Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Department of Humanities, Social and Political Sciences
Prof. Dr. U. Brandes

Master Thesis in Applied Mathematics

Fall Semester 2024

Lilian Bonnet

Embedding the Soccer Game States

Supervisors:
Mishra Siddartha
Ulrik Brandes
Hugo Fabrègues

April 5, 2025

Abstract

When modeling football matches as Markov processes, the states of the game among which it transitions are central. This thesis explores the application of a pre-training method leveraging graph neural network capabilities to learn an informative hidden representation of the football game states from tracking data. The game states are first represented in a clearly defined graph format, making use of newly developed, flexible frameworks to represent the positions of the players on the pitch from tracking data (shape graphs, goal-aligned coordinate system). The pre-training task, carefully selected as being attribute masking, also allows to derive new results on the application of various graph neural network architectures to the prediction of some player position based only on its context at the time of the observation of the game. While results could not be drawn on the effectiveness of transfer learning, an important effort was set on laying the foundations towards a more thorough exploration of such a framework.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to **Prof. S. Mishra** and **Prof. U. Brandes** for giving me the opportunity to realize my thesis in such an interesting domain and with such amazing conditions. An additional thank you to Prof. Brandes for his expertise in the topic that helped me shape the topic and direction of this thesis.

I would also like to extend a very special thank you to my great friend **Hugo Fabrègues**, for his valuable time, suggestions, constructive criticism, and insightful expertise, which have significantly contributed to the improvement of my work. He has always been around to support me when I needed and he knew how to give me back faith in my work in times when I struggled.

Another special thank you to my long-time friend and roommate **Léon Noirclerc**, who endured with me these months of the thesis, both through good and bad times. He has also provided invaluable help by just listening to my problems and (always) suggesting some smart solutions to come out of what sometimes looked like quicksand. His advice and discussions have made this process much more manageable and enriching.

Then, a thank you to all my friends, among them **Arthur Serres**, **Feodor Doval** and **Lomán Vezin** (but also many of them that I surely forget), who supported me when I was in an unclear situation and gave me their wise advice.

I am deeply grateful to my family, especially **my mother, my father, and my beloved sisters**, for their endless love, patience, support, and understanding. Their unwavering belief in me has been my greatest motivation throughout this endeavor.

Finally, I would like to acknowledge **ETHZ** for providing the necessary resources and a conducive environment for conducting this thesis, and my Master's degree in general. I would not be the same person without all those years of hard-working study.

This thesis would not have been possible without the support and encouragement of all these individuals, to whom I am forever thankful.

Contents

Notation

In this very short chapter, we include some notation used throughout the thesis, such as vector notation $\mathbf{x}$ vs. scalar notation x, so that the reader can always refer to them if not clear from the context.

Here are some abbreviations used in this thesis:

GNN Graph Neural Network

NN Neural Network

RNN Recurrent Neural Network

NLP Natural Language Processing

MLP Multi-Layer Perceptron

VAEP Value Action by Estimating Probabilities

EPV Expected Possession Value

xT Expected Threat

xG Expected Goal

Throughout this report, the terms association football and soccer will be used interchangeably, though football will be the preferred term, reflecting its common usage in Europe. The term soccer, originally a British abbreviation of association football, remains widely used in North America and some other English-speaking regions. However, as this study is written from a European perspective, football will be the default terminology unless clarity requires otherwise. Soccer was kept in the title only to make a clear distinction from other types of football from the title.

Chapter 1

Introduction

Sports Analytics

> "The management of structured historical data, the application of predictive analytic models that utilize data, and the use of information systems to inform decision-makers and enable them to help their organizations gain competitive advantage in the field of play." B. Alamar, [Ala24].

The practice of sports has been present in human societies for centuries, ranging from recreational activities to structured competitions. Historical evidence of competition in sports can be traced back to ancient civilizations, with the Olympic Games in Ancient Greece being a particularly renowned example. Even though the precise origins of certain sports, such as association football (soccer), remain unclear, their popularity testifies to their important role in contemporary society.

With the advent of industrialization and other societal developments, the increasing availability of leisure time led to a surge in sports participation as a recreational activity, and also to the emergence of professionalization in sport. Together with globalization, this institutionalization facilitated the establishment of codified rules, regulatory organizations, and formalized competitions. Standardization in turn helped expand accessibility and contributed to the emergence of a comprehensive ecosystem surrounding sports, supported by growing popularity. As fan engagement grew, the sports industry became more financially attractive, drawing in greater resources and fostering always more intense competition among athletes and teams.

Among various sports, football has distinguished itself through its rapid expansion and widespread popularity. Its simplicity and minimal equipment requirements are some of the key factors that contribute to its global adoption. The formal codification of the game saw milestones such as the creation of the Football Association (FA) in 1863, the International Football Association Board (IFAB) in 1886, and the Fédération Internationale de Football Association (FIFA) in 1904. The FIFA World Cup, held every four years since 1930, has become one of the most anticipated and widely followed sporting events worldwide, illustrating the sport's profound influence on global culture.

As sports, and especially football, became increasingly competitive and popular, there gradually arose a need for analytical tools to enhance understanding and performance optimization, for both sports organizations and fans. In this context, the recording of data started naturally with the first records dating back centuries. In cricket, scorekeeping already started being recorded in the 18th century ([WES]). In football, rudimentary tracking tables and shot attempt analyses emerged in the early 20th century. From those data records, some statistics were first computed (Henry Chadwick's box scores) and the first notable analytical studies

appeared in the mid-20th century, with Charles Reep pioneering statistical football analysis in the 1940s. Another important milestone in the domain was the creation of the Society for American Baseball Research, more famously known as SABR, in 1971. Indeed it gave birth to the subsequent sabermetrics (empirical analysis of baseball) through the work of Bill James in the 1970s and 1980s. Sabermetrics ultimately led to the famous success story of Billy Beane with the Oakland Athletics.

Refined over the years, modern sports analytics encompasses the systematic study of various aspects of sports organizations to gain a competitive advantage, as defined by Alamar in [Ala24] (cf. initial quote). Many various examples of it could be informing tactical decision-making, supporting player development, optimizing recruitment, aiding injury prevention, but also enhancing in-game strategies and opponent scouting. As competition has intensified, the demand for deeper insights has grown, prompting the development of increasingly sophisticated and comprehensive analytical techniques. Simultaneously, the expansion of data collection has further propelled progress within the field.

In recent years, machine learning has further changed the sports analytics industry, improving the accuracy and efficiency of predictive models. These technological advancements have enabled a more granular understanding of player performance, tactical effectiveness, and game dynamics, having an impact on the strategic landscape of modern sports.

With this in mind, we delve into this thesis in the study of sports analytics. Even though there are many different perspectives in which we can think of applying mathematical modeling to sports analytics, this thesis does not aim or pretend to provide an exhaustive review of all the research in the domain. In a rather simpler way, we seek to explore specific applications through a mathematical and statistical lens. Moreover, the perspective taken here is more that of a researcher rather than a professional practitioner, guiding and explaining the choice of topics covered.

While sports analytics is a broad domain, the primary focus of this work is the analysis of the game itself, i.e. the processes underlying its evolution. This excludes the study of all the surrounding topics discussed before. The objective is to develop novel approaches that effectively capture the intricate mechanisms underlying a match's evolution. Also, football is the main application of interest in the following, although the methodologies discussed apply to other sports.

To model the game's evolution, we need to understand how football is structured. Football, like every sport, is governed by a defined set of rules that dictate game progression. Clubs and teams have historically sought to refine their strategies within these constraints to maximize their chances of victory. In recent years, the increasing availability of data and the advancement of analytical techniques have opened new avenues for strategic refinement. But how to design and develop such methods?

Before constructing a model, it is essential to gather relevant data. As highlighted in Alamar [Ala24], different applications require different types of data. As this thesis focuses on analyzing the evolution of the game itself, it necessitates data that capture its dynamic progression. The study does not seek to develop new methods for data collection but instead builds upon existing datasets. In football, two primary forms of in-game data exist: *event data* and *tracking data*. Event data consist of notation obtained through manual or automated tagging of actions (mainly on-ball) happening during matches, whereas tracking data are made of locations of every player on the pitch and the ball at a high frequency for the full duration of the match, obtained via sensors or optical/video system. Thus, event data are more reliable and complete but don't capture all the information on the pitch and are much more scarce. While event data have been quite extensively utilized, tracking data are a more recent development. Advances in algorithmic processing have facilitated the large-scale collection

and analysis of tracking data. This work centers on leveraging its potential.

For meaningful analysis, a formal mathematical representation of the game is required. Various modeling techniques exist depending on the purpose, ranging from traditional linear models to more complex, data-driven approaches. The progression of statistical analysis and machine learning presents an opportunity to explore newly developed methodologies that effectively capture the complexities of football. Depending on the task of interest and the data at hand, a model should then be chosen.

However, once the data are collected, a fundamental challenge that remains is defining a representation of the game that integrates well into a chosen modeling approach. Two complementary visions exist here: identifying an appropriate model and determining suitable input features, or first constructing a meaningful representation of the game and subsequently selecting a model based on the application's requirements. In this thesis, we try to adopt the latter approach, aiming to develop a comprehensive representation of the game state that could be used in different modeling frameworks. As the question of which features to represent is always unclear from the beginning (especially for non-specialists), we try to bring a first solution to model the game states without manually crafting features.

Ultimately, this report aims to explore representations of the game state that are both comprehensive and computationally efficient. Hence, the general purpose of this work can be summarized as follows: given tracking data at a specific moment in time, can we construct an informative representation of the football game state in a low-dimensional vector space? The goal aligns with the broader vision of developing a "foundation model" for soccer, wherein all relevant game information is encapsulated within a compact vector in $\mathbb{R}^d$, ideally with a minimal value of d.

Guided by this objective, this thesis explores existing work in the field and proposes a novel flexible framework for constructing such an embedding of game states. By capturing the full spatiotemporal structure of the game, this study investigates the application of Graph Neural Networks (GNNs) to soccer analytics, assessing their ability to model the intricate dynamics of gameplay. The overarching goal is to contribute to the development of more expressive, interpretable, and powerful models for understanding and predicting football actions and outcomes.

One example that helped define the scope of this work is the concept of expected possession value (EPV) in football, as introduced in [FBC21]. This serves as a motivating application for the broader objective of constructing generalizable frameworks for analyzing soccer gameplay and is an application that we will keep in mind.

Chapter 2

Background

Before delving into the core aspects of the proposed model, this chapter establishes the necessary context by providing an overview of existing research within the domain, as well as the theoretical frameworks essential for a comprehensive understanding of the subject matter. This chapter serves as a stepping stone, ensuring that readers are well-equipped with the background knowledge required to engage with the subsequent discussions.

In the first section, a thorough detail of similar research is given. By doing so, we aim to bridge the gap between previous contributions and our approach, highlighting the principles that inform our model's development and placing our work in its context.

In the second section, we outline the key theoretical background necessary for understanding the following chapters. This approach enables the rest of the thesis to focus on the intuition and rationale behind the model's selection rather than delving into theoretical specifics. The reader may refer to this section for additional background details as needed.

2.1 Context

2.1.1 The need for a general state's representation

To understand where our work takes its root, we give an overview of some past works in soccer analytics. Thus, we can better understand how they represent the game in their analysis.

Within the sports analytics domain, the development of statistics and data analysis brought several new frameworks and tools to analyze football games. To cite only some examples, Pollard and Reep [PR97] were among the first to propose a framework to quantify what they called the "yield", which represents the probability of a goal being scored minus the probability of one being conceded and to introduce the use of logistic regression to compute the probability of scoring a goal from a shot. This initial *expected goal* model was then expanded in the work of Pollard *et al.* [EPT05] and more recently with the use of tracking data in Lucey *et al.* work [Luc+15]. In the realm of passes analysis, we can cite the work of Bransen *et al.* [BV19], which proposed an approach to model the impact of passes on the scoreline or the QPass methodology from Gyarmati and Stanojevic [GS16] to measure a player's evaluation based on their passes.

Regarding the study of the game's evolution, mathematical frameworks became finer with time: from one of the first works using Markov processes to model the game by Hirotsu and Wright [HW02], through Singh's *expected threat* framework [Sin19], or VAEP (Valuing Actions by Estimating Probabilities) general framework to value player actions by Decroos *et al.* [Dec+19], to the definition of the EPV (Expected Potential Value) using Markov reward and

decision processes in Fernández *et al.* [FBC21] (taking its root from the original basketball's work from Cervone *et al.* [Cer+16]).

With the models' development, there was also a transition from working exclusively with event data (restricted to on-ball actions) like in Decroos *et al.* [Dec+19] to the spread of use of tracking data (including off-ball situations with higher frequency) like in the work of Fernández *et al.* [FBC21] or even more recently in Google Deepmind's research [Wan+24].

A striking similarity between all those studies, ranging from expected goals to passes and expected value analyses, is the significant amount of manual feature extraction that remains necessary for each task. These features are often specifically tailored to the task at hand, with limited transferability across different applications. Given the increasing volume of available data and the recent advancements in machine learning—particularly the empirical success of neural network models—it is reasonable to explore whether a unified framework being applicable to all these approaches could be developed.

The work of Decroos *et al.* [Dec+19] represents a promising initial step in this direction. However, even in this study, a substantial degree of manual feature engineering is performed before constructing their "language" of actions for evaluating player behavior. Therefore, our objective is to advance this abstraction further.

Hence, let's take a step back and try to conceptualize the data and features needed in model development and understand how we can try to generalize that.

Soccer is inherently both a spatial and temporal game. Ideally, we would like to develop a framework that captures the full evolution of the game, which requires an understanding of both its spatial and temporal dynamics. To do this, it is necessary to define what exactly is evolving—namely, the game itself. At any given moment, the game must be mathematically represented to enable its evolution from time t to $t+1$, which introduces the need for a "state" representation of the game. We intentionally use a broad definition of "state" here, as our discussion applies to a wide range of analyses.

Many analyses of football games, whether focusing on passing/shooting models or the study of play phases, encounter the challenge of representing the game's state. To define the game as a stochastic process (e.g., a Markov Reward or Decision Process) that transitions from one state to another with a certain probability, one must first clearly define what constitutes a state. Naturally, this definition varies depending on the specific application, as it is shaped by the underlying objective.

For instance, in a very simple stochastic model, if our objective is simply to predict the outcome of a match, we might model only the evolution of the score, since this is the primary determinant of a match's result. However, for more in-depth analyses—such as tactical evaluation, strategic planning, or opponent assessment—a more detailed understanding of the game's unfolding is required, necessitating a more precise definition of the state.

In other dimensions like in a model designed to predict the probability of scoring, we might take into consideration the striker's position relative to the goal (to assess shot difficulty) along with the goalkeeper's position. In another application, when predicting the results of a competition over a season, one might rely solely on the scores of past matches. However, further refinement may be necessary. For an expected goals (xG) model, factors such as defensive pressure, timing within the match, and the individual skills of the striker become relevant. Similarly, for season predictions, additional attributes like player development over time or external events (e.g., a change in coach) could play significant roles.

These hand-crafted attributes could always be chosen differently and tailored to the specific use case (even selected in an automated way, based on past historical data). The effectiveness of different attributes depends on the model in question. Nevertheless, in each case, the model is provided with a "biased" version of the current general game state, a version that

is determined by the attributes selected by the user. When new analyses are introduced—for example, studying free-kick situations—entirely new features are typically designed based on the analyst's experience and judgment, without necessarily reusing features from previous, unrelated tasks. Or all the sets of previously used features are incorporated again to determine the ones that are useful and coherent.

It is precisely this feature-design process that we aim to examine and ultimately model. Ideally, we seek to create a model that can learn to summarize the information in a game state in such a way that other models can utilize this representation for various analytical tasks. Such a model would encapsulate its understanding of the game in a coherent, transferable format, making it suitable for new forms of analysis, in a similar fashion to how we (as intelligent humans) would think about it and process the information. In the machine learning literature, this is referred to as feature learning or representation learning, where the objective is to automatically discover meaningful patterns in the data that can effectively represent the underlying phenomena.

More concretely, let's focus on an example of interest to better understand this concept. As mentioned earlier, we will examine a general task to demonstrate the usefulness of such an approach. While expected goals (xG) and season prediction models are valuable, they either focus on a specific aspect of the game (goals) or operate at too high a level to provide insights into individual matches (season predictions). Therefore, we need another, more intermediate-level, task that centers on the evolution of the game itself. In this context, using the "potential" value of a game's state appears reasonable.

What is the "potential" value? It is a value between -1 and 1 assigned to each game state, representing the probability of scoring minus the probability of conceding, from the perspective of a team ([Rud11]). With this in mind, we focus on the Markov Reward/Decision Process framework used in [FBC21] to study the Expected Possession Value (EPV). This mathematical framework describes the transition from one state to another, yet it still requires a clear definition of what constitutes a general state. Different interpretations of this definition can lead to varying analyses, as demonstrated in the work of Fabrègues [Fab24], who attempts to simplify the complex state representation used in [FBC21] while maintaining the same learning architecture (at the cost of introducing an infinite state space). This is precisely where the idea of obtaining an "efficient representation" of each state for use in different models arises, though this concept is still to be fully defined.

At the same time, while defining what states are, we must also consider appropriate models for this general task. The architectures used in [FBC21] and [Fab24] (specifically, Convolutional Neural Networks, CNNs, for capturing spatial data arrangements) are effective for learning potential values. However, they might not be suited for other tasks and more recent methods have emerged that focus on learning intermediate state representations that could be transferable, particularly in the domain of "foundation" models, including transformer architectures and graph neural networks (GNNs).

By naturally modeling the interactions and relationships between players as nodes and edges in a graph, GNNs provide a framework that represents soccer not as a collection of isolated actions but as a dynamic system of interconnected agents. Thus, still getting powerful general representation guarantees as transformers, they also allow to represent the game on a finer level.

Usually, such approaches make use of what is called embeddings and representation learning techniques. As its name suggests an embedding aims to find a way of "incorporating" the state into a vector space (namely $\mathbb{R}^d$ for some d). In that way, we seek to represent efficiently (d should be as low as possible) the state. Thus, our task basically reduces to finding a function ψ that maps our set of defined states $\mathbb{S}$ to $\mathbb{R}^d$. An illustration of that is given in

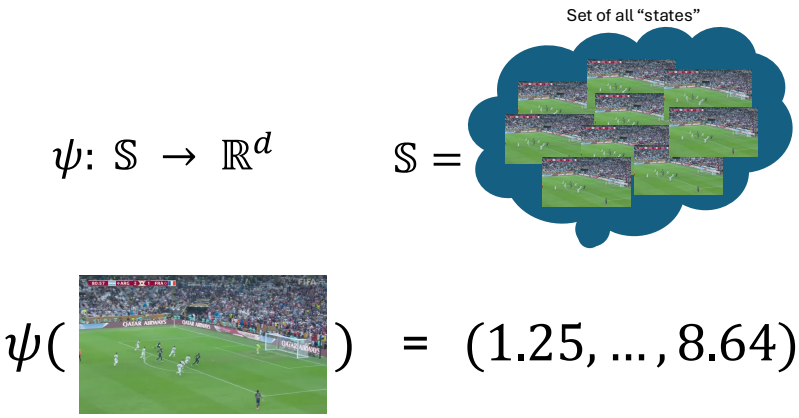

Figure 2.1: What is an embedding of the game state? Pictures of the match were obtained from the freely accessible online video recording of the 2022 FIFA World Cup [FIF22]

Fig. 2.1. Again, the exact definition of a state needs to be precise, since we need to represent this information initially. We deal with this topic in chapter 3 on page 15.

Applications of such an approach are very diverse: the defined states could be used as inputs for other models, but also they could be used for phase detection, or even detecting similar formations on the pitch at a given time. We try to explore some of these applications further in this thesis.

2.1.2 Related work

With the task at hand now more clearly established, we proceed to review prior work employing these models, whose goals were to construct an initial version of the proposed general embedding.

Taking inspiration from NLP techniques Word2Vec and Doc2Vec, Magdaci [Mag21] builds Action2Vec and Player2Vec from events data, thus creating actions and vectors embedding.

Based on the success of the transformer architecture ([Vas+17]) in NLP and other domains, Simpson *et al.* [Sim+22] developed Seq2Event, the first model in football leveraging Recurrent Neural Network (RNN) and transformer-based architectures to predict the next event in a match. They make use of both events and tracking data. Then, Baron [BHS24] completed their work by modeling the inputs slightly differently, thus allowing for turnovers in its action-predicting framework.

For another task, Pei *et al.* [PDC24] used a transformer-based model on tracking data to predict the pass end locations and analyze how the movements of players influence the pass outcome.

Capellera [Cap+24] uses a newly developed encoder-decoder transformer-based architecture, FootBots in the context of motion prediction, outperforming baselines on this task.

In different works, Mendes *et al.* [MMM24b], [MMM24a] and [MMM24c] introduce Large Event Models (LEMs), a framework similar to Large Language Models (LLMs) but for football, the events being the building blocks of football akin to words in language. In the first paper, they define LEMs, whose objective is to learn the next events from a sequence of past events using a deep learning architecture (Multi-Layer Perceptrons, MLPs, and not

transformer-based). Then, in their second paper, they apply their LEMs framework to a specific use case which is the evaluation of player performance and the effects of potential transfers on teams. The last paper is a summary of the LEMs framework, giving a detailed analysis of the framework.

From a pre-training perspective, Rudolph *et al.* [RB24] pre-trained a transformer-based encoder-decoder model on a multiagent trajectories prediction task from tracking data, outlining the efficiency of pre-training.

Adjileye [Adj24] used an approach similar to BERT to pre-train a transformer model to learn match-specific player representations, based on event data.

As our work makes use of graph neural networks, we also explore how this topic has been applied to football. Stöckl *et al.* [Stö+21] are among the first ones to use graph neural networks to analyze defensive play by learning three different models.

Camilo Campos [Jua21] trained similarly to Word2Vec a GNN-based model to learn an embedding of actions, Pass2Vec, from events data.

Then, Xenopoulos and Silva [XS21] designed a model with a representation of the game states as graphs, thus being able to use GNNs (Graph Convolutional Networks) to predict sports outcomes (yards gained in American football as well as esports win probability). They provided a comparison of their model with the xT and VAEP frameworks.

Liu *et al.* [Liu+22] make use of GNNs to improve a reinforcement learning framework to train agents to play football. Even though this is not directly linked to our work, this work shows the great potential for the application of GNNs to the analysis of soccer.

In their work, Hu and Sukthankar [HS22] present an approach to learning agent paths that makes use of the Spatio-Temporal Graph Convolutional Network (ST-GCN). They apply their method to football and show conclusive results.

Anzer *et al.* [Anz+22] trained a semi-supervised GNNs-based architecture to learn classifying overlapping runs.

Luo and Krishnamurthy [LK23] developed a new method using Graph Attention Networks for predicting player performance in sports.

In the general framework of Multiagent Systems (MAS) study, Everett *et al.* [Eve+23] retrieved precisely the players' locations from limited observations using LSTM and GNN-based architecture on events data.

In his thesis, Castellano [Cas23] studied the application of GNNs (a combination of RNNs and Graph Attention Networks to be exact) to detect different events during a match.

Gonçalves *et al.* [GSN24] used GNNs to create a player performance evaluation from event data, comparing it with VAEP and xT frameworks.

Bekkers and Sahasrabudhe [BS24] give a full framework leveraging GNNs (especially Crystal Convolutional Networks) and tracking data to learn the outcome of counterattacks, in gender-specific models.

Wang *et al.* [Wan+24] built an AI assistant for tactics, training a GNN to get player representations and then analyzing corner kick situations, like prediction of the receiver, shot prediction, and tactic recommendation.

Wang *et al.* [Zey+24] proposed a framework to recommend new formations from trained graph neural networks, being able to guide the choice of players on the pitch.

The work from Jiang *et al.* [JCK25] explored different GNN-based architectures (Graph Attention Network and Transformer-based models) to learn the change in xT brought by players to quantify their impact, based on event data.

Recently, Ochin *et al.* [Och+25] used both tracking data and events data to try to predict the position of players, showing strong results to improve automating tracking data collection.

2.1.3 Our Contribution

In this thesis, we focus on trying to learn a hidden representation of the players (nodes of our graphs) to learn an embedding of the game states. To achieve this, we use a pre-training approach using a GNN architecture, trained with tracking data. Thus the pre-training step allows us to also explore the intermediate result we obtain via it. Our contribution thus lies on two different levels:

- predict accurately one player or ball position and velocity using a GNN architecture (during our pre-training task), using only the information from the game state at time t of all other actors,
- creating a reusable embedding of the game states (after the pre-training).

2.2 Theoretical framework

This section provides a concise theoretical overview of the principal concepts and frameworks that underpin the present thesis. By outlining the key theoretical elements, it aims to establish a foundation for the subsequent analysis and discussion, where the reader can always come back.

2.2.1 Machine Learning

Defining what machine learning is, in general, quite a broad task. Consequently, we do not aim to provide a unique or exhaustive definition here. We rather present a selection of definitions that are particularly relevant for establishing the theoretical context of this work.

Machine learning can be broadly described as a subset of artificial intelligence focused on the development of algorithms that enable systems to learn patterns from data and make predictions or decisions without being explicitly programmed for specific tasks. A central aspect of machine learning is the use of data to improve performance on a given task through experience.

Within the broad field of machine learning, there are three main studied paradigms: supervised learning, unsupervised learning, and semi-supervised learning. In supervised learning, the algorithm is trained on a labeled dataset, meaning that each input is associated with a corresponding output or target value. The goal is to learn a mapping function that can accurately predict the output for new, unseen inputs. This approach is fundamental to numerous applications, ranging from image classification to natural language processing and predictive modeling.

In contrast, unsupervised learning operates on datasets that lack explicit labels. Instead of learning a direct mapping from inputs to outputs, the algorithm seeks to identify underlying structures, patterns, or distributions within the data. Common techniques in unsupervised learning include clustering and dimensionality reduction, which facilitate applications such as customer segmentation, anomaly detection, and data compression. By uncovering hidden patterns, unsupervised learning is particularly valuable in exploratory data analysis and situations where labeled data is scarce or unavailable.

Semi-supervised learning bridges the gap between supervised and unsupervised learning by leveraging a small amount of labeled data in conjunction with a larger pool of unlabeled data. This paradigm capitalizes on the idea that unlabeled data, when used effectively, can improve the model's generalization ability and mitigate the reliance on extensive labeled datasets. Semi-supervised learning is widely applied in scenarios where acquiring labeled

data is costly or time-consuming, such as medical diagnosis, speech recognition, and text classification. By combining both labeled and unlabeled data, this approach enhances model performance while reducing the demand for exhaustive human annotation efforts.

2.2.2 Neural Networks

Let's start by recalling what a neural network is and why these models have spread over the past decades in the machine learning domain.

In full generality, a neural network is described by the following formulation:

Definition 2.2.1. Let $L \in \mathbb{N}$ and $N_0, \dots, N_L \in \mathbb{N}$. A neural network ϕ is a map $\phi : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ defined by:

$$\phi = \begin{cases} W_1, & \text{if } L = 1 \\ W_2 \circ \rho \circ W_1, & \text{if } L = 2 \\ W_L \circ \rho \circ W_{L-1} \circ \cdots \circ \rho \circ W_1, & \text{if } L \geq 3 \end{cases}$$

where, for $l \in \{1, \dots, L\}$, $W_l : \mathbb{R}^{N_{l-1}} \to \mathbb{R}^{N_l}, W_l(x) := A_l x + b_l$ are affine maps with $A_l \in \mathbb{R}^{N_l \times N_{l-1}}$ and $b_l \in \mathbb{R}^{N_l}$, and $\rho : \mathbb{R} \to \mathbb{R}$ is the associated activation function that acts component-wise, i.e. $\rho(x_1, \dots, x_n) = (\rho(x_1), \dots, \rho(x_n))$.
We denote by $\mathcal{N}_{d,d'}$ the set of all neural networks with input dimension $N_0 = d$ and output dimension $N_L = d'$.

Remark 2.2.1. In some definitions, the activation function is different in each layer, i.e. there are different ρ_l after each composition. Except stated otherwise, we consider only neural networks with the same activation function in each layer.

Also, in this thesis, we will mainly consider neural networks with the ReLu activation function:

$$\text{ReLu} : \mathbb{R} \to \mathbb{R},\ \text{ReLu}(x) = \max\{0, x\}$$

When confronted with it for the first time, we can wonder why this type of function has been so popular and so successful. One of the reasons is that a lot of universal approximation theorems about them were proven (maybe because they were invented to try to mimic how the brain behaves with neurons). Probably one of the most cited ones is the universal approximation theorem of neural network with infinite width:

Theorem 2.2.1. *Let $C^0(X, \mathbb{R}^m)$ denote the set of continuous functions from a subset X of a Euclidean $\mathbb{R}^n$ space to a Euclidean space $\mathbb{R}^m$. Let $\rho \in C^0(\mathbb{R}, \mathbb{R})$. Note that $(\rho \circ x)_i = \rho(x_i)$, so $\rho \circ x$ denotes ρ acting component-wise.*

Then ρ is not polynomial if and only if for every $n \in \mathbb{N}$, $m \in \mathbb{N}$, compact $K \subseteq \mathbb{R}^n$, $f \in C^0(K, \mathbb{R}^m), \varepsilon > 0$ there exist $k \in \mathbb{N}$, $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $C \in \mathbb{R}^{m \times k}$ such that

$$\sup_{x \in K} \|f(x) - g(x)\| < \varepsilon$$

where $g(x) = C \cdot (\rho \circ (A \cdot x + b))$.

Hence, it states that a one hidden layer neural network ($L = 2$ above) can approximate (for the infinite norm) any continuous function on any compact set up to any threshold, if we choose a non-polynomial, continuous activation function *rho* and if we allow the width (N_1 above) to be of arbitrary size. However, remark that this theorem does not give any empirical guarantee as the width could depend on the function and the maximal error allowed on the set. But with enough computational power, we can imagine that we can probably do a lot.

Thus, with the recent development of computational capacity, neural networks took a central role in machine learning by being powerful approximates.

2.2.3 Graph Neural Networks

With the strong properties of neural networks, the question of whether we could apply this framework to graph-structured data emerged. First, let's start to recall what a graph is.

Definition 2.2.2. A graph (undirected) $G = (V, E)$ is a mathematical object made of a set of nodes V and a set of vertices between those nodes E, where elements of E are of the form (v_i, v_j), for $v_i, v_j \in V$ with $i \neq j$.

Except stated otherwise, in the following, all graphs that we consider are undirected.

Now that we define our building blocks (graphs), let's see how we can adapt the neural network architecture to graph-structured data. As we will extensively use them in the following, we introduce the following what *Graph Neural Networks* are.

Generic Message Passing Neural Network

Graph Neural Networks (GNNs) is a generic name given to any artificial neural network architectures designed to be applied to graph-structured data. Let $G = (V, E)$ be a graph, where each node $v \in V$ (respectively each edge $e \in E$) is associated with a feature vector $\mathbf{x}_v$ (resp. $\mathbf{x}_e$). In the most general form of graph neural network, the graph itself can also have a set of attributes in a feature vector $\mathbf{x}_G$. Given this graph as an input, a GNN iteratively updates the graph representations by aggregating information that takes advantage of the graph structure (using relations between each object of the graph) and keeps graph symmetries (permutation invariance). This is done by iteratively passing all the information to one/different artificial neural networks, aggregating information based on the graph structure. Give a more detailed intuitive explanation of GNNs.

A GNN is composed of several layers of Message Passing Neural Networks (MPNNs). Formally, an MPNN acting on the nodes features is as follows: at layer $k+1$, the hidden representation of a node v is computed as:

$$\mathbf{h}_v^{(k+1)} = \phi^{(k)} \left(\mathbf{h}_v^{(k)}, \bigoplus_{u \in N_v} \psi(\mathbf{h}_v^{(k)}, \mathbf{h}_u^{(k)}, \mathbf{h}_{e_{uv}}^{(k)}) \right),$$

Where:

- $\mathbf{h}_v^{(0)} = \mathbf{x}_v$, for all $v \in V$,
- ψ is a differentiable function that computes a value to update the information of node v based on a given neighbor u,
- $\bigoplus$ is a permutation invariant aggregation function (e.g., sum, mean, max, or attention-weighted sum) that collects information from all neighbors $\mathcal{N}(v)$,
- $\phi^{(k)}$ is an update function, differentiable (typically a neural network) that processes the aggregated information along with the current representation of v.

The same kind of MPNN can also be applied to update the edge and the graph representation vectors.

Then, after applying some layers, depending on the task of interest, the output is computed (whether a final aggregation should be applied or not, depends on the task). Hence, it is a quite general architecture. It could be used for different tasks on graph-structured data, mainly:

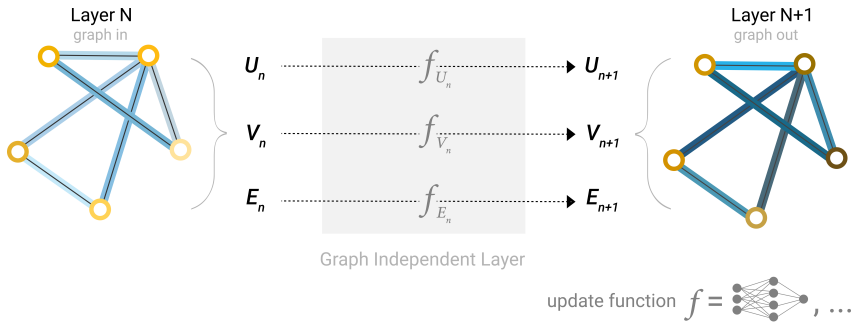

Figure 2.2: The most simple MPNN, figure taken from [San+21]

- graph-level task,
- node-level task,
- edge-level task.

Our focus here is graph-level and node-level tasks.

This is the most general definition of it.

To get a better idea, let's start with its simplest form. Each feature vector has its update function (neural network) that does not take into account the graph structure (no aggregation). This would be represented as in Fig. 2.2.

Then, we can start changing the parameters in the MPNN to improve it.

Graph Convolutional Neural Networks

The first kind of MPNN we study is the Graph Convolutional Networks (GCNs), where aggregation is performed via normalized summation.

$$\mathbf{h}_v^{(k+1)} = \phi^{(k)}\left(\mathbf{h}_v^{(k)}, \sum_{u \in \mathcal{N}(v)} \frac{e_{u,v}}{\sqrt{\hat{d}_u \hat{d}_v}} \mathbf{h}_v^{(k)}\right)$$

where:

- $\hat{d}_u = 1 + \deg(u)$, $\deg(u)$ being the degree of node u,
- $e_{u,v}$ is the weight associated with the edge (u, v), only allowed to be one-dimensional.
- $\phi^{(k)}$ is a simple affine function, followed by an activation function acting component-wise (the implementation in PyTorch doesn't include the application of the activation function).

It is noted that an over-smoothing problem was observed when piling up too many layers of GCN one after the other. Basically, the information is too much propagated through the graph and every node happens to converge towards a similar representation, losing the initial purpose of the GNNs. Hence, when using GCN (or similar architectures that aggregate information from its neighbors without allowing adaptation of the aggregation coefficients between layers), we should be careful to not blindly assemble the layers and stick to a low number. This is what drives our choice of hyper-parameter tuning as well (for the number of layers).

Other MPNNs

The other GNN architectures used in this thesis are very similar to GCN ones, hence we don't detail them all here and let the reader do the task of checking them.

2.2.4 Representation learning

General definition

The general idea behind representation learning is to define methods to automatically find a suitable representation of the data of interest for some defined tasks, avoiding the need for manual feature selection. It could seem quite general but it has shown quite interesting results in the domain of machine learning (NLP with BERT, image processing with CLIP or DiNo). Usually, this is done through the means of an embedding. An embedding is a mapping, projecting ("embedding") the data of interest into a mathematical representation of this data (usually $\mathbb{R}^d$ for some given d) that preserves the "structure" of the initial data. What is meant by structure is domain and task-dependent, thus the general definition. That's the reason why the task of embedding some given data needs a careful definition of the goal and methodology to achieve this structure preservation.

Application to graphs

Representation learning is especially of interest when studying graph-structured data. In fact, to be able to use common machine learning models, data should be first turned into a format that can be fed to those algorithms. Concretely, most of them take as inputs real vectors, i.e. one needs to embed this high-dimensional data into a feature vector that could later be used by other models. A careful review of the different methods used in graph representation learning can be found in the work Hamilton *et al.* [HYL17] and [Ham20]. We redirect the interesting reader to those documents if further information is needed. One interesting fact to be noted is the broad scope of methods that exist beyond the use of GNNs to learn a graph representation. This could lead to many comparisons to our work.

In the following, we will only focus on the framework of representation learning using GNNs.

2.2.5 Pre-training and transfer learning

Lately, pre-training has shown promising results in some tasks, such as NLP. Notably, we can cite the work of BERT ([Dev+19]) in NLP that paved the way for today's advances in Generative Pre-Training (GPT) models.

The idea of the pre-training is to train a model on a self-supervised task to apply representation learning to the data. But then, instead of mapping statically every object of the dataset to some vectors (static embedding like Word2Vec), the full model used on the self-supervised task can be reused (and take more context than just individual objects). Thus it can be fine-tuned on other downstream tasks, starting from the state of the pre-training, so already being aware of the data it handles in a sense. By doing that, the hope is that the model will learn some useful representation of the data during the pre-training task, that could be useful for other downstream tasks. And it is successful in diverse domains. Let's take the example of BERT and describe it to get a clearer picture of the subject.

The core architecture of BERT is based on the Transformer encoder, which utilizes self-attention mechanisms to model contextual relationships between words in a sentence. BERT is pre-trained on large corpora using two unsupervised learning objectives:

- Masked Language Modeling (MLM): During pre-training, a percentage of the input tokens are randomly masked, and the model is trained to predict the original value of these masked tokens based on their context.
- Next Sentence Prediction (NSP): This objective trains the model to understand the relationship between sentence pairs by predicting whether a given sentence B logically follows another in the training data.

After pre-training, BERT can be fine-tuned with additional output layers for specific downstream tasks such as question answering, sentiment analysis, or named entity recognition. Fine-tuning typically involves only a few epochs and minimal architecture modifications, as BERT already encodes rich language representations.

Another important point to note is that after BERT was presented, RoBERTa ([Liu+19]) presented another framework to show that the NSP task could be removed from the pre-training without impacting the results on the downstream tasks, thus simplifying the learning task.

This is the framework leveraged by Adjileye in [Adj24] to learn a foundational model for football from events data. However, to make such an application of the BERT framework, Adjileye needs to define clearly how to represent a football game in a similar way as language.

Pre-training for GNNs

While pre-training showed various successes, it has been tested in the framework of graph representation learning and at first was deceptive. As shown by Veličković *et al.* [Vel+18], GNN's architecture has proven to capture already some information without the need for pre-training. However, this has evolved with the work of Hu *et al.* [Hu+20], which brings new perspectives on how to build new strategies for pre-training on graph data. Indeed, Hu *et al.* showed on many different tasks that with careful design of both node-level followed by graph-level tasks, significant transfer learning could be achieved. This paved the way for many other works on the topic of pre-training graph neural networks.

Chapter 3

Model

In light of the recent advances exposed in the last chapter, the question remains open whether the methods discussed have their application in football, especially in our framework of interest, that is to represent the state of the game. In this chapter, we try to open a path in this direction by proposing a method to embed the states of the game (where the state is obtained through tracking data). In this chapter, we set up the necessary theoretical framework around the model used by justifying the way we built it.

3.1 Finding a representation

Since football is a collective game, there have been many studies that used networks to represent the in-game dynamics of this sport. The flexible and expressive framework given by modeling the interactions as a graph makes this choice suited for our objective. Indeed, while the recent successes and popularity of NLP modeling led to many attempts to represent the game states as words and apply NLP models to them, there were far fewer attempts to use graph neural networks in a pre-training objective. This is despite the very broad possibilities they offer (transformers can be viewed as some specific kinds of GNNs [Cha20]).

In this context, it becomes natural to start wondering if GNNs could help us achieve this representation of the game (embedding). In the following sections, we thus present how the game states (represented as tracking data) could be modeled as graphs and then which model we try to apply to those graphs to obtain an embedding that could be useful. In Fig. 3.1, we summarize graphically the general idea behind our process and how it is used in practice.

3.1.1 Construction of the graphs

Firstly, since we would like to make use of a model that takes graphs as inputs, we need to decide on how to represent our frames of tracking data (they represent the available information for us, i.e. the game states) into graph-structured data. In the tracking data, we have access to the positions and velocities of all players and the ball, and the teams. In all the following, the players and the ball represent the nodes of our graph.

However, as they are quite an important part of any MPNN, we need to take care of how edges are drawn between this set of nodes to define our initial input graphs. Different approaches could be taken, as we can see in [BS24] and [XS21], where different graph designs are used.

We choose here to represent our data following two different designs, which we can compare in the following. In both abstractions, the players and the ball represent the nodes, and

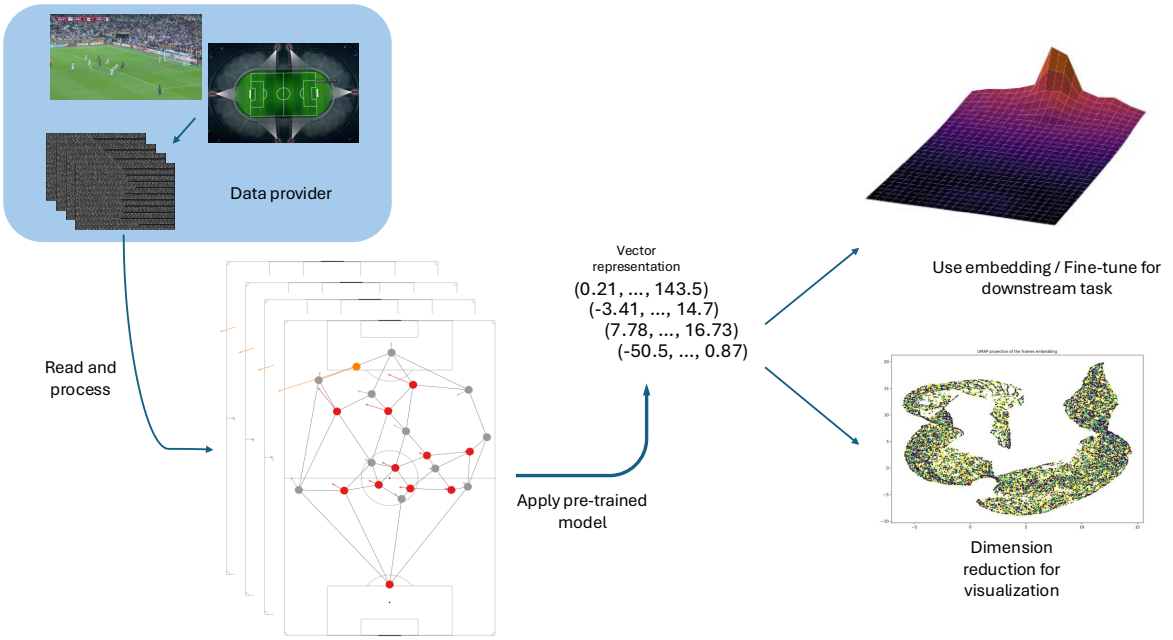

Figure 3.1: How to use the pre-trained model in practice? The top right figure is an xT plot, taken from [Sin19].

the edges of the graph are drawn differently. We only make use of undirected graphs. Here are the two approaches:

1. fully connected graph,

2. shape graphs.

In the fully connected graph approach, we decide to simply draw edges between every node, i.e. every player is linked to every other player and the ball.

In the second approach, we decide to use special graphs that are called *shape graphs*. In nature, shape graphs are (stable) refinements of Delaunay triangulations (we remind the reader that Voronoi tessellations, the duals of Delaunay triangulations, can be used to simply represent a form of control of the pitch). Hence, they represent some kind of interaction between players and we can already encode some spatial information in the edges themselves (whether you are linked to a player or not depends on the context around you). We expect that the model can extract meaningful insights from it. For an exact construction of the graphs and empirical exploration of it, we refer the interested reader to [Sch24].

To get a clearer idea of what a representation of the state looks like when we build the graph in this fashion, we illustrate a shape graph built from a frame in Fig. 3.2.

We construct all the shape graphs by ignoring the ball. Then, we add edges from every player to the ball. This encodes some basic spatial information about the relative arrangements of players while still being able to take into account the ball whatever the position of a player on the pitch is. Indeed, as the ball is governing the evolution of the game, we expect it to be (and should be?) the focal attention of every player, whatever their positions and whatever the state of the game.

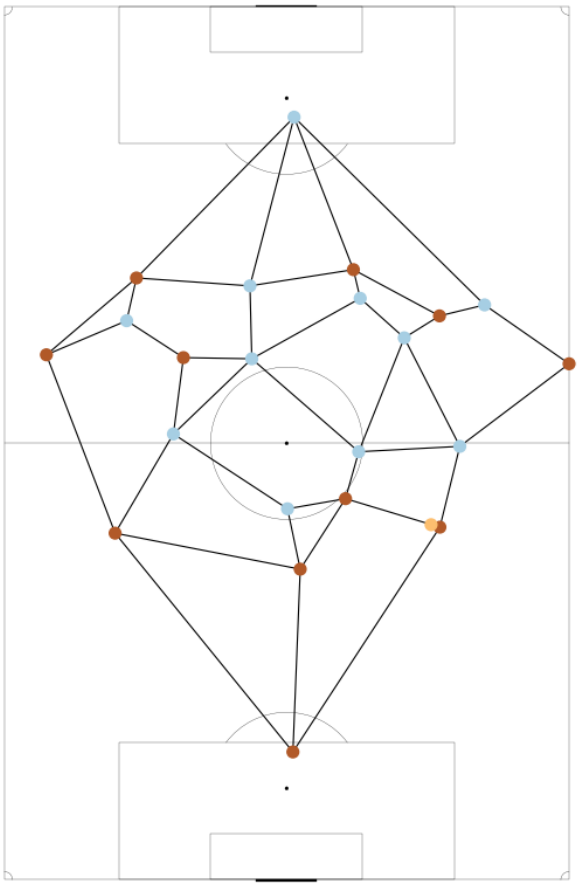

Figure 3.2: An example of a shape graph constructed from a tracking data frame. The ball is represented in yellow and is not included as a node of the shape graph.

3.1.2 Features

Then, to be able to fully take advantage of the framework offered by GNNs, we define the initial features to represent the nodes. Having in mind that we want to build an embedding that captures some information about the game but still stays as general as possible, we would like to keep general features as well. Indeed, we recall from our goal that the idea is to avoid the crafting of specific features but let the model choose what information is important to extract.

Hence, in the final version of our graphs, we decided to attach the following features to each node:

- $(x, y) \in \mathbb{R}^2$, coordinates of the player/ball on the pitch, regarded in the goal-aligned coordinate system, i.e. $-34 \leq x \leq 34$ and $0 \leq y \leq 105$. The position is used directly from TRACAB data.

- $(v_x, v_y) \in \mathbb{R}^2$, velocities of the player/ball along each axis, derived from TRACAB data. Velocities are computed from positions using the symmetric quotient formula (see equation (3.1)) and smoothed using a Savitzky–Golay filter.

- $s = \sqrt{v_x^2 + v_y^2} \in \mathbb{R}_+$, speed. This speed is not computed from the velocities along each axis but is already given in TRACAB data. Hence we don't know exactly by which means it is computed or smoothed. We include it since TRACAB data are reliable and this allows us to have a more trustworthy process in case there are some important erroneous data points in the positions.

- $t \in \{-1, 0, 1\}$, the team categorical variable, being -1 if the node represents a player in the away team, 0 if the node represents the ball, and 1 if the node represents a player in the home team.

- $b \in \{0, 1\}$, a categorical variable indicating if the node is the ball or not. We duplicate the information of the ball category (already contained in the last feature)

- $o \in \{-1, 1\}$, a categorical variable representing the team in possession of the ball in the current state of the game: -1 for the away team owning the ball, 1 for the home team owning the ball. This information is given in TRACAB data.

For a sequence of values $z_i \in \mathbb{R}$, $i \in \{1, \ldots, n\}$ representing records of a real value of the interest at regular interval δ in time, the symmetric quotient formula used to compute velocities is given as follows:

$$\begin{cases} \frac{z_i - z_{i+1}}{\delta}, & \text{if } i = 1 \\ \frac{z_{i-1} - z_{i+1}}{2\delta}, & \text{if } i \in \{2, \ldots, n-1\} \\ \frac{z_{i-1} - z_i}{\delta}, & \text{if } i = n \end{cases} \tag{3.1}$$

Regarding the edge features, we set them as follows (when the model allows multi-dimensional edge features, otherwise, we just set the weight to 1, as when it's not multi-dimensional, it's only allowed to be non-negative): (e_1, e_2) with $e_1 = 1$ and $e_2 = 0$ if both nodes are in the same team, $e_1 = -1$ and $e_2 = 0$ if both nodes are in opposing teams and $e_1 = 0$ and $e_2 = 1$ if a node is a player and the other node is the ball.

3.1.3 Geometric considerations

Guided by geometric considerations and the need for a symmetric framework regarding the length of the pitch (i.e. invariant if a team changes sides), we applied the following:

- The pitch was flipped depending on the masked player team to always indicate to the model the direction of play, i.e. when masking a player, we always made sure that its team is playing from bottom to top (vertical pitch according to [Bra24]) by rotating all the coordinates of the players and the ball if needed.
- We didn't implement anything regarding symmetry along the length of the pitch (vertical symmetry in the goal-aligned coordinate system) since we would like to keep the separation between the left and right sides. We leave this consideration to the user of the embedding.

This allows us to interpret the obtained embedded frames as being the representations of the game states from the point of view of the team playing from bottom to top (in the goal coordinate system, since we fix an absolute coordinate system of the pitch that doesn't change).

3.2 Embedding/Pre-Training through GNNs

Let's say that we defined a task of interest that we would like to process (for instance, predicting a player's position, or predicting the ball location). We would like to define a model that takes an input graph G and returns an output that corresponds to the defined task. In our case, the task of interest would be to predict the expected possession/potential value of a situation knowing the information of the game at time t, represented in a graph. The exact representation of the states is still to be defined.

We define here a general procedure/model to tackle the task of interest. We will then precise how we use this general framework in our specific case. Following the general pre-training idea with attribute masking, we define a model with 2 modules:

- Embedding and Encoder: this module first transforms the features of the graph $G^{in} = (V, E, X_v, X_e)$, through some Graph Neural Network (GNN) layer, into a hidden representation $G^{out} = (V, E, H_v, H_e)$, where (V, E) has not been changed. Indeed, this is a message-passing layer, i.e. the backbone of the graph does not change and only the features are updated. This first step serves to aggregate the information about the input graph. Then, through another GNN layer applied to the hidden representation, this module encodes the information for each node into an encoded vector $f^i \in \mathbb{R}^d$, for $i \in \{1, \dots, |V|\}$. In this fashion, the input graph features are embedded into $|V|$ vectors f^i.
- Task module: this module converts the output from the Embedding & Encoder module into a final output, depending on the task of interest.

Now that the general model's architecture is defined, let us focus on our task of interest. Let $G_t = (V_t, E_t)$ be a graph (undirected) representing all the players on the field at time t. Each vertex in $V_t = \{p_t^1, \dots, p_t^n\}$ represents a player with some corresponding features $p_t^i = (p_{t,1}^i, \dots, p_{t,d}^i)^T \in \mathbb{R}^d$ and each edge in $E_t = \{e_t^1, \dots, e_t^m\}$ represents some relations (to be defined) between the corresponding players, with some features attached $f_{e_t^i} = (f_{t,1}^i, \dots, f_{t,r}^i)^T \in \mathbb{R}^r$. We stay as general as possible by not precisely defining the nodes and edges features for

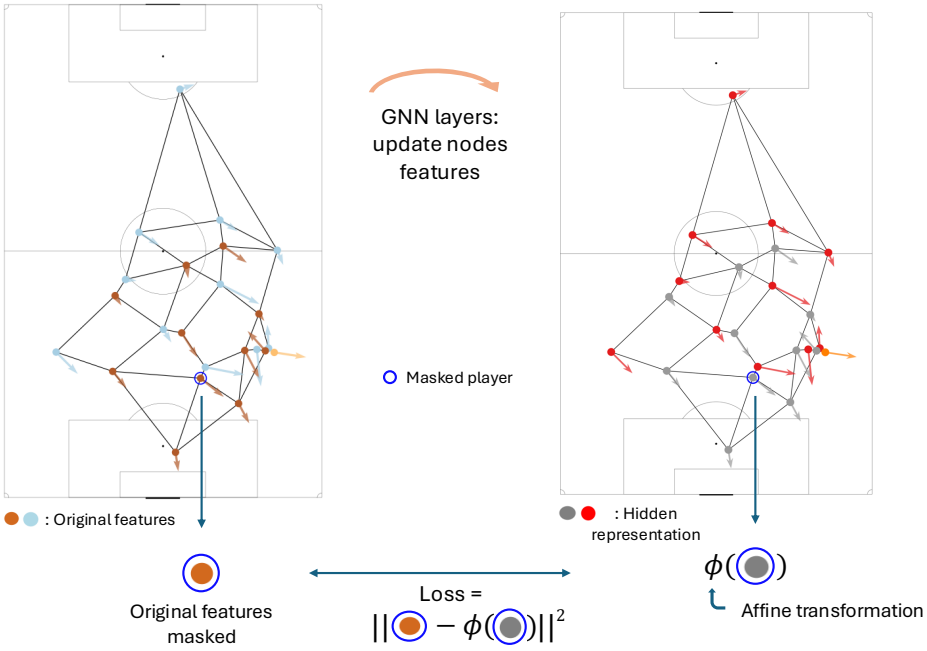

Figure 3.3: How is pre-training working?

our procedure to be applicable with different inputs.
Let us also define for each $i \in \{1, \dots, n\}$, $G_t^{-i} = (V_t^{-i}, E_t$ as being the graph with i-th player masked (mask features associated with this player but keep it connected to the network). Those graphs will be the inputs of the model defined above.

Taking inspiration from the pre-training idea (with attribute masking, as defined in [Hu+20]), we use the previously defined architecture and we pre-train our model before using it for any task.
For the pre-training, we use G_t^{-i}, for one random $i \in \{1, \dots, n\}$, as input training graphs (for all t), on the task of predicting the features of the masked player node. The task head used for pre-training is an affine layer that transforms the players embedding $(\tilde{p}^1, \dots, \tilde{p}^{n-1})$ (obtained after the first module) into a vector of attributes $\hat{p}_t^i = \phi(\tilde{p}^1, \dots, \tilde{p}^{n-1})$, that is to be compared with p_t^i for pre-training.
Then, for other downstream tasks, the task head is left out and replaced with another task head, suited for the task of interest, that is fine-tuned or kept frozen. This reduces (resp. removes) the overhead of training again the Embedding & Encoder module, which keeps similar parameters. Hence, for predicting the expected possession/potential value, the task head will be a fully connected neural network that acts on the embedding of the player and outputs a single real value.

In particular, if the downstream task is related to the full graph, i.e. we need a representation for the full graph, we apply an aggregation function on all nodes features representation before doing the task of interest. For instance, let's say we would like to do a task of classification (like the phase classification of chapter 4 on page 24), then we aggregate the information of all nodes (by a global mean pooling) to obtain a hidden representation of the entire graph (state).

We illustrate the pre-training task in our context in Fig. 3.3.

It is particularly interesting to study the performance of the mask player prediction task,

but more importantly, this approach provides a framework for reusing the first part of the architecture by training only a new model on its output, rather than retraining the entire architecture. Ideally, this should be compared against a model trained on the full architecture to evaluate performance differences (a real downstream task of interest).

We compare the following GNNs architectures in the following:

1. GINEConv (with fully connected graph)
2. GINEConv (with shape graph)
3. GCN (with 2 different shape graphs, one for each team)
4. GCN (with shape graph)
5. SAGEConv (with shape graph)
6. EdgeConv (with fully connected graph)
7. WLContinuous (with shape graph)

We justify quickly why we kept those architectures among all the architectures possible:

1. We would like to compare the fully connected graph against the shape graph to see if there is any difference.
2. We selected the GINE layer based on the work of Hu *et al.* [Hu+20],
3. GCN is the initial layer we trained, and we initially started with two different shape graphs, so this is one of the baselines we kept since the beginning.
4. As stated GCN is the most popular MPNN and proves to be efficient in many tasks.
5. SAGE was invented to overcome the problem of over-smoothing of the GCN layer and is known to be efficient for graph-level tasks (which is the final task with the embedding).
6. EdgeConv is a layer that was created to adapt its weights in its aggregation step, depending on the information received (that's the reason why we use it with the fully connected graph).
7. WLContinuous is a simple replication of the WL algorithm, we need to be careful as this layer does not include any activation function so it will just aggregate the information of its neighbors for 2 layers. It serves as a baseline to see how a simple aggregation performs on the task evaluated.

3.3 Practical set up

3.3.1 Data

We use tracking data from the 64 matches of the 2022 FIFA World Cup in Qatar, which has kindly been provided by FIFA. The data provider is TRACAB (optical tracking data). The complete dataset comprises 64 matches, each containing approximately 80,000 frames captured at a frequency of 25 Hz. To ensure consistency and relevance of the data, set pieces and break periods—identified by a tag in supplied data from TRACAB—were excluded. Additionally, the z-coordinate of the ball was disregarded due to concerns about its reliability.

Only the first and second halves of each match were retained to maintain comparability across matches.

All frames containing at least one missing value were removed from the dataset. Given the overall high quality of the data, the number of such frames was limited, and the remaining dataset was sufficiently large for training purposes.

Player and ball positions are stored according to a goal-aligned coordinate system that remains fixed throughout the game, ensuring consistency even after half-time when teams switch sides.

To process player positions and the corresponding derived velocities, a newly developed library based on the method proposed in [Bra24] was employed. This library enables the translation of all frames to a unified pitch, facilitating a standardized representation of player locations across all matches. Even for this specific study, it was not useful as every match happened on a pitch at the standard IFAB size dimensions.

Outliers in velocities (after applying the filter) were removed by excluding velocity values exceeding (in absolute value) 12 m/s for players and 75 m/s for the ball.

In order to manage computational load and allow for timely model training, the dataset was downsampled by a factor of 100 in the comparison of the 7 models below.

The dataset was divided into training, validation, and test sets with a split of 70%, 20%, and 10% respectively. Masked features within the dataset were assigned a placeholder value of -100, although this value was not specifically fine-tuned.

Lastly, the dataset includes information about the team in possession for each frame, as provided by TRACAB.

We need to mask a random node (a player or the ball) for each frame. We do this by randomly selecting one object for each frame when we build the dataset from tracking data.

3.3.2 Optimization

MSE loss was used and was computed between the prediction and the target value (mask player) during the pre-training task, as represented in Fig. 3.3.

For the optimizer, Adam optimizer was used with a learning rate of 1×10^{-2}, betas of 0.9 and 0.999 (default values), and a weight decay of 0 (default value).

Results reported are the loss, the average batch distance for positions (between the predicted and the target) and the average norm of speed difference (between the predicted and targeted speeds along each axis, the value of the norm of the speed given by TRACAB is not taken into account in this computation).

3.3.3 Hyperparameters

Hyperparameter tuning for the models was conducted using the validation dataset. The design of the Graph Neural Network (GNN) was guided by the framework proposed in [YYL20].

The hyperparameter search space included the following configurations:

- Batch normalization: enabled or disabled
- Aggregation function: global mean pooling or global max pooling
- Number of GNN layers: 2 or 4
- Number of hidden channels in convolutional layers: 32 or 64

- Number of hidden channels in neural network layers: 5, 10, or 15
- Number of training epochs: 50, 75, or 100

Initial experiments included both input normalization and batch normalization; however, these configurations resulted in poorer performance. Hence, the final models were trained without any form of normalization.

Finally for all models evaluated the hyperparameter set was:

- Batch normalization: disabled
- Aggregation function: global mean pooling
- Number of GNN layers: 2
- Number of hidden channels in convolutional layers: 32
- Number of hidden channels in neural network layers: 10
- Number of training epochs: 75

and the final size of the embedding was set to 10 in all simulations compared.

3.3.4 Implementation details

For the implementation and training of the GNNs, we used PyTorch Geometric (a library built on PyTorch). The code should be asked of the author if needed.

UMAP (Uniform Manifold Approximation and Projection) was used to project the embedding space (of dimension 10) into a 2-dimensional one. It is a dimensionality reduction technique that preserves both global and local data structure by modeling the data as a manifold and optimizing a low-dimensional representation of it. It is widely used for visualization and preprocessing of high-dimensional data due to its ability to maintain meaningful relationships between data points. That's the reason why we chose it compared to PCA.

For the other tasks of interest, the basic (default value) implementations of scikit learn LogisticRegression, LinearRegression, KMeans, and NearestNeighbors were used.

Chapter 4

Results

4.1 Pre-training task

Firstly, we report the results obtained on the pre-training task, as they, themselves, constitute some findings (since this task was not carefully studied before). In table 4.1 and table 4.2 on the following page, we report losses and distances obtained on both the training and the validation sets. Values are in meters for the distance and in m/s for the speed.

We recall here the model and input configuration:

1. GINEConv (with fully connected graph)
2. GINEConv (with shape graph)
3. GCN (with 2 different shape graphs, one for each team)
4. GCN (with shape graph)
5. SAGEConv (with shape graph)
6. EdgeConv (with fully connected graph)
7. WLContinuous (with shape graph)

The baseline that we consider for the distance in this task is the performance of the center of mass (of the team of the player masked). The center of mass predicts the players' positions with an average distance of 18.01 m (over all the dataset).

	MSE (Loss)		
Model/Input config	Train	Val	Test
1	64.13	61.48	62.76
2	19.25	17.77	**18.76**
3	42.80	41.78	41.43
4	30.13	30.50	31.76
5	19.40	19.27	20.27
6	65.49	62.09	63.18
7	47.47	48.02	50.03

Table 4.1: Comparison of model performance metrics.

Model/Input config	Distance (m)			Speed (m/s)		
	Train	Val	Test	Train	Val	Test
1	15.04	17.80	14.81	1.85	1.80	**1.85**
2	6.39	6.18	**6.33**	2.34	2.29	2.29
3	11.16	10.88	10.85	2.01	1.88	1.85
4	8.73	8.84	8.85	1.96	1.92	1.96
5	6.62	6.56	6.57	1.93	1.96	1.99
6	14.97	14.71	14.67	2.58	2.52	2.57
7	10.70	10.67	10.71	2.06	1.93	1.98

Table 4.2: Comparison of model performance metrics.

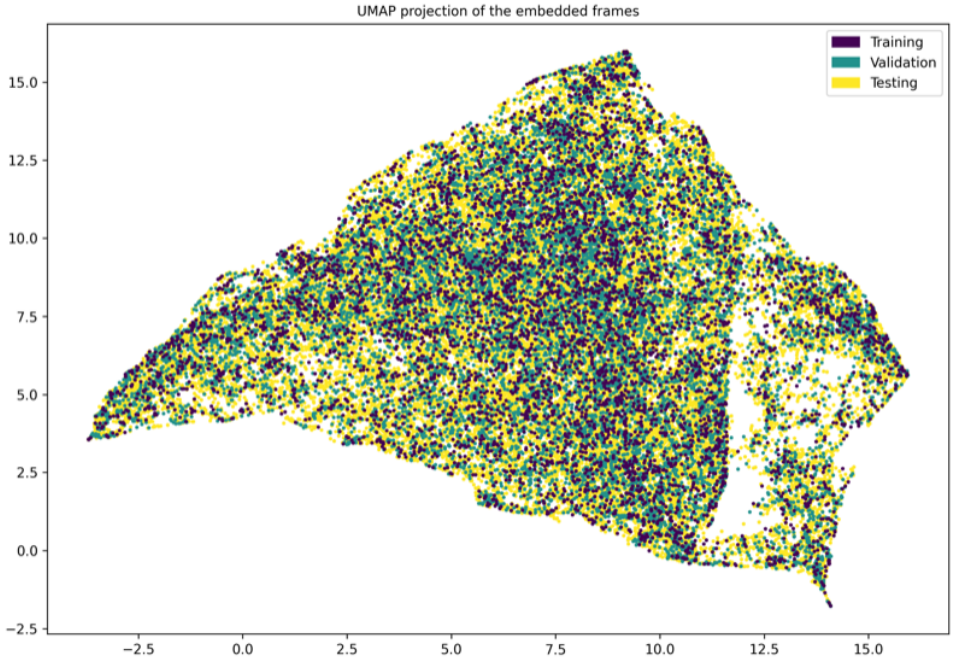

Figure 4.1: UMAP projection of the embedding—labels train-val-test

4.2 Embedding assessment

In a second time, we try to assess the obtained embedding in different ways. After the pre-training task, we drop the final linear layer (that serves to predict the masked attributes) and we apply the model to each graph in the training, validation, and test sets, to assess the obtained vectors (representing the associated frames). We show in this section only the figures for the model we selected (GINE with shape graphs). However, we put all other figures and results in the Appendix.

We use in the plots all the points we computed the embedding values for, i.e. for training, validation, and testing graphs (note that they are not masked here, so not seen anyway by the training). We justify our choice by showing in Fig. 4.1 that all the different points (training, validation, testing) are spread equally in the embedding space.

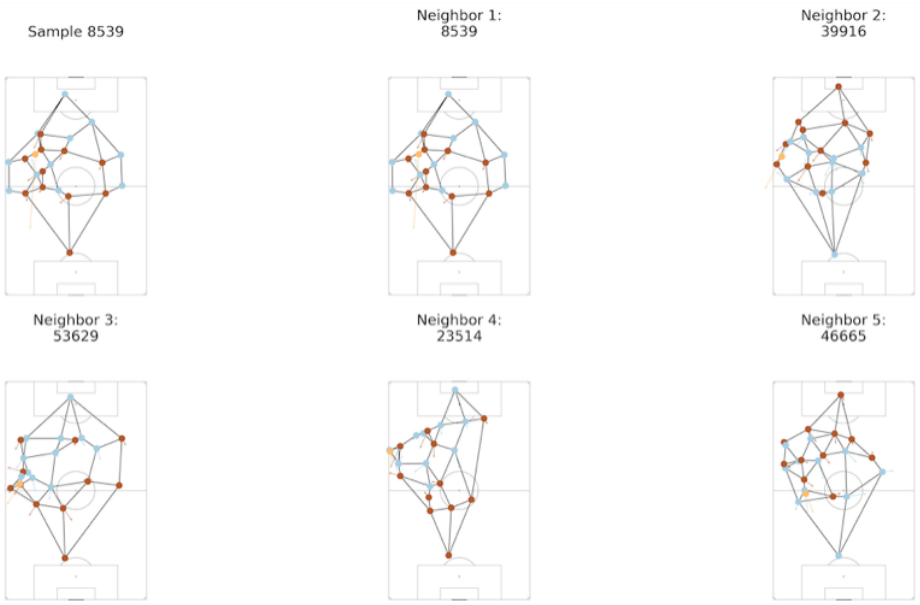

Figure 4.2: First sample and its nearest neighbors in the embedding

K-Nearest Neighbors analysis

Firstly, we can have an insight into the embedding's quality by looking at the 5-nearest neighbors of 2 random given frames in the testing set (to be sure it's completely unknown to the model). We show the 2 samples together with their neighbors in Fig. 4.2 and Fig. 4.3 (ignore the first neighbor in the plot as they are always the same frame).

We observe in Fig. 4.2 a coherent similar structure between all the neighbors. For instance, observe the ball position or even the goalkeeper's positioning. But also, we can see that the shapes of the graphs are close: the goalkeepers are only linked with 3 defenders on all frames except one.

In Fig. 4.3, we also observe a very similar structure with the bottom goalkeepers. Also, notice that our geometric consideration correctly clustered together frames even if different teams are playing from bottom to top. Thus, we can interpret the vector as being the point of view dependent, from the point of view of the team playing from bottom to top.

Phases retrieval

We have a deeper look at 3 different phases of the game, based on the position of the focal team concerning the vertical coordinate (y-coordinate). The phases of play are created as follows: the focal team is said to be in the last third of the pitch if the median y-coordinate is in the last third of the pitch (top-third with our view), respectively in the middle-third if the median y-coordinate is in the mid-third of the pitch, respectively in the first-third if the median y-coordinate is in the first third of the pitch. We annotate each frame depending on which phase the game is at this specific moment (the point of view is from the focal team).

To get a sense of whether the embedding can differentiate between them, we plot them on the same figures and see if the model "clustered" (or separated) them in an observable way. This would be a sign that the embedding contains this information and that we can retrieve it easily from it. We correctly observe the clusters in Fig. 4.4 and in Fig. 4.5.

We further try to understand if we can differentiate those defined clusters by getting some

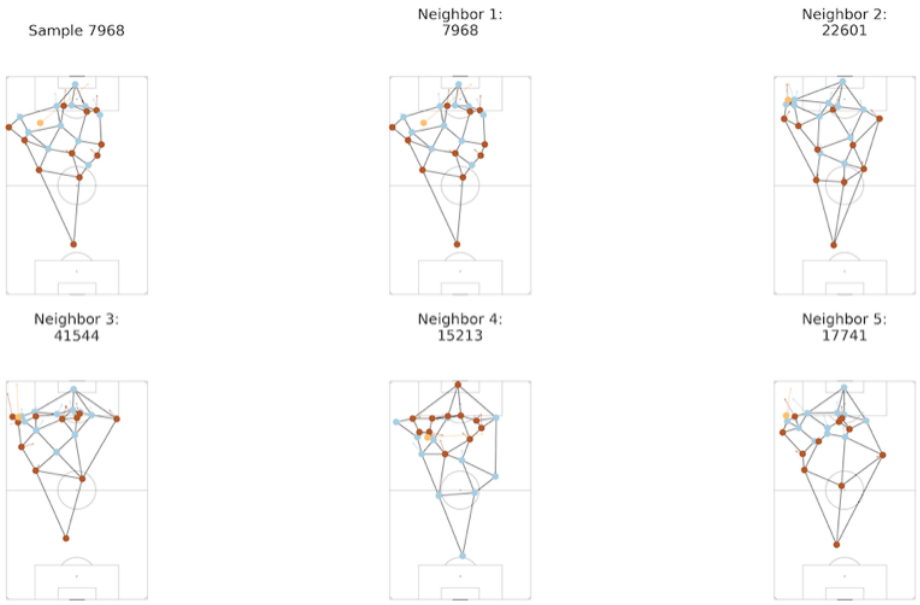

Figure 4.3: Second sample and its nearest neighbors in the embedding

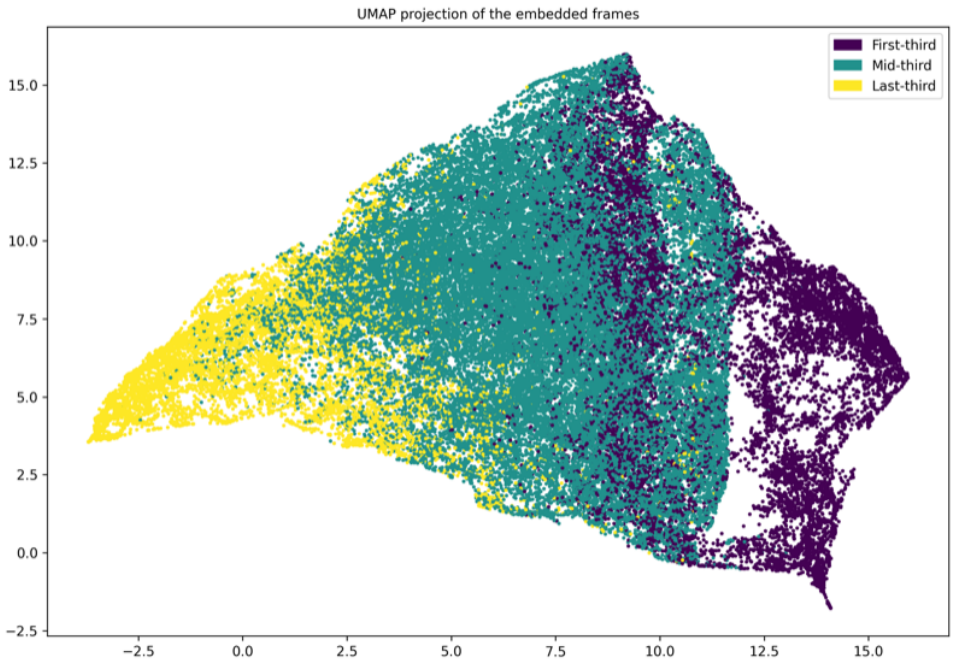

Figure 4.4: UMAP projection of the embedding—labels phases

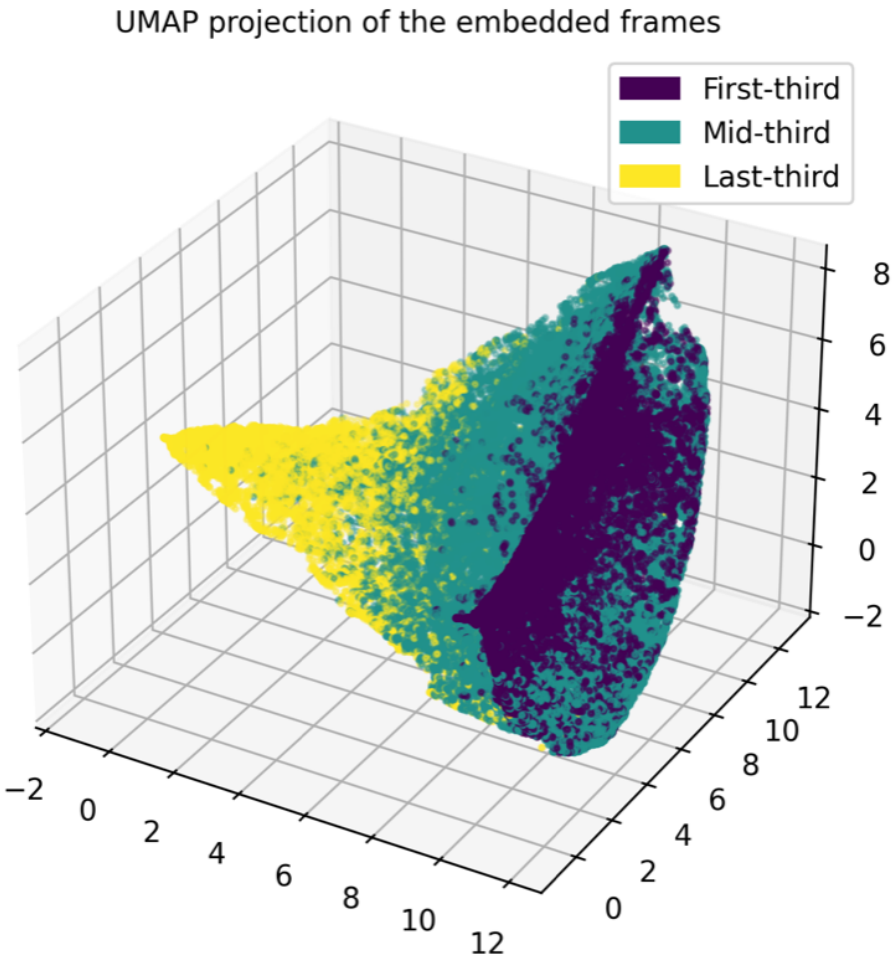

Figure 4.5: UMAP 3D projection of the embedding—labels phases

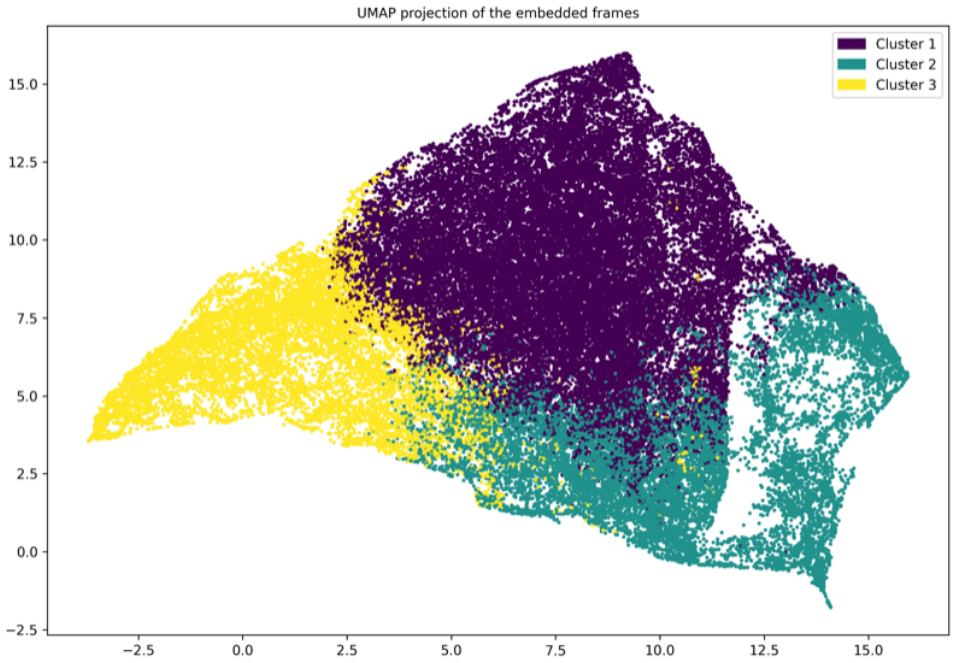

Figure 4.6: Plot of the K-means clustering

metrics about them. For that, we train a simple logistic regression classifier on the testing dataset only (not seen during training) to try to separate the clusters. In table 4.3, we can observe the result of such a classification task. We observe that we can retrieve a considerable proportion of the labels by training the classifier. Note that the baseline we aim for is a 100% correct classifier as the information is fully contained in the initial frame data (hence, just having access to the initial data would allow us to get the correct answer). The results are the scores obtained on a random test set of the data, i.e. we separate the dataset into a training and test set, with a 70-30% split, train the classifier on the training data and show the result evaluated on the testing set.

Phase	Precision	Recall	F1-score
First-third	0.93	0.80	0.86
Mid-third	0.90	0.95	0.92
Last-third	0.87	0.86	0.87

Table 4.3: Classification results on the task of retrieving the phases.

Similarity analysis

Then, we can see if the clusters made by the model have some meaning regarding the data. We process a K-means clustering on the embedding, shown in Fig. 4.6, to see the clusters. We can see that they are close to the clusters from the phases, i.e. the embedding captures the y-dimensionality of the game, even in an unsupervised manner.

Linear Probing

Similarly, to understand further if the embedding kept the dimensionality of the pitch, we create a linear probing test on the center of mass of all players (and ball) on the pitch. That is we evaluate a simple linear regression. We expect that we can linearly retrieve this information from the embedded data. We report the results in table 4.4, which proves the strong correlation (the result is the R^2 score obtained on a random test set of the data, i.e. we separate the dataset into a training and test set, with a 70-30% split, train the linear regression on the training data and show the result evaluated on the testing set).

Regression task	R^2
x-coordinate	0.92
y-coordinate	0.91

Table 4.4: Linear probing results on predicting the center of mass of all players on the pitch.

Owning Team

Finally, we expect to be able to extract one other important information about the game's state from the embedding: which team is currently in possession of the ball. Fortunately, we have access to this information in the dataset, i.e. for each frame we get a label giving the team in possession of the ball. We report the result of such a task in table 4.5. Again for the classifier, we use a random train-test split as before.

Owning Team	Precision	Recall	F1-score
Home	0.51	0.72	0.60
Away	0.51	0.30	0.38

Table 4.5: Classification results on the task of retrieving the owning team.

We also provide the plot of the embedding in Fig. 4.7 with the points labeled with the owning team label, so that we can get a clearer visualization of such a classification result.

We add this plot from UMAP that we find insightful, see Fig. 4.8. It represents how points are linked in the high-dimensional embedding space (linked to how the manifold projection is done), but the links are bundled together to keep only the main ones (see UMAP doc for more information).

Finally, we did a last assessment. From the value of the embedding (evaluated on the frame without the ball), we tried to predict the position of the ball with a simple one-hidden layer neural network (dimension 200). Here are the results. The mean distance error on the test set was computed using a simple neural network trained with 200 hidden layers over 500 epochs. This evaluation was performed on frames in which the ball was not present (we removed the ball node as well as the ball edges). The resulting mean distance error was **14.99m**. This result is intended for comparison with the findings reported in [Nir24], which managed to get an error of around 13 m trying to predict the position of the ball from the players with complex GNN architecture.

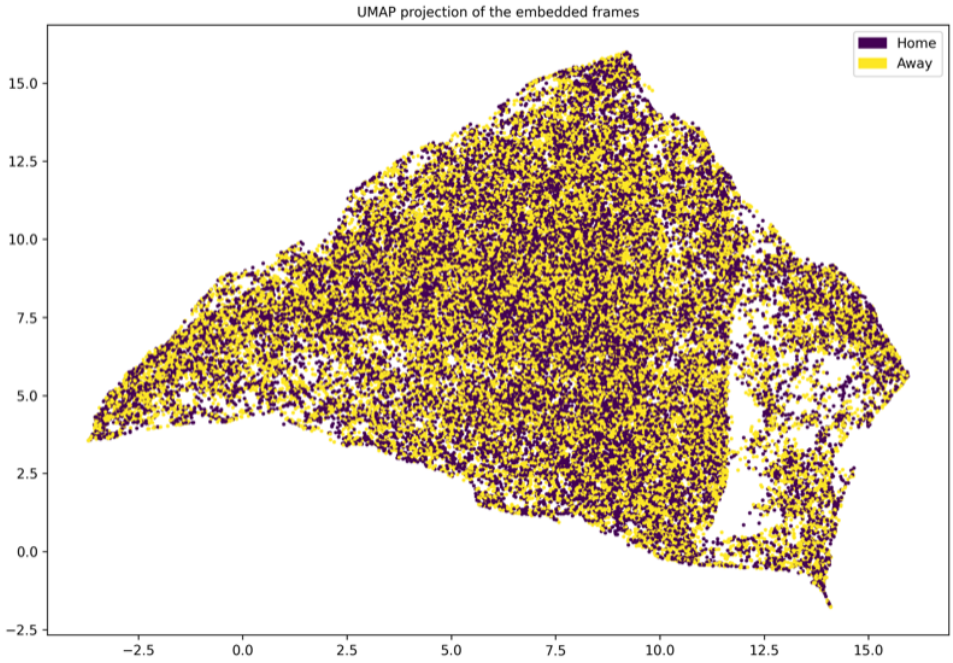

Figure 4.7: UMAP plot of the embedding—labels owning team

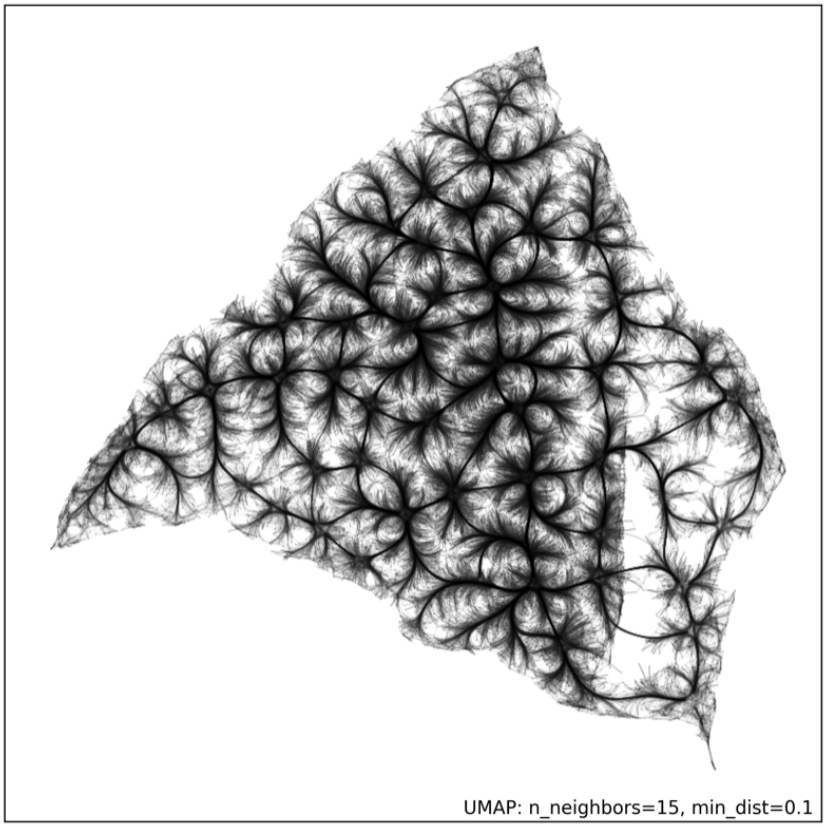

Figure 4.8: UMAP plot of the embedding—connectivity

Chapter 5

Discussion

Further assessment of the learned embeddings is required through a more concrete downstream task, such as learning a potential function. Indeed, just from the observation of the embedding, it's quite difficult to get any conclusive result. Also, the linear separation result can seem great but when compared with other models (that were less accurate on the pre-training task), this puts the results into perspective.

A potential path to explore to improve the embedding could be about the pre-training process, by incorporating a graph-level task, or multiple such tasks, as suggested in [Hu+20].

Another enhancement involves including categorical variables in the prediction of masked features. We initially attempted this approach but it proved unsuccessful due to the need for careful handling of loss functions (MSE and cross-entropy loss are not comparable as is).

Inspired by the BERT (and RoBERTa) framework, an augmentation strategy could be implemented where, with a random probability, a masked player's features are either not masked or replaced by randomly sampled values from the dataset, or not masked (to let the model see some full frames).

A further research direction involves integrating temporal dependencies within the graph structure.

Several other unexplored approaches (that proved successful in other domains or tasks) remain, including:

- Spatiotemporal Graph Neural Networks (GNNs)
- Graph transformers
- Graph recurrent neural networks
- Pre-training on dynamic graph neural networks
- Pre-training via prompting
- Autoencoder-based GNN approaches such as Graph Masked Autoencoder (GMAE), GMAE2,
- Variational Graph Autoencoder (VGAE), and Graph Contrastive Learning (GraphCL).

Chapter 6

Conclusion

The findings of this study indicate that the model is capable of extracting meaningful information about the game, demonstrating an ability to learn specific aspects of play. However, its applicability remains too narrow, as it is primarily optimized for a single task. To achieve a more comprehensive understanding of game dynamics, the model should be trained on a broader set of tasks.

As with any model, the representation learned is inherently limited and offers only a partial perspective of the game. This limitation is reflected in the model's results.

Nonetheless, the architecture of Graph Neural Networks (GNNs) has shown promise in its ability to capture both local and global information effectively. Furthermore, the flexibility of the pre-training approach allows for fine-tuning of the model while leveraging the learned embeddings for various downstream applications. These findings underscore the potential of GNN-based approaches for analyzing sports data and highlight opportunities for further refinement and generalization.

Bibliography

[Adj24] Akedjou Achraff Adjileye. *RisingBALLER: A Player Is a Token, a Match Is a Sentence, A Path towards a Foundational Model for Football Players Data Analytics*. Oct. 1, 2024. DOI: 10.48550/arXiv.2410.00943. arXiv: 2410.00943 [cs]. URL: http://arxiv.org/abs/2410.00943 (visited on 02/07/2025). Pre-published.

[Ala24] Ben Alamar. *Sports Analytics: A Guide for Coaches, Managers, and Other Decision Makers*. Second edition. New York: Columbia University Press, 2024. 164 pp. ISBN: 978-0-231-20520-7.

[Anz+22] Gabriel Anzer et al. "Detection of Tactical Patterns Using Semi-Supervised Graph Neural Networks". In: MIT Sloan Sports Analytics Conference. Boston, USA, Mar. 2022. URL: https://cdn.prod.website-files.com/5f1af76ed86d6771ad48324b/6227709e4d7acb78147f7bcf_Detection%20of%20Tactical%20Patterns%202.pdf.

[BHS24] Ethan Baron, Daniel Hocevar, and Zach Salehe. *A Foundation Model for Soccer*. July 18, 2024. DOI: 10.48550/arXiv.2407.14558. arXiv: 2407.14558 [cs]. URL: http://arxiv.org/abs/2407.14558 (visited on 03/20/2025). Pre-published.

[BS24] Joris Bekkers and Amod Sahasrabudhe. *A Graph Neural Network Deep-Dive into Successful Counterattacks*. Dec. 30, 2024. DOI: 10.48550/arXiv.2411.17450. arXiv: 2411.17450 [cs]. URL: http://arxiv.org/abs/2411.17450 (visited on 03/20/2025). Pre-published.

[Bra24] Ulrik Brandes. "A Goal-Aligned Coordinate System for Invasion Games". In: *Journal of Sports Analytics* 9.4 (Feb. 2, 2024), pp. 261–271. ISSN: 2215-020X, 2215-0218. DOI: 10.3233/JSA-220706. URL: https://journals.sagepub.com/doi/10.3233/JSA-220706 (visited on 03/19/2025).

[BV19] Lotte Bransen and Jan Van Haaren. "Measuring Football Players' On-the-Ball Contributions from Passes During Games". In: *Machine Learning and Data Mining for Sports Analytics*. Ed. by Ulf Brefeld et al. Vol. 11330. Cham: Springer International Publishing, 2019, pp. 3–15. ISBN: 978-3-030-17274-9. DOI: 10.1007/978-3-030-17274-9_1. URL: https://link.springer.com/10.1007/978-3-030-17274-9_1 (visited on 03/20/2025).

[Cap+24] Guillem Capellera et al. "Footbots: A Transformer-Based Architecture for Motion Prediction in Soccer". In: *2024 IEEE International Conference on Image Processing (ICIP)*. 2024 IEEE International Conference on Image Processing (ICIP). Abu Dhabi, United Arab Emirates: IEEE, Oct. 27, 2024, pp. 2313–2319. ISBN: 979-8-3503-4939-9. DOI: 10.1109/ICIP51287.2024.10647396. URL: https://ieeexplore.ieee.org/document/10647396/ (visited on 03/20/2025).

[Cas23] Giovanni Castellano. "Graph Neural Networks for Events Detection in Football". MA thesis. KTH, School of Electrical Engineering and Computer Science (EECS), 2023. 49 pp. URL: https://kth.diva-portal.org/smash/get/diva2:1845172/FULLTEXT01.pdf.

[Cer+16] Daniel Cervone et al. "A Multiresolution Stochastic Process Model for Predicting Basketball Possession Outcomes". In: *Journal of the American Statistical Association* 111.514 (Apr. 2, 2016), pp. 585–599. ISSN: 0162-1459, 1537-274X. DOI: 10.1080/01621459.2016.1141685. URL: https://www.tandfonline.com/doi/full/10.1080/01621459.2016.1141685 (visited on 03/20/2025).

[Cha20] Chaitanya Joshi. *Transformers Are Graph Neural Networks*. NTU Graph Deep Learning Lab. Feb. 12, 2020. URL: https://graphdeeplearning.github.io/post/transformers-are-gnns/.

[Dec+19] Tom Decroos et al. "Actions Speak Louder than Goals: Valuing Player Actions in Soccer". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Anchorage AK USA: ACM, July 25, 2019, pp. 1851–1861. ISBN: 978-1-4503-6201-6. DOI: 10.1145/3292500.3330758. URL: https://dl.acm.org/doi/10.1145/3292500.3330758 (visited on 03/19/2025).

[Dev+19] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North*. Proceedings of the 2019 Conference of the North. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: http://aclweb.org/anthology/N19-1423 (visited on 03/30/2025).

[EPT05] J. Ensum, R. Pollard, and S. Taylor. "Applications of Logistic Regression to Shots at Goal in Association Football". In: *The Proceedings of the Fifth World Congress on Sports Science and Football*. 5. London: Routledge, 2005, Chapter 31. ISBN: 978-0-203-41299-2. DOI: 10.4324/9780203412992. URL: https://www.taylorfrancis.com/chapters/edit/10.4324/9780203412992-78/applications-logistic-regression-shots-goal-association-football.

[Eve+23] Gregory Everett et al. *Inferring Player Location in Sports Matches: Multi-Agent Spatial Imputation from Limited Observations*. Feb. 13, 2023. DOI: 10.48550/arXiv.2302.06569. arXiv: 2302.06569 [cs]. URL: http://arxiv.org/abs/2302.06569 (visited on 03/31/2025). Pre-published.

[Fab24] Hugo Fabrègues. "Markov Modelling of Soccer Matches and Learning of Game States' Potential". In: (July 1, 2024), 46 p. DOI: 10.3929/ETHZ-B-000691549. URL: http://hdl.handle.net/20.500.11850/691549 (visited on 03/20/2025).

[FBC21] Javier Fernández, Luke Bornn, and Daniel Cervone. "A Framework for the Fine-Grained Evaluation of the Instantaneous Expected Value of Soccer Possessions". In: *Machine Learning* 110.6 (June 2021), pp. 1389–1427. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-021-05989-6. URL: https://link.springer.com/10.1007/s10994-021-05989-6 (visited on 03/19/2025).

[FIF22] FIFA, director. *Argentina v France | Final | FIFA World Cup 2022 | Full Match Replay*. Dec. 18, 2022. URL: https://www.plus.fifa.com/en/content/7047fe21-4af1-476c-9fb8-e4655c5668e2?gl=fr.

[GSN24] Thales De Oliveira Gonçalves, Claudio Silva, and Luis Gustavo Nonato. *Gnn-Vsp: A Graph Neural Network Model for Valuing Soccer Players*. 2024. DOI: 10.2139/ssrn.4765249. URL: https://www.ssrn.com/abstract=4765249 (visited on 03/20/2025). Pre-published.

[GS16] Laszlo Gyarmati and Rade Stanojevic. *QPass: A Merit-based Evaluation of Soccer Passes*. Aug. 8, 2016. DOI: 10.48550/arXiv.1608.03532. arXiv: 1608.03532 [cs]. URL: http://arxiv.org/abs/1608.03532 (visited on 03/20/2025). Pre-published.

[Ham20] William L. Hamilton. "Graph Representation Learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14.3 (2020), pp. 1–159.

[HYL17] William L. Hamilton, Rex Ying, and Jure Leskovec. *Representation Learning on Graphs: Methods and Applications*. Version 3. 2017. DOI: 10.48550/ARXIV.1709.05584. URL: https://arxiv.org/abs/1709.05584 (visited on 03/30/2025). Pre-published.

[HW02] N Hirotsu and M Wright. "Using a Markov Process Model of an Association Football Match to Determine the Optimal Timing of Substitution and Tactical Decisions". In: *Journal of the Operational Research Society* 53.1 (Jan. 2002), pp. 88–96. ISSN: 0160-5682. DOI: 10.1057/palgrave/jors/2601254. URL: http://www.palgrave-journals.com/doifinder/10.1057/palgrave/jors/2601254 (visited on 03/20/2025).

[HS22] Shengnan Hu and Gita Sukthankar. *Predicting Team Performance with Spatial Temporal Graph Convolutional Networks*. June 21, 2022. DOI: 10.48550/arXiv.2206.10720. arXiv: 2206.10720 [cs]. URL: http://arxiv.org/abs/2206.10720 (visited on 03/31/2025). Pre-published.

[Hu+20] Weihua Hu et al. *Strategies for Pre-training Graph Neural Networks*. Feb. 18, 2020. DOI: 10.48550/arXiv.1905.12265. arXiv: 1905.12265 [cs]. URL: http://arxiv.org/abs/1905.12265 (visited on 03/19/2025). Pre-published.

[JCK25] Jacky Hao Jiang, Jerry Cai, and Anastasios Kyrillidis. *Unveiling Hidden Pivotal Players with GoalNet: A GNN-Based Soccer Player Evaluation System*. Mar. 12, 2025. DOI: 10.48550/arXiv.2503.09737. arXiv: 2503.09737 [cs]. URL: http://arxiv.org/abs/2503.09737 (visited on 03/20/2025). Pre-published.

[Jua21] Juan Camilo Campos. "Determining the Phases of Play Using Graph Neural Network Embeddings". In: Statsbomb Conference. Nov. 2021. URL: https://statsbomb.com/wp-content/uploads/2021/11/Juan-Camilo-Campos.pdf.

[Liu+22] Jinglong Liu et al. "Graph Neural Network Based Agent in Google Research Football". In: *2nd International Conference on Artificial Intelligence, Automation, and High-Performance Computing (AIAHPC 2022)*. 2nd International Conference on Artificial Intelligence, Automation, and High-Performance Computing (AIAHPC 2022). Ed. by Ligu Zhu. Zhuhai, China: SPIE, Nov. 11, 2022, p. 121. ISBN: 978-1-5106-5771-7. DOI: 10.1117/12.2641817. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/12348/2641817/Graph-neural-network-based-agent-in-Google-Research-Football/10.1117/12.2641817.full (visited on 03/20/2025).

[Liu+19] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. Version 1. 2019. DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692 (visited on 03/31/2025). Pre-published.

[Luc+15] Patrick Lucey et al. "Quality vs Quantity: Improved Shot Prediction in Soccer Using Strategic Features from Spatiotemporal Data". In: *MIT*. MIT Sloan Conference (2015).

[LK23] Rui Luo and Vikram Krishnamurthy. *Who You Play Affects How You Play: Predicting Sports Performance Using Graph Attention Networks With Temporal Convolution*. Mar. 29, 2023. DOI: 10.48550/arXiv.2303.16741. arXiv: 2303.16741 [cs]. URL: http://arxiv.org/abs/2303.16741 (visited on 03/31/2025). Pre-published.

[Mag21] Ofir Magdaci. *Embedding the Language of Football Using NLP*. Hands-on Tutorials. June 15, 2021. URL: https://towardsdatascience.com/embedding-the-language-of-football-using-nlp-e52dc153afa6/.

[MMM24a] Tiago Mendes-Neves, Luís Meireles, and João Mendes-Moreira. *Estimating Player Performance in Different Contexts Using Fine-tuned Large Events Models*. Apr. 26, 2024. DOI: 10.48550/arXiv.2402.06815. arXiv: 2402.06815 [cs]. URL: http://arxiv.org/abs/2402.06815 (visited on 03/20/2025). Pre-published.

[MMM24b] Tiago Mendes-Neves, Luís Meireles, and João Mendes-Moreira. *Forecasting Events in Soccer Matches Through Language*. Apr. 26, 2024. DOI: 10.48550/arXiv.2402.06820. arXiv: 2402.06820 [cs]. URL: http://arxiv.org/abs/2402.06820 (visited on 03/20/2025). Pre-published.

[MMM24c] Tiago Mendes-Neves, Luís Meireles, and João Mendes-Moreira. "Towards a Foundation Large Events Model for Soccer". In: *Machine Learning* 113.11–12 (Dec. 2024), pp. 8687–8709. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-024-06606-y. URL: https://link.springer.com/10.1007/s10994-024-06606-y (visited on 03/20/2025).

[Nir24] Nirvin, Gustav. "Ball Tracking in Association Football - Leveraging Player Detections to Locate the Ball Using Graph Attention Networks". Lund University, 2024. URL: http://lup.lub.lu.se/student-papers/record/9172881.

[Och+25] Jérémie Ochin et al. "Game State and Spatio-Temporal Action Detection in Soccer Using Graph Neural Networks and 3D Convolutional Networks:" in: *Proceedings of the 14th International Conference on Pattern Recognition Applications and Methods*. 14th International Conference on Pattern Recognition Applications and Methods. Porto, Portugal: SCITEPRESS - Science and Technology Publications, 2025, pp. 636–646. ISBN: 978-989-758-730-6. DOI: 10.5220/0013161100003905. URL: https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0013161100003905 (visited on 03/31/2025).

[PDC24] Yisheng Pei, Varuna De Silva, and Mike Caine. "Passing Heatmap Prediction Based on Transformer Model Using Tracking Data for Football Analytics". In: *Intelligent Systems and Pattern Recognition*. Ed. by Akram Bennour, Ahmed Bouridane, and Lotfi Chaari. Vol. 1940. Cham: Springer Nature Switzerland, 2024, pp. 162–173. ISBN: 978-3-031-46335-8. DOI: 10.1007/978-3-031-46335-8_13. URL: https://link.springer.com/10.1007/978-3-031-46335-8_13 (visited on 03/20/2025).

[PR97] Richard Pollard and Charles Reep. "Measuring the Effectiveness of Playing Strategies at Soccer". In: *Journal of the Royal Statistical Society: Series D (The Statistician)* 46.4 (Dec. 1997), pp. 541–550. ISSN: 0039-0526, 1467-9884. DOI: 10.1111/1467-9884.00108. URL: https://onlinelibrary.wiley.com/doi/10.1111/1467-9884.00108 (visited on 03/20/2025).

[Rud11] Sarah Rudd. "A Framework for Tactical Analysis and Individual Offensive Production Assessment in Soccer Using Markov Chains". New England Symposium on Statistics in Sports. 2011. URL: https://nessis.org/nessis11/rudd.pdf.

[RB24] Yannick Rudolph and Ulf Brefeld. "Masked Autoencoder Pretraining for Event Classification in Elite Soccer". In: *Machine Learning and Data Mining for Sports Analytics*. Ed. by Ulf Brefeld et al. Vol. 2035. Cham: Springer Nature Switzerland, 2024, pp. 24–35. ISBN: 978-3-031-53833-9. DOI: 10.1007/978-3-031-53833-9_3. URL: https://link.springer.com/10.1007/978-3-031-53833-9_3 (visited on 03/20/2025).

[San+21] Benjamin Sanchez-Lengeling et al. "A Gentle Introduction to Graph Neural Networks". In: *Distill* 6.8 (Aug. 17, 2021), 10.23915/distill.00033. ISSN: 2476-0757. DOI: 10.23915/distill.00033. URL: https://distill.pub/2021/gnn-intro (visited on 03/20/2025).

[Sch24] Andrej Scheuer. "Recognising Group-Tactical Manoeuvres from Player Positions". In: (Sept. 1, 2024), 46 p. DOI: 10.3929/ETHZ-B-000693656. URL: http://hdl.handle.net/20.500.11850/693656 (visited on 03/31/2025).

[Sim+22] Ian Simpson et al. "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*. KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Washington DC USA: ACM, Aug. 14, 2022, pp. 3898–3908. ISBN: 978-1-4503-9385-0. DOI: 10.1145/3534678.3539138. URL: https://dl.acm.org/doi/10.1145/3534678.3539138 (visited on 03/20/2025).

[Sin19] Karun Singh. *Introducing Expected Threat (xT)*. karun.in/blog. 2019. URL: https://karun.in/blog/expected-threat.html.

[Stö+21] Michael Stöckl et al. "Making Offensive Play Predictable - Using a Graph Convolutional Network to Understand Defensive Performance in Soccer". In: *Proceedings of the 15th MIT Sloan Sports Analytics Conference*. MIT Sloan Sports Analytics Conference. Virtual, Apr. 2021. URL: https://www.statsperform.com/wp-content/uploads/2021/04/Making-Offensive-Play-Predictable.pdf.

[Vas+17] Ashish Vaswani et al. *Attention Is All You Need*. Version 7. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762 (visited on 04/04/2025). Pre-published.

[Vel+18] Petar Veličković et al. *Deep Graph Infomax*. Version 2. 2018. DOI: 10.48550/ARXIV.1809.10341. URL: https://arxiv.org/abs/1809.10341 (visited on 03/30/2025). Pre-published.

[Wan+24] Zhe Wang et al. "TacticAI: An AI Assistant for Football Tactics". In: *Nature Communications* 15.1 (Mar. 19, 2024), p. 1906. ISSN: 2041-1723. DOI: 10.1038/s41467-024-45965-x. URL: https://www.nature.com/articles/s41467-024-45965-x (visited on 03/19/2025).

[WES] WEST SUSSEX RECORD OFFICE. *The Earliest Known Written Rules of Cricket, 1727 (Goodwood Mss 1884)*. WEST SUSSEX RECORD OFFICE. URL: https://westsussexrecordofficeblog.com/2016/07/11/the-earliest-known-written-rules-of-cricket-1727-goodwood-mss-1884/.

[XS21] Peter Xenopoulos and Claudio Silva. "Graph Neural Networks to Predict Sports Outcomes". In: *2021 IEEE International Conference on Big Data (Big Data)*. 2021 IEEE International Conference on Big Data (Big Data). Orlando, FL, USA: IEEE, Dec. 15, 2021, pp. 1757–1763. ISBN: 978-1-6654-3902-2. DOI: 10.1109/BigData52589.2021.9671833. URL: https://ieeexplore.ieee.org/document/9671833/ (visited on 03/20/2025).

[YYL20] Jiaxuan You, Rex Ying, and Jure Leskovec. *Design Space for Graph Neural Networks*. Version 2. 2020. DOI: 10.48550/ARXIV.2011.08843. URL: https://arxiv.org/abs/2011.08843 (visited on 03/31/2025). Pre-published.

[Zey+24] Zeyu Wang et al. "Graph Neural Network Recommendation System for Football Formation". In: *Applied Science and Biotechnology Journal for Advanced Research* 3.3 (May 29, 2024), pp. 33–39. ISSN: 2583-553X. DOI: 10.5281/ZENODO.12198843. URL: https://zenodo.org/doi/10.5281/zenodo.12198843 (visited on 03/31/2025).

Appendix A

Other GNN layers

We report here for information purposes all the losses (on training and validation) that we have obtained by testing different GNN architectures, before changes in the model, which further improved the accuracy, were operated. As the final approach is close to the one tested, we consider these results to guide the model selection for the pre-training task accuracy, to understand which architecture could work well in our setup. Note that we didn't analyze the results on the embedding for all the GNN layers, hence we cannot conclude anything regarding this.

All layer architectures tested were tested with 2 layers each, a hidden dimension of 64 and the task head was a fully-connected neural network with a hidden layer of dimension 10. The output was a 10-dimensional hidden layer.

While the careful reader will notice that those results are slightly better than the ones exposed in chapter 4 on page 24, this is not a mistake. Indeed, this is because the final affine layer was, at the time of the simulations, replaced by a neural network with one hidden layer (of dimension 10). Thus, the a slight improvement for some models. However, as the project went through, we questioned this choice and moved to an affine layer. The reason behind that is that we would like the hidden representation to be as representative as possible. Hence, by only allowing the model to use an affine transformation in the final layer, we force it to learn as much as possible in the final hidden representations (and in the parameters that allow this representation before) and not in another intermediate hidden representation that we discard after the pre-training task. Regarding the interpretation of those results, we were still able to observe some interesting insights into the hidden representations of the learned models, which guided our choice of the selected models. This is also the reason why we keep this part in the appendix since this represents an intuition rather than a strictly speaking result.

Layer	Loss (MSE)		Distance (m)	
	Training	Validation	Training	Validation
GCN	51.7	55.9	8.2	8.6
GraphConv	29.1	30.5	5.5	5.5
ChecbConv ($K = 2$)	15.2	16.7	4.7	4.8
SAGEConv	18.8	20.2	5.2	5.3
GatedGraphConv	195	216	17.8	17.8
ResGatedConv	18.3	25.4	5.8	5.9
GATConv	232	243	19.5	19.9
GATv2Conv	362	376	24.4	24.7
TransformerConv	41.9	53.8	7.5	7.6
AGNNConv	186	187	17.0	17.5
TAGConv	13.5	15.5	4.4	4.7
GINConv	69.6	75.3	10.0	10.0
GINEConv	61.0	64.0	9.2	9.7
ArmaConv	20.0	19.8	5.5	5.4
SGCConv ($K = 1$)	55.5	52.8	8.9	8.6
SSGConv ($\alpha = 0.5$)	41.9	40.2	7.7	7.6
APPNP ($K = 1, \alpha = 0.5$)	208	195	18.1	17.4
CGConv	23.2	20.1	5.8	5.4
MFConv	24.8	33.4	6.0	6.9
NNConv	376	368	24.9	24.6
EdgeConv	17.3	18.9	5.1	5.3
FeatStConv	68.7	71.4	9.7	9.5
LEConv	24.7	26.3	6	6.1
ClusterGCNConv	16.7	16.6	4.9	4.8
GENConv	17.7	44.3	5.1	8.0
GCN2Conv	63.7	68.9	9.3	9.4
PANConv	55.3	61.6	8.7	9.5
WLContinuous	226	227	19	19
FAConv	225	229	18.9	19.1
EGConv	376	368	24	24.6
PDNConv	53	61	8.5	9.1
LGConv	232	232	19.4	19.4
AntiSymmetricConv	172	169	16.7	16.6
MixHopConv	17.2	19.6	4.9	5.3

Table A.1: Comparison of different layer performance on the training and validation data for the pre-training task.

Appendix B

Embedding assessment—other models

Ball prediction using a simple NN:

- Config 4: Mean distance: 12.95 m,
- Config 3: Mean distance: 13.03 m,
- Config 5: Mean distance: 11.97 m,
- Config 7: Mean distance: 13.10 m,
- Config 1: Mean distance: 14.09 m,
- Config 6: Mean distance: 19.87 m.

Phase	Precision	Recall	F1-score
First-third	0.93	0.93	0.93
Mid-third	0.94	0.95	0.95
Last-third	0.92	0.87	0.90

Table B.1: Config 4, Classification results on the task of retrieving the phases.

Regression task	R^2
x-coordinate	0.997
y-coordinate	0.997

Table B.2: Config 4, Linear probing results on predicting the center of mass of all players on the pitch.

Owning Team	Precision	Recall	F1-score
Home	0.54	0.61	0.57
Away	0.52	0.45	0.48

Table B.3: Config 4, Classification results on the task of retrieving the owning team.

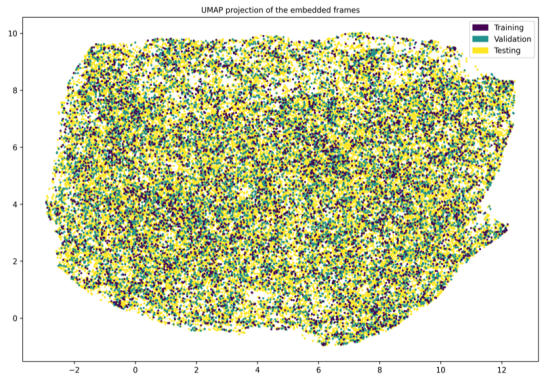

Figure B.1: Config 4, UMAP projection of the embedding—labels train-val-test

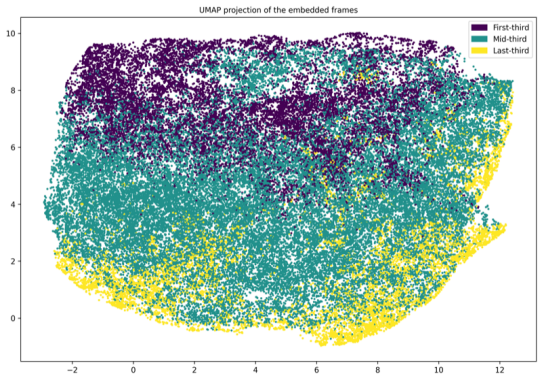

Figure B.2: Config 4, UMAP projection of the embedding—labels phases

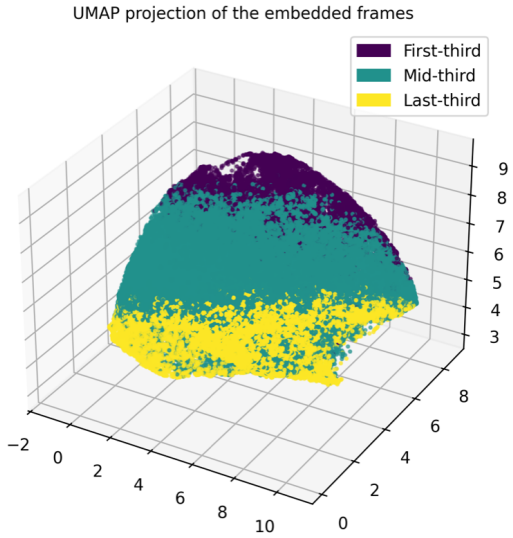

Figure B.3: Config 4, UMAP 3D projection of the embedding—labels phases

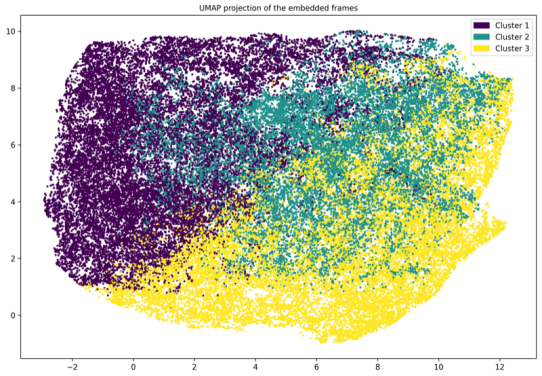

Figure B.4: Config 4, Plot of the K-means clustering

Phase	Precision	Recall	F1-score
First-third	0.92	0.93	0.93
Mid-third	0.95	0.95	0.95
Last-third	0.92	0.90	0.91

Table B.4: Config 3, Classification results on the task of retrieving the phases.

Regression task	R^2
x-coordinate	0.995
y-coordinate	0.996

Table B.5: Config 3, Linear probing results on predicting the center of mass of all players on the pitch.

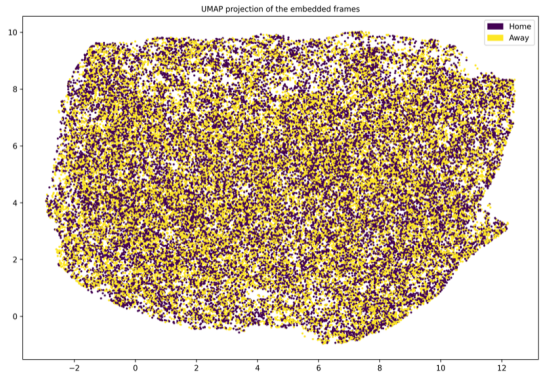

Figure B.5: Config 4, UMAP plot of the embedding—labels owning team

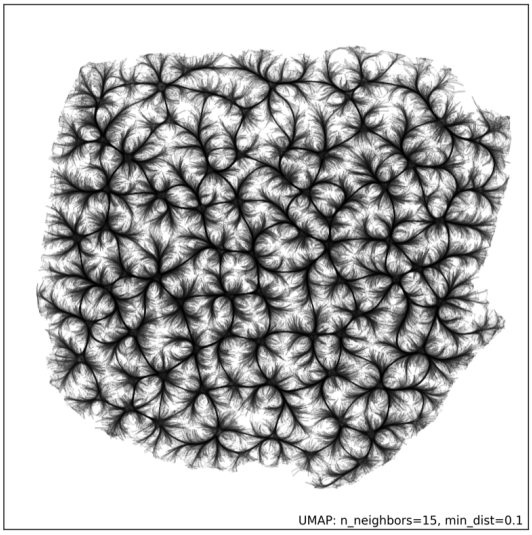

Figure B.6: Config 4, UMAP plot of the embedding—connectivity

Owning Team	Precision	Recall	F1-score
Home	0.51	0.85	0.63
Away	0.47	0.14	0.21

Table B.6: Config 3, Classification results on the task of retrieving the owning team.

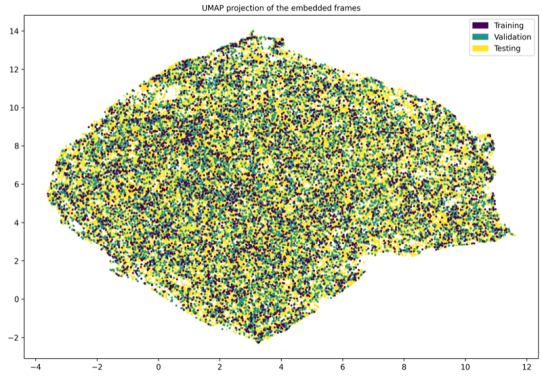

Figure B.7: Config 3, UMAP projection of the embedding—labels train-val-test

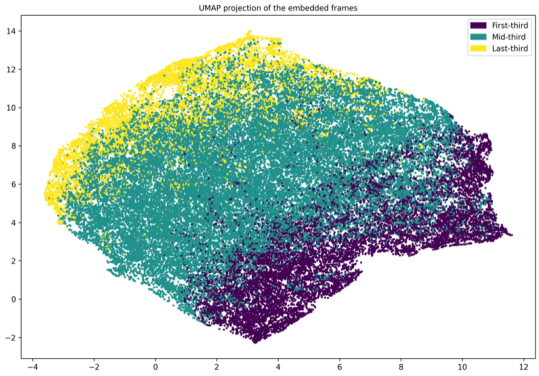

Figure B.8: Config 3, UMAP projection of the embedding—labels phases

Phase	Precision	Recall	F1-score
First-third	0.92	0.89	0.90
Mid-third	0.91	0.94	0.93
Last-third	0.91	0.85	0.88

Table B.7: Config 5, Classification results on the task of retrieving the phases.

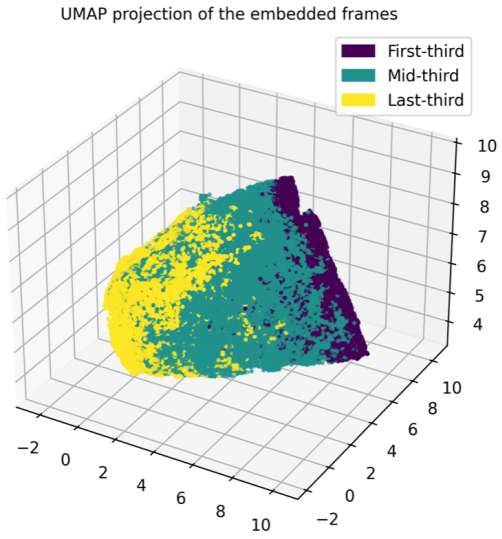

Figure B.9: Config 3, UMAP 3D projection of the embedding—labels phases

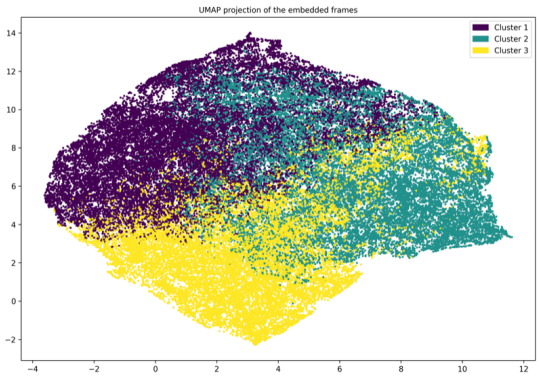

Figure B.10: Config 3, Plot of the K-means clustering

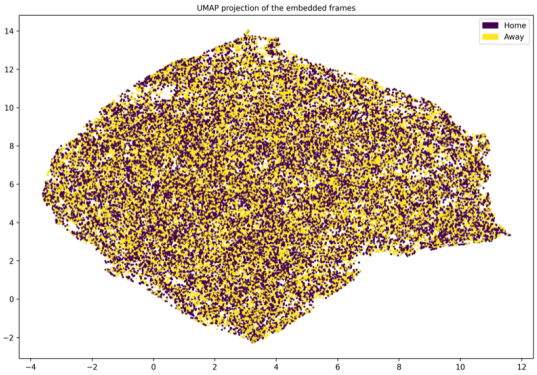

Figure B.11: Config 3, UMAP plot of the embedding—labels owning team

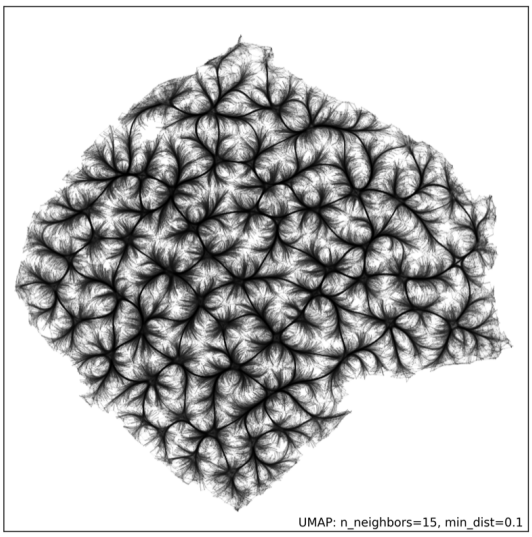

Figure B.12: Config 3, UMAP plot of the embedding—connectivity

Regression task	R^2
x-coordinate	0.94
y-coordinate	0.96

Table B.8: Config 5, Linear probing results on predicting the center of mass of all players on the pitch.

Owning Team	Precision	Recall	F1-score
Home	0.50	0.68	0.58
Away	0.45	0.29	0.35

Table B.9: Config 5, Classification results on the task of retrieving the owning team.

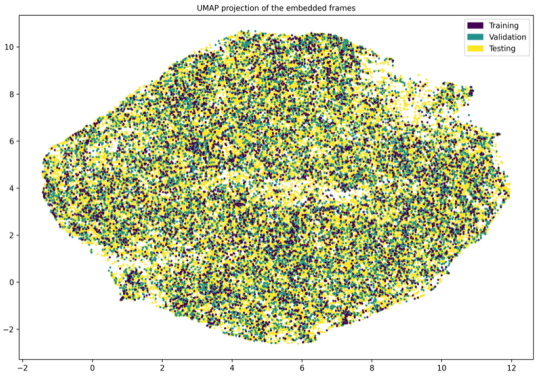

Figure B.13: Config 5, UMAP projection of the embedding—labels train-val-test

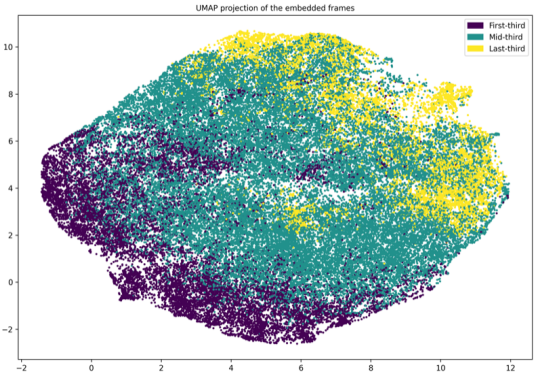

Figure B.14: Config 5, UMAP projection of the embedding—labels phases

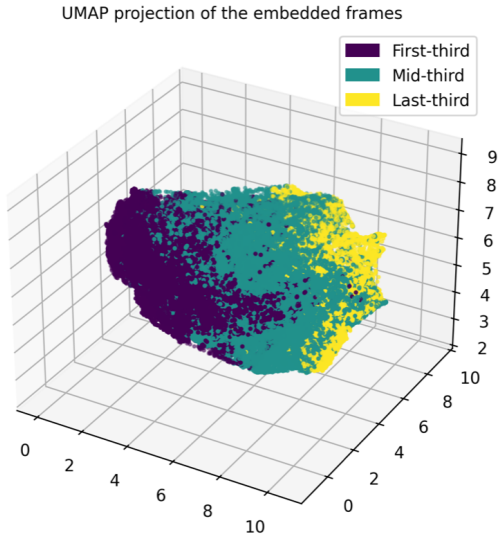

Figure B.15: Config 5, UMAP 3D projection of the embedding—labels phases

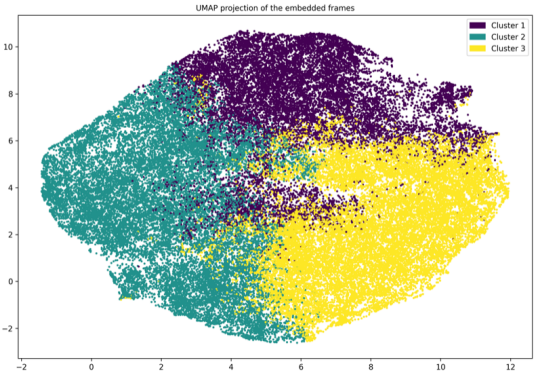

Figure B.16: Config 5, Plot of the K-means clustering

Phase	Precision	Recall	F1-score
First-third	0.92	0.91	0.91
Mid-third	0.92	0.94	0.93
Last-third	0.89	0.83	0.86

Table B.10: Config 7, Classification results on the task of retrieving the phases.

Regression task	R^2
x-coordinate	0.99
y-coordinate	0.99

Table B.11: Config 7, Linear probing results on predicting the center of mass of all players on the pitch.

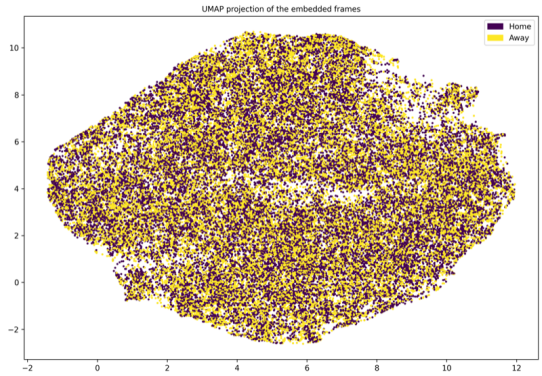

Figure B.17: Config 5, UMAP plot of the embedding—labels owning team

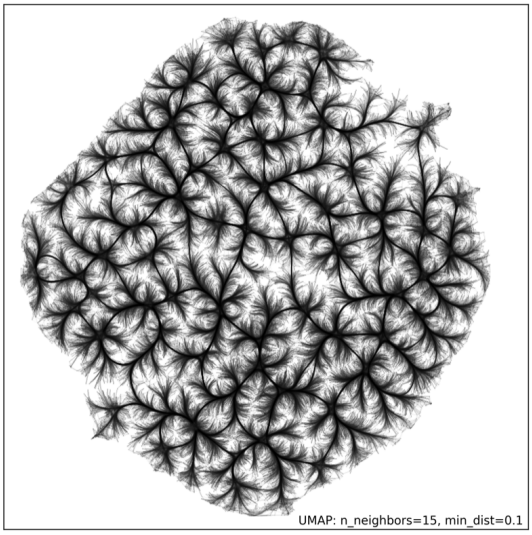

Figure B.18: Config 5, UMAP plot of the embedding—connectivity

Owning Team	Precision	Recall	F1-score
Home	0.80	0.80	0.80
Away	0.78	0.78	0.78

Table B.12: Config 7, Classification results on the task of retrieving the owning team.

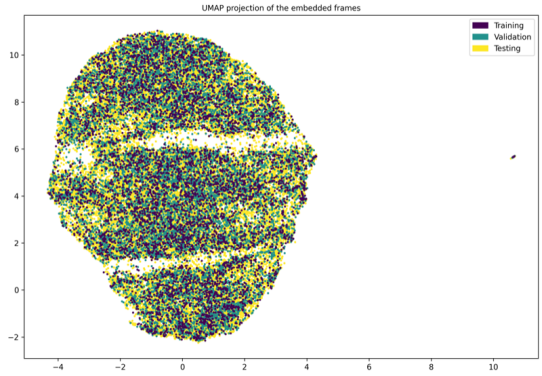

Figure B.19: Config 7, UMAP projection of the embedding—labels train-val-test

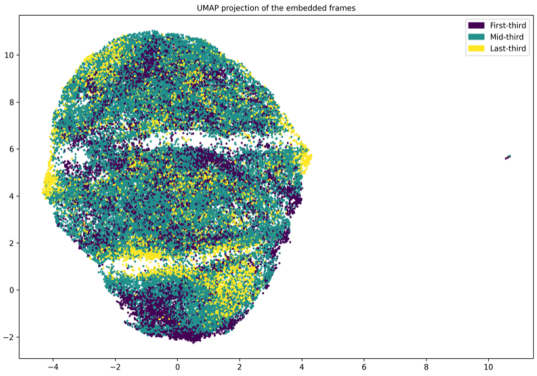

Figure B.20: Config 7, UMAP projection of the embedding—labels phases

Phase	Precision	Recall	F1-score
First-third	0.00	0.00	0.00
Mid-third	0.62	1.00	0.76
Last-third	0.00	0.00	0.00

Table B.13: Config 7, Classification results on the task of retrieving the phases.

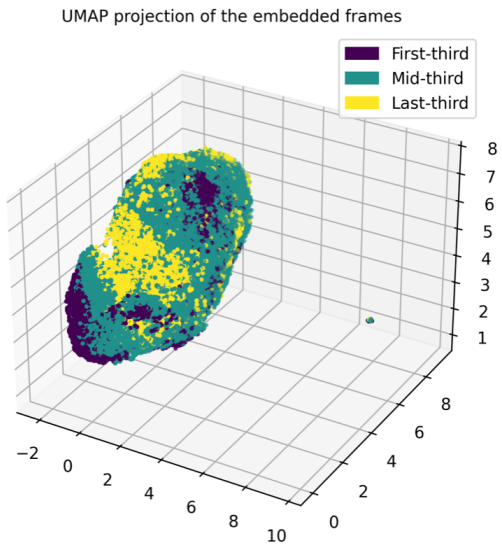

Figure B.21: Config 7, UMAP 3D projection of the embedding—labels phases

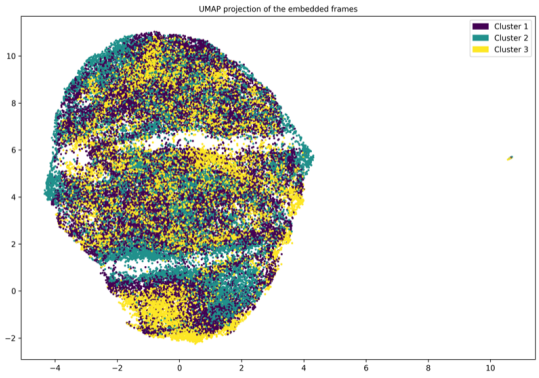

Figure B.22: Config 7, Plot of the K-means clustering

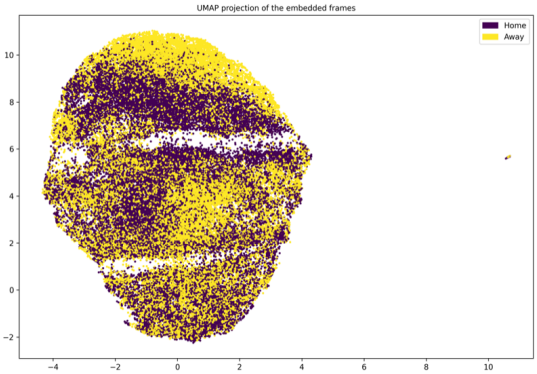

Figure B.23: Config 7, UMAP plot of the embedding—labels owning team

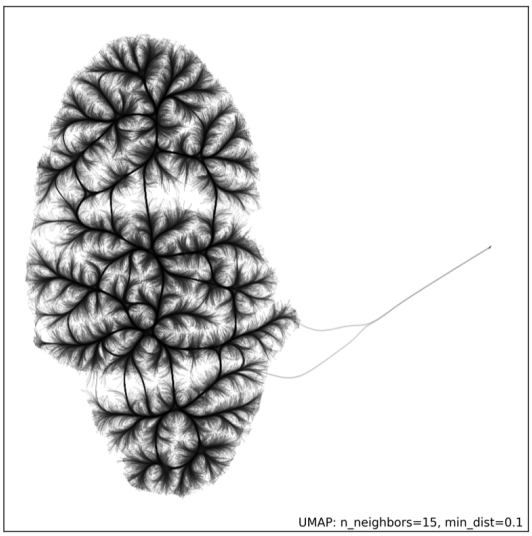

Figure B.24: Config 7, UMAP plot of the embedding—connectivity

Regression task	R^2
x-coordinate	0.83
y-coordinate	0.98

Table B.14: Config 1, Linear probing results on predicting the center of mass of all players on the pitch.

Owning Team	Precision	Recall	F1-score
Home	0.51	0.76	0.61
Away	0.50	0.25	0.34

Table B.15: Config 1, Classification results on the task of retrieving the owning team.

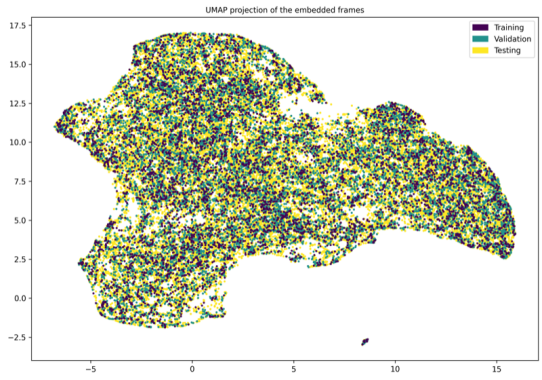

Figure B.25: Config 1, UMAP projection of the embedding—labels train-val-test

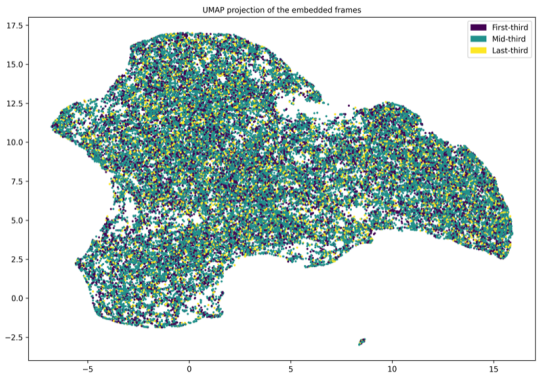

Figure B.26: Config 1, UMAP projection of the embedding—labels phases

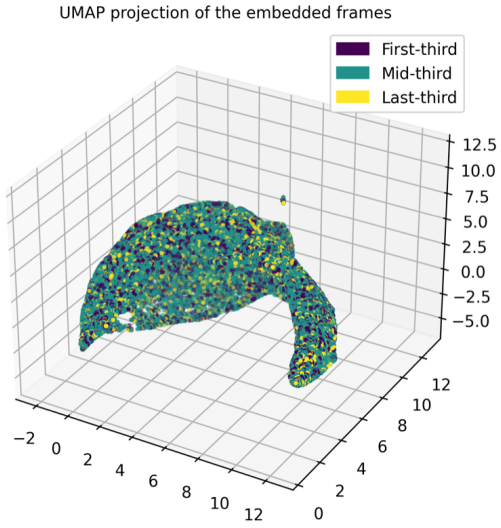

Figure B.27: Config 1, UMAP 3D projection of the embedding—labels phases

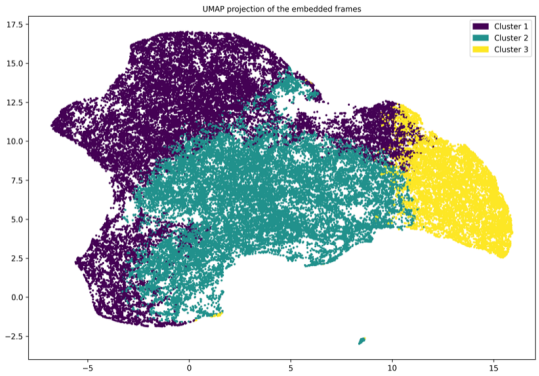

Figure B.28: Config 1, Plot of the K-means clustering

Phase	Precision	Recall	F1-score
First-third	0.00	0.00	0.00
Mid-third	0.62	1.00	0.76
Last-third	0.00	0.00	0.00

Table B.16: Config 6, Classification results on the task of retrieving the phases.

Regression task	R^2
x-coordinate	0.78
y-coordinate	0.81

Table B.17: Config 6, Linear probing results on predicting the center of mass of all players on the pitch.

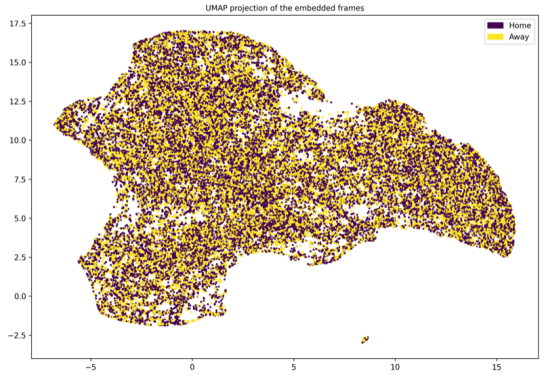

Figure B.29: Config 1, UMAP plot of the embedding—labels owning team

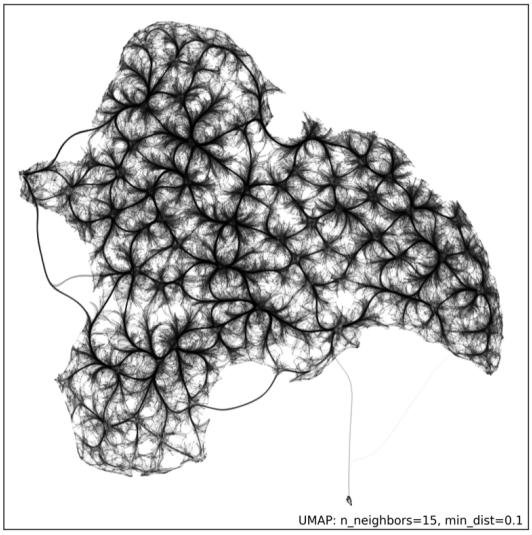

Figure B.30: Config 1, UMAP plot of the embedding—connectivity

Owning Team	Precision	Recall	F1-score
Home	0.51	0.72	0.60
Away	0.50	0.30	0.37

Table B.18: Config 6, Classification results on the task of retrieving the owning team.

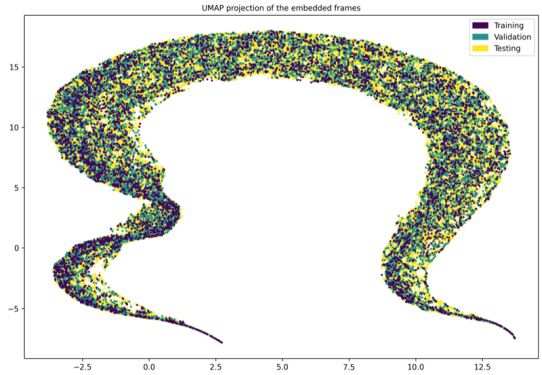

Figure B.31: Config 6, UMAP projection of the embedding—labels train-val-test

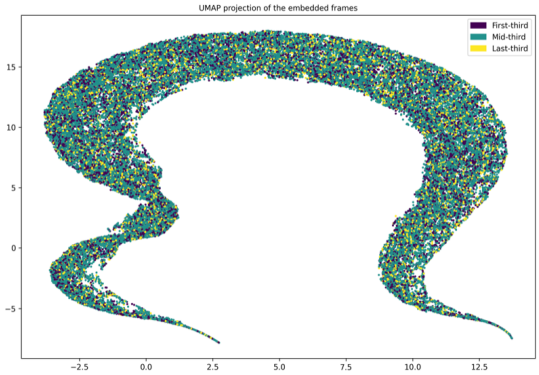

Figure B.32: Config 6, UMAP projection of the embedding—labels phases

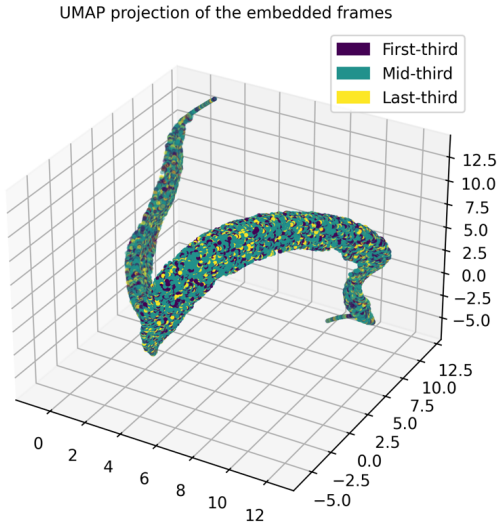

Figure B.33: Config 6, UMAP 3D projection of the embedding—labels phases

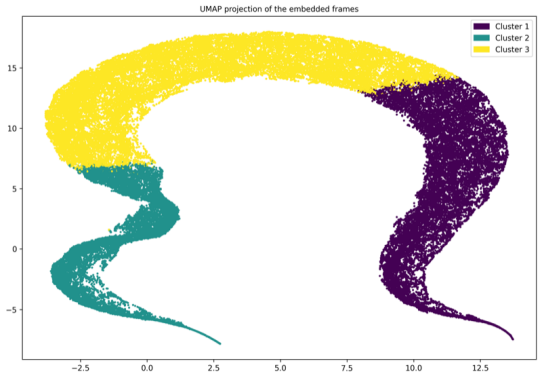

Figure B.34: Config 6, Plot of the K-means clustering

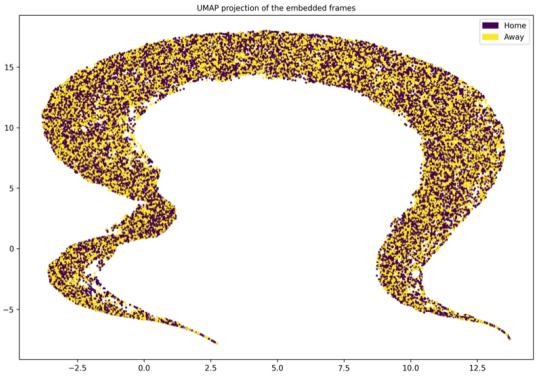

Figure B.35: Config 6, UMAP plot of the embedding—labels owning team

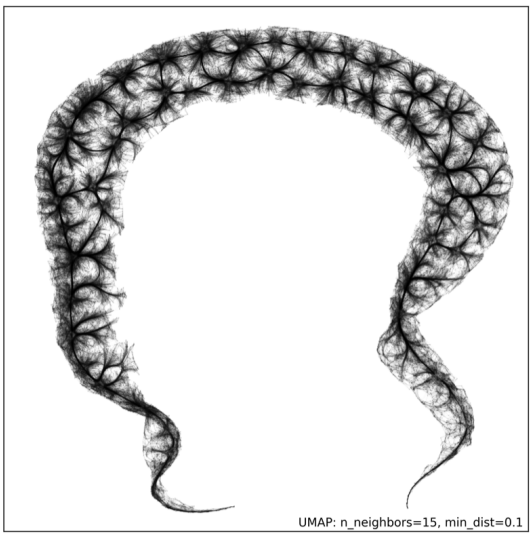

Figure B.36: Config 6, UMAP plot of the embedding—connectivity