIFA World Cup 2014. *International Journal of Performance Analysis in Sport, 15*(1), 80–96. https://doi.org/10.1080/24748668.2015.11868778

Decroos, T., Bransen, L., Van Haaren, J., et al. (2019). Actions speak louder than goals: valuing player actions in soccer. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. ACM, Anchorage AK USA, (pp. 1851–1861), https://doi.org/10.1145/3292500.3330758,

Fernández, J., Bornn, L., et al. (2021). Soccermap: A deep learning architecture for visually-interpretable analysis in soccer. Applied Data Science and Demo Track. In Y. Dong, G. Ifrim, & D. Mladenić (Eds.), *Machine learning and knowledge discovery in databases* (pp. 491–506). Cham: Springer.

Fernández, J., Bornn, L., & Cervone, D. (2021). A framework for the fine-grained evaluation of the instantaneous expected value of soccer possessions. *Machine Learning, 110*(6), 1389–1427. https://doi.org/10.1007/s10994-021-05989-6

Garnica-Caparrós, M., & Memmert, D. (2021). Understanding gender differences in professional European football through machine learning interpretability and match actions data. *Scientific Reports, 11*(1), 10805. https://doi.org/10.1038/s41598-021-90264-w

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735https://direct.mit.edu/neco/article/9/8/1735-1780/6109.

Huang, J., & Chang, KCC. (2023). Towards reasoning in large language models: A survey. http://arxiv.org/abs/2212.10403, arXiv:2212.10403 [cs]

Hubáček, O., Šourek, G., Železný, F. (2019). Deep learning from spatial relations for soccer pass prediction. In: Machine learning and data mining for sports analytics. (vol. 11330, pp. 159–166) Springer, Cham. https://doi.org/10.1007/978-3-030-17274-9_14,

Kharrat, T., McHale, I. G., & Peña, J. L. (2020). Plus-minus player ratings for soccer. *European Journal of Operational Research, 283*(2), 726–736. https://doi.org/10.1016/j.ejor.2019.11.026

Kingma, DP., & Ba, J. (2017). Adam: A method for stochastic optimization. arXiv:1412.6980 [cs] http://arxiv.org/abs/1412.6980

Mendes-Neves, T., & Mendes-Moreira, J. (2020). Comparing state-of-the-art neural network ensemble methods in soccer predictions. In: Foundations of intelligent systems, (vol. 12117, p 139–149). Springer, Cham. https://doi.org/10.1007/978-3-030-59491-6_13,

Mendes-Neves, T., Mendes-Moreira, J., & Rossetti, R. J. F. (2021). A data-driven simulator for assessing decision-making in soccer. In: Progress in artificial intelligence, (vol. 12981, pp. 687–698). Springer, Cham. https://doi.org/10.1007/978-3-030-86230-5_54,

Merhej, C., Beal, RJ., Matthews, T., et al. (2021). What happened next? Using deep learning to value defensive actions in football event-data. In Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining. ACM, Virtual Event Singapore, (pp. 3394–3403),https://doi.org/10.1145/3447548.3467090,

Pappalardo, L., Cintia, P., Rossi, A., et al. (2019). A public data set of spatio-temporal match events in soccer competitions. *Scientific Data, 6*(1), 236. https://doi.org/10.1038/s41597-019-0247-7

Paszke, A., Gross, S., Massa, F., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In *NIPS'19: Proceedings of the 33rd international conference on neural information processing systems*, https://doi.org/10.5555/3454287.3455008,

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine learning in python. *Machine Learning in Python, 10*(5555/1953048), 2078195.

Radford, A., Narasimhan, K., Salimans, T., et al. (2018). Improving language understanding by generative pre-training. N/A https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

Rudd, S. (2011) A framework for tactical analysis and individual offensive production assessment in soccer using Markov chains. In New England symposium on statistics in sports, http://nessis.org/nessis11/rudd.pdf

Shaw, L., & Gopaladesikan, S. (2020). Routine inspection: A playbook for corner kicks. In: Machine learning and data mining for sports analytics. (vol. 1324, pp. 3–16). Springer, Cham. https://doi.org/10.1007/978-3-030-64912-8_1,

Simpson, I., Beal, RJ., Locke, D., et al. (2022). Seq2Event: Learning the language of soccer using transformer-based match event prediction. In Proceedings of the 28th ACM SIGKDD Conference on knowledge discovery and data mining. ACM, Washington DC USA, (pp. 3898–3908),https://doi.org/10.1145/3534678.3539138,

Singh, & Karun. (2019). Introducing expected threat (xT). https://karun.in/blog/expected-threat.html

Tuyls, K., Omidshafiei, S., Muller, P., et al. (2021). Game plan: What AI can do for football, and what football can do for AI. *Journal of Artificial Intelligence Research, 71*, 41–88. https://doi.org/10.1613/jair.1.12505

Valmeekam, K., Sreedharan, S., Olmo, A., et al. (2022). Large language models still can't plan (A benchmark for LLMs on planning and reasoning about change). arXiv http://arxiv.org/abs/2206.10498v2

Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. arXiv:1706.03762 [cs] http://arxiv.org/abs/1706.03762

Vaswani, A., Ganguly, R., Shah, H., et al. (2020). An autoencoder based approach to simulate sports games. In: machine learning and data mining for sports analytics. (vol. 1324, pp. 40–50). Springer, Cham. https://doi.org/10.1007/978-3-030-64912-8_4,

Wang, Z., Veličković, P., Hennes, D., et al. (2024). TacticAI: An AI assistant for football tactics. *Nature Communications, 15*(1), 1906. https://doi.org/10.1038/s41467-024-45965-x

Yeung, CCK., Sit, & T., Fujii, K. (2023). Transformer-based neural marked spatio temporal point process model for football match events analysis. http://arxiv.org/abs/2302.09276