A unified spatio-temporal graph model to predict multi-agent movement

Ricardo Furbino[1,2], João Lucas Lage Gonçalves[1,3], Gabriel Valadão Meira[1,4], Wagner Meira Jr.[1], Thiago C. Porto[5], and Adriano C. M. Pereira[1]

[1] Universidade Federal de Minas Gerais, Belo Horizonte MG 31.270-901, Brazil
{ricardofurbino,joao.lucas,gabriel.valadao,meira,adrianoc}@dcc.ufmg.br
[2] src | ftbl
ricardo@srcftbl.com
[3] Gemini Sports Analytics
joao@geminisports.ai
[4] Clube Atlético Mineiro
gabriel.valadao@atletico.com.br
[5] Atalanta Bergamasca Calcio
thiago@atalanta.it

Abstract. Predicting movement in multi-agent continuous systems, such as football, presents significant challenges due to the dynamic and interactive nature of the environment. This work proposes a novel approach to movement prediction by leveraging a graph-unified representation, where football players are modeled as nodes and their interactions in time and space as edges. The proposed architecture, GuardiolAI, integrates Graph Neural Networks (GNNs) with Generative AI techniques, specifically Variational Autoencoders (VAEs), to capture both spatial and temporal dependencies in a unified manner. Unlike traditional methods that process spatial and temporal data separately, our approach models these aspects concurrently within a single graph structure. The methodology involves encoding tracking data from possession sequences into graph-based representations and employing GATv2 layers to learn adaptive attention weights across spatial and temporal dimensions. The model is evaluated using standard movement prediction metrics such as the average displacement error (ADE), mean squared error (MSE), and final displacement error (FDE), and is compared against a constant-velocity baseline. Experimental results demonstrate that the proposed approach achieves competitive performance and provides insight into player interactions and team dynamics.

Keywords: Multi-Agent Systems · Graph Neural Networks · Movement Prediction · Trajectory Forecasting · Generative AI · Football Analytics.

1 Introduction

Recent advances in generative modeling introduce new ways to create and process information. Attention-based models [22] and other techniques to generate

highly relevant and coherent outputs [14, 9] achieved impressive performance in tasks like text and image generation. Another area of interest is the modeling of **multi-agent continuous systems**, in which multiple entities interact within a shared, continuous environment over time. Advances in Graph Neural Networks (GNNs) have shown promise in capturing intricate relationships in such systems. Researchers are exploring how attention mechanisms within GNNs can improve the understanding of complex structured data [23, 5], leading to better predictive models in dynamic environments.

Football is a prime testbed: twenty-two players and a ball interacting in a shared space, with modern tracking systems recording their positions up to 30 Hz. In addition, the trajectory of each player depends both on personal intent and on the real-time movements of teammates and opponents, creating rich, interdependent patterns that drive game strategy. While attention-enhanced GNNs can capture such interdependencies in a graph structure, applying them to forecast future player positions remains underexplored, as far as we were able to verify. Therefore, we raise the question: *Can we build a model that accurately predicts player movements during a football match using generative AI and GNNs?*

This work aims to address this **Movement Prediction (MP)** problem by developing an architecture that predicts agent movements within the complex environment of a football match, using recent advances in deep learning. We cast a possession sequence, during which the team in possession of the ball remains the same, as a **unified graph**: each player in a frame is a node and every pair of players is linked by edges that encode both spatial proximity and temporal succession. The model draws on the generative capabilities of Variational Autoencoders (VAE) [14] and integrates them with the learning power of Graph Neural Networks (GNN) [5], using the unified graph as input and predicting the next seconds of the possession sequence. We benchmark against a constant-velocity baseline, report ADE/FDE/MSE, and inspect attention weights for interpretability, underscoring the promise of unified graph-generative approaches for movement prediction in sports analytics.

2 Related Work

Variational Autoencoders (**VAEs**) [14, 11] learn compact latent distributions and generate diverse, realistic samples across different domains. Although trajectory-forecasting literature often employs conditional VAEs (CVAEs), injecting both the observed past and future as explicit conditional to the decoder [16, 24], this is not strictly required. Recent work shows that a plain VAE—where the decoder receives only a sampled latent z that encodes the past—is sufficient for accurate future prediction: Syed & Morris [21] use a recurrent VAE that encodes pedestrian motion history (plus interaction/scene context) and decodes future trajectories from the latent code. We encode players' trajectories and decode the future movements for players. However, the VAE framework alone does not deal with inter-agent and temporal relationships.

Complex **multi-agent** problems have used Graph Neural Networks (**GNN**s), which extend the Deep Learning framework to handle data structures beyond traditional tabular formats, as an architecture due to its permutation invariance characteristic. GAT [23] stands out by utilizing **attention mechanisms** to prioritizes important nodes, enhancing the model's focus on critical relationships in graph data without relying on heavy computational operations. **GATv2** [5] further improves the flexibility of attention mechanisms, allowing dynamic computation of attention coefficients, particularly useful in complex and dynamic graph structures. Meanwhile, spatio-temporal hybrids (e.g. ASTGCN [10], T-GCN [27]) layer temporal filters on graph convolutions to forecast traffic. These approaches still process space and time separately.

In sports analytics, static GCNs evaluate individual frames for player impact [20, 15], and set-piece models fuse CVAEs with GATv2 for corner prediction [25]. RNN+GCN hybrids map events to tracking data [7], and also use VAE to solve overlapping runs [3] and counterattack [8] detection in soccer. Transformer-only schemes [6] and imitation learning models [12] forecast positions from sequences, while diffusion denoising techniques [13] are being employed with transformers to improve tracking data quality, modeling player movements to enhance data provision. *Yet none embed continuous, multi-frame space–time dependencies in a single graph enhanced with attention.* Our work fills this gap by unifying spatial and temporal modeling for prediction of continuous football movement.

3 Methodology

In this section, we explain the data used to train the model, how we use it to construct a graph representing a possession sequence, the model that encodes and decodes the possession, the experiments made, and the evaluation of the model.

3.1 Spatio-Temporal Possession Sequences Using Tracking Data

We use PFF FC's 30 Hz broadcast tracking data[6] (190 English Premier League 22/23 matches) obtained with computer vision algorithms, fetched via GraphQL / JSON using 'gandula' python package[7] and preprocessed into pandas[8] DataFrames with position, velocity, and acceleration features. We extract continuous **possession sequences**—periods of uninterrupted ball control by a team—and downsample the frames to 5 Hz. Also, we removed possession sequences that included players with abnormal velocities, an issue that can happen in collection.

In sports analytics research, prior work typically separates temporal (RNN or transformer) and spatial (GNN or transformer) modules, or prioritize creating time embeddings before processing spatial relationships, others follow the reverse order, and some fuse them via custom attention schemes [7, 8, 3, 13, 2, 1, 6, 26].

[6] https://fc.pff.com
[7] https://pypi.org/project/gandula/
[8] https://pandas.pydata.org/docs/

We propose a novel approach to model the game as a unified graph structure, that integrates both spatial and temporal dimensions, allowing for simultaneous processing. Formally, the input graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes representing the players, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges encoding spatial and temporal relationships.

In our formulation, a node $v \in \mathcal{V}$ corresponds to a player p_i in a given frame f_{in_j}. Within the same frame, each player p_i is connected to every other player $p_{k \neq i}$ via a **spatial edge** e_s, forming the set $\mathcal{P}_j = p_1, p_2, \ldots, p_{\mathcal{N}}$. This spatial connectivity captures player interactions within each frame.

To incorporate temporal continuity, we construct **temporal edges** e_t that link each player to its previous-frame ($f_{in_{j-1}}$) self, effectively capturing motion dynamics over time. This unified representation enables the simultaneous processing of spatial and temporal relations.

The decoder predicts the frames $j = \mathcal{M} + 1 \ldots \mathcal{M} + \mathcal{T}$, reusing the frame $\mathcal{M}$ as the first output. Sequences shorter than $\mathcal{M} + \mathcal{T}$ are zero-padded with "ghost" nodes to match node counts.

Each node and edge contains multiple features to enrich the learning process. Specifically, we define:

- Node features: (x,y) position, velocity decomposed into x and y axes, acceleration in x and y, angle to the ball, distance to the ball, and a boolean indicating team possession. Position is normalized from 0 to 1 according to pitch length (105m x 68m), and the other features are normalized from 0 to 1 according to max-min in the possession.
- Edge features: booleans indicating same-team relationships, within-frame connectivity (all players in the same frame are connected), and cross-frame temporal links.

We implement all graphs with PyTorch Geometric[9] and focus exclusively on open-play sequences (set-pieces excluded) [17]. For experimentation, we define $\mathcal{M} = 15$ input frames and $\mathcal{T} = 14$ output frames, which correspond to 3 and 2.8 seconds, respectively. For sequences >30 frames, we slide a fixed 30-frame window ($\mathcal{M} + \mathcal{T}$) and backtrack at the end to fit exactly 30 frames.

3.2 GuardiolAI: A Graph-Unified Encoder–Decoder

GuardiolAI (short for **G**raph-**U**nified **A**rchitecture for **D**ynamic **I**nference and **O**ptimization in **L**earning with **A**rtificial **I**ntelligence), is a VAE–based [14] graph encoder–decoder for player-trajectory prediction (Fig. 1). We cast each possession sequence as a unified graph—nodes are player–frames, edges capture both spatial proximity and temporal succession—and train GuardiolAI to compress the observed M input frames into a latent representation before reconstructing the next T frames.

In our design, the encoder processes only the past M frames' node and edge features through stacked GATv2 layers, producing per-node parameters μ and σ.

[9] https://pytorch-geometric.readthedocs.io/en/latest/

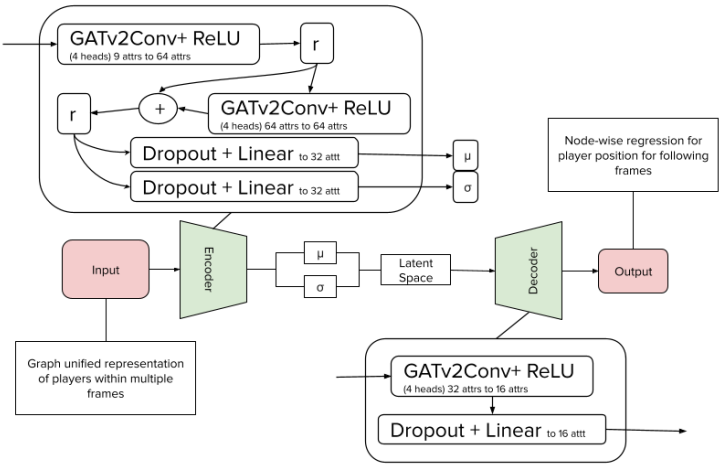


Fig. 1. Representation of the proposed architecture.

A latent sample z drawn from $\mathcal{N}(\mu, \sigma^2)$ then drives the decoder, which mirrors the encoder's GATv2 stack to predict the future positions. Crucially, the decoder is driven solely by z (and its own previous outputs when rolling forward), so all temporal and spatial context is captured within z and no past frames are re-introduced at decoding time.

GATv2 is particularly well-suited for this task due to its ability to capture node dependencies flexibly while incorporating edge features, which allows us to take advantage of our data design by assessing the model to balance weights across both spatial and temporal dimensions. Specifically, each GATv2 layer aggregates: *(i)* current-frame neighbors (spatial context); *(ii)* self-links from prior frames (temporal continuity); and *(iii)* indirect cross-frame context via stacked layers.

This layered propagation mechanism ensures that a player's representation in the latent space is informed not only by their immediate spatial surroundings but also by the influence of prior movements, leading to more coherent and temporally smooth predictions when decoding as well.

Reconstruction and Regularization The loss function employed in this work is designed to balance the trade-off between reconstruction accuracy and latent space regularization in a variational autoencoder framework. The total loss, denoted as $\mathcal{T}$ consists of two primary components: the reconstruction loss and the regularization loss.

The reconstruction loss, $\mathcal{T}_{\text{rec}}$, quantifies the discrepancy between the predicted output $\hat{y}$ and the original output y, ensuring that the model generates accurate predictions. Specifically, the reconstruction loss is computed using the Average Displacement Error (ADE) metric, which measures the Euclidean distance

between predicted and ground-truth trajectories across all time steps. Given a set of predicted positions $\hat{y}_t$ and ground-truth positions y_t for a sequence of length $1 + \mathcal{T}$, where $\mathcal{T}$ denotes the number of output frames plus the last frame of the input, for $\mathcal{N}$ players, ADE is defined as $\text{ADE} = \frac{1}{\mathcal{N}} \frac{1}{\mathcal{T}+1} \sum_{p=1}^{\mathcal{N}} \sum_{t=1}^{\mathcal{T}+1} \|\hat{x}_t^p - x_t^p\|_2$.

This metric provides a comprehensive evaluation of the model's prediction accuracy by averaging displacement errors across the entire trajectory, thereby penalizing deviations from the expected trajectory over time. To ensure stable optimization together with the regularizations, the reconstruction loss is normalized using a dynamic scaling factor γ, which is adaptively updated during training to reflect the minimum observed reconstruction error, leading to the formulation $\mathcal{L}\text{rec}\prime = \frac{\mathcal{L}_{\text{rec}}}{\gamma} + \log(\gamma)$, in which γ is updated iteratively as $\gamma = \min(\gamma, \mathcal{L}_{\text{rec}})$ [4].

The regularization loss, $\mathcal{L}\text{reg} = D\text{KL}\big(q(z|x) \parallel p(z)\big)$, is based on the Kullback-Leibler (KL) divergence, which penalizes deviations of the latent variable distribution $q(z|x)$ from a predefined prior $p(z)$. This regularization enforces a structure latent space by encouraging the approximate posterior $q(z|x)$ to align with a known prior distribution, in our work, a standard Gaussian $\mathcal{N}(0, 1)$. By doing so, the model learns to map similar inputs to nearby points in latent space, ensuring smoothness and continuity in the learned representations. Therefore, during inference, when sampling from the prior $p(z)$, the model is more likely to generate diverse and realistic samples that align with the data distribution seen in the training phase. Without proper regularization, the latent space may become too sparse, leading to unreal sample generation.

The final loss, referred to as the Dynamic Generative Loss (DGL) [4], is then formulated by combining scaled reconstruction and regularization:

$$\mathcal{L} = \frac{\mathcal{L}\text{rec}}{\gamma} + \log(\gamma) + \beta \cdot \mathcal{L}\text{reg} \tag{1}$$

3.3 Experiments

The dataset used was divided into train, test and validation, in a rate of 0.8, 0.1, 0.1, respectively. We used train and validation in the experiments and the test set to evaluate the model. We find the best parameters using optuna[10] framework, which led to the architecture having 2 layers with 4 heads of size 64 each in the encoder, that outputs embeddings of size 32, and having 1 layer with 4 heads of size 32 each in the decoder. Both encoder and decoder have a dropout of 0.1.

We perform a Learning Rate Range Test by exponentially increasing the learning rate from 0.000001 to 1 over a single epoch and selected the rate at which the training loss reached its minimum before diverging (0.01) [18]. Then, using that value as max_lr, we applied a OneCycleLR schedule with dividing factor = 25, final dividing factor = 0.0001, and cosine annealing to ramp up and down smoothly over the 20 training epochs. AdamW optimizer was employed as the learning rate optimization algorithm, with initial learning rate being 0.0004 1e-2/25 and having weight decay = 0.0001.

[10] https://optuna.org

3.4 Forecasting Schemes and Evaluation Metrics

Metrics. We report *Average Displacement Error* (ADE), *Mean Squared Error* (MSE) and *Final Displacement Error* (FDE), all in meters. A naïve **constant velocity** extrapolator—each player's last observed velocity projected forward—serves as our primary baseline.

Why re-anchoring? The raw GuardiolAI decoder predicts future frames in absolute pitch coordinates. Because the last observed frame f_M is fed back into the decoder, small sensor noise at f_M becomes amplified, producing a visible error spike in the first two predicted frames (see Fig. 3). To suppress this artefact we apply a re-anchoring transformation: for each possession we translate every predicted trajectory by an offset vector Δ computed as follows:

1. Select anchor frame:
   - f_{M+1} for `hybrid_first_frame`;
   - f_{M+2} for `hybrid_second_frame`.
2. Reference point: extrapolate the baseline from f_M to the anchor time, yielding $\hat{\mathbf{x}}_p^{\text{constant-velocity}}$.
3. Offset: $\Delta = \hat{\mathbf{x}}_p^{\text{constant-velocity}} - \hat{\mathbf{x}}_p^{\text{raw}}$.
4. Shift: add Δ to every future-frame prediction for every player.

Forecasting schemes. We therefore evaluate four variants that differ only in how they handle the first two predicted frames and the re-anchoring step:

- **raw**: keep the repeated frame (f_M appears twice) exactly as decoded.
- **exclude_repeated**: drop the repeated frame; no re-anchoring.
- **hybrid_first_frame**: drop the repeat, then re-anchor using the constant velocity estimate at f_{M+1}.
- **hybrid_second_frame**: use the constant velocity path for the first two unseen frames and re-anchor the rest at f_{M+2}.

This suite isolates the benefit of the rigid shift while keeping the core model unchanged.

4 Experimental Results

The GuardiolAI variants show clear strengths and weaknesses relative to the constant velocity baseline (Table 1). While the baseline still achieves the lowest ADE (2.01 m) and MSE (12.58 $\mathbf{m^2}$), all GuardiolAI modes improve on final-step accuracy: both the raw and exclude_first_frame predictions cut FDE from 4.38 m to 4.10 m (a 6.4 % reduction). Re-anchoring further closes the gap in full-trajectory metrics. The hybrid_first_frame scheme lowers ADE to 3.23 m and MSE to 18.39 $\mathbf{m^2}$, and the hybrid_second_frame setting reduces ADE to 2.66 m (35 % lower than 4.09 m and 0.65 m away from baseline) and MSE to 15.20 $\mathbf{m^2}$ (36 % lower than 23.57 $\mathbf{m^2}$). Although these still trail the baseline, they

Table 1. ADE, MSE, and FDE (in meters) for the constant velocity baseline and GuardiolAI under four prediction modes. Note: 'raw' an extra frame.

Model	ADE	MSE	FDE
Constant Velocity	**2.02**	**12.58**	4.38
GuardiolAI (raw)	4.12	23.88	**4.10**
GuardiolAI (exclude_first_frame)	4.09	23.57	**4.10**
GuardiolAI (hybrid_first_frame)	3.23	18.39	5.55
GuardiolAI (hybrid_second_frame)	2.67	15.20	4.51

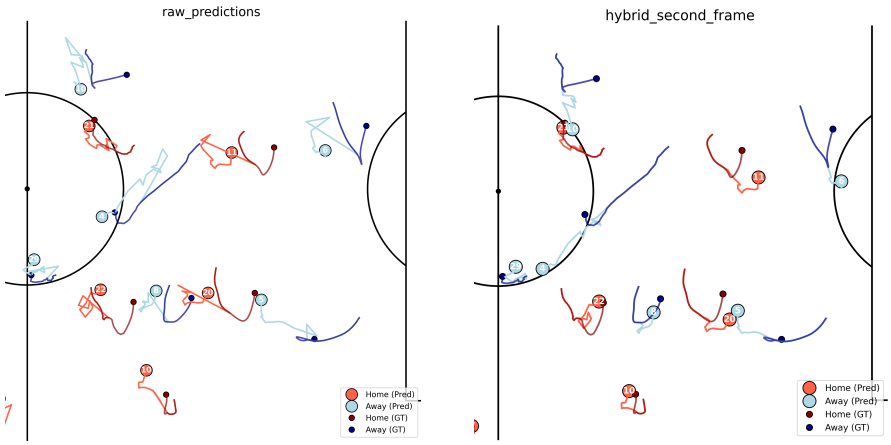


Fig. 2. Sample of prediction made with GuardiolAI in raw (left) and hybrid_second_frame (right) modes.

demonstrate that simple frame-anchoring strategies can substantially improve both average and squared-error metrics.

Figure 3 shows per-frame error with respect to the actual player position in that frame. The raw predictions spike on the first two frames – reflecting high jitter—then level off. When we reanchor (either first or second frame), the error at the whole trajectory drops significantly and it starts to have the same constant behavior as the constant velocity error. Considering the error difference between frames 17 and 28, it is remarkable that the error associated with Constant Velocity grows faster than the error associated with GuardiolAI configurations. In particular, velocity_baseline error increased 460%, while hybrid_second_frame increased 369%. Figure 2 shows a prediction sample for the hybrid_second_frame strategy.

Figure 4 shows how attention is distributed between spatial and temporal edges across the network. Given that each node's hidden state is updated by

$$h_u^{(t)} = \phi\Big(h_u^{(t-1)}, \bigoplus_{v \in N_u} a\big(h_u^{(t-1)}, h_v^{(t-1)}, e_{vu}, g\big)\, \psi\big(h_v^{(t-1)}\big)\Big).$$

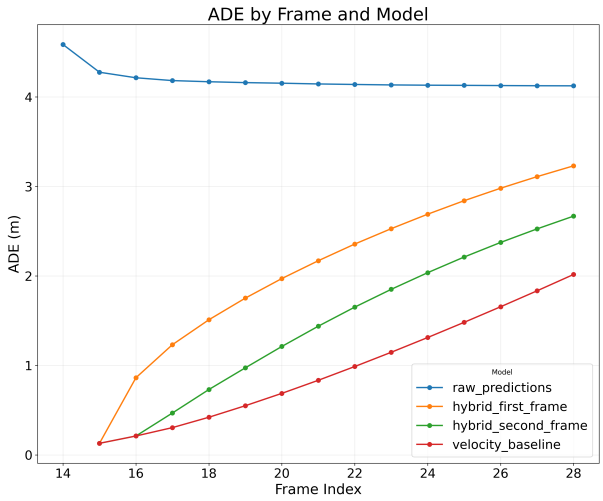


Fig. 3. Per-frame cumulative ADE for raw, reanchored predictions and constant velocity

where $a(\cdot)$ computes the attention weight between node u and its neighbor v based on their previous hidden states $h_u^{(t-1)}, h_v^{(t-1)}$, the edge features e_{vu}, and global context g, while $\psi(\cdot)$ transforms the neighbor's features and $\phi(\cdot)$ fuses the aggregated message with the node's prior state [5]. The values in the figure are the attention score per head per layer, indicating how much each neighbor type contributes to the node update. In the encoder, spatial edges carry more weight in the first layer, but in a low magnitude. Temporal edges become dominant by the second layer and remain overwhelmingly dominant throughout the decoder. This pattern confirms that, while inter-player relationships are encoded, the model still relies primarily on each player's own prior position to predict their next move.

5 Conclusion and Future Work

In this work, we explore how to represent a possession sequence in football using graphs that unify spatial and temporal dimensions simultaneously. A basic set of node features was defined, which can be extended to incorporate additional attributes. We propose two types of edges, which are, in practice, represented by two sets of edge features: *spatial* edge features that capture player relationships within the same timestamp; and *temporal* edge features that model relationships between the same player across different timestamps.

Using this representation, we designed and trained a state-of-the-art architecture, GuardiolAI, to predict player movements on the field, a task referred to as Movement Prediction (MP). The attention mechanisms introduced by GATv2

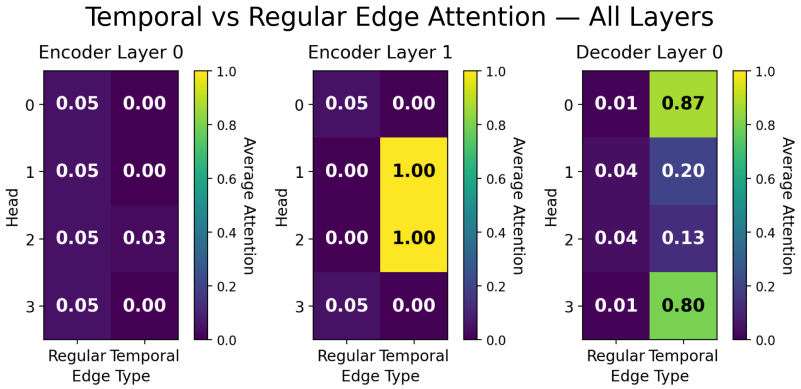


Fig. 4. Attention score per head per layer, indicating how much each neighbor type contributes to the node update and revealing the shift in focus from spatial relationships in early layers to temporal edges, with higher values, in later layers.

[5] empower the model with the capability to predict agent trajectories using dynamically assigned weights to nodes, while also giving some interpretability to results. Future work will explore alternative model architectures – such as integrating graph neural networks with temporal learning, employing transformer-based representations, and latent diffusion models – to benchmark different approaches for movement prediction. The goal is to find solutions to the absence of temporal edges in the first frame, which affected the predictions, as the experimental results demonstrated.

The VAE-based encoder-decoder architecture facilitates the exploration of latent space representations, an area we intend to investigate further in future studies. Additionally, we aim to extend the movement prediction framework to a Conditional Movement Prediction (CMP) approach by adapting the VAE structure into a CVAE [19], which would allow the model to condition predictions based on prior information.

Acknowledgements This work was partially supported by CNPq, CAPES, FAPEMIG and by project IAIA - INCT on AI.

References

1. Alcorn, M.A., Nguyen, A.: baller2vec++: A look-ahead multi-entity transformer for modeling coordinated agents. Pre-print (2021)
2. Alcorn, M.A., Nguyen, A.: baller2vec: A multi-entity transformer for multi-agent spatiotemporal modeling. Pre-print (2021)
3. Anzer1, G., Bauer, P., Brefeld, U., Fassmeyer, D.: Detection of tactical patterns using semi-supervised graph neural networks. MIT Sloan (2022),
4. Asperti, A., Trentin, M.: Balancing reconstruction error and kullback-leibler divergence in variational autoencoders. IEEE Access **8**, 199440–199448 (2020). https://doi.org/10.1109/ACCESS.2020.3034828

5. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? (2021), https://arxiv.org/abs/2105.14491
6. Capellera, G., Ferraz, L., Antonio Rubio, A.A., Moreno-Noguer, F.: Footbots: A transformer-based architecture for motion prediction in soccer. IEEE ICIP (2024), https://arxiv.org/abs/2406.19852
7. Everett, G., Beal, R.J., Matthews, T., Early, J., Norman, T.J., Ramchurn, S.D.: Inferring player location in sports matches: Multi-agent spatial imputation from limited observations (2023), https://arxiv.org/abs/2302.06569
8. Fassmeyer1, D., Anzer, G., Bauer, P., Brefeld, U.: Toward automatically labeling situations in soccer. Front. Sports Act. Living (2021), https://www.frontiersin.org/journals/sports-and-active-living/articles/10.3389/fspor.2021.725431/full
9. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks (2014), https://arxiv.org/abs/1406.2661
10. Guo, S., Lin, Y., Feng, N., Song, C., Wan, H.: Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19) (2019), https://ojs.aaai.org/index.php/AAAI/article/view/3881
11. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: beta-vae: Learning basic visual concepts with a constrained variational framework. ICLR (2017), https://openreview.net/forum?id=Sy2fzU9gl
12. Hoang, L., Carr, P., Yue, Y., Lucey, P.: Data-driven ghosting using deep imitation learning. MIT Sloan Conference 2016 (03 2016), www.sloansportsconference.com/research-papers/data-driven-ghosting-using-deep-imitation-learning
13. Hughes, H., Horton, M., Wei, F., Stokl, M., Gammulle, H., Fookes, C., Sridharan, S., Pedagadi, S., Lucey, P.: Approaching in-venue quality tracking from broadcast video using generative ai (2024),
14. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2013), https://arxiv.org/abs/1312.6114
15. Sahasrabudhe, A., Bekkers, J.: A graph neural network deep-dive into successful counterattacks. MIT Sloan (2023)
16. Salzmann, T., Ivanovic, B., Chakravarty, P., Pavone, M.: Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. ECCV (2020), https://link.springer.com/chapter/10.1007/978-3-030-58523-5$_4$0
17. Shirozu, Y.: What is the transition between set pieces and open play? (2022),
18. Smith, L.N., Topin, N.: Super-convergence: Very fast training of neural networks using large learning rates. arXiv (2017), https://arxiv.org/pdf/1708.07120
19. Sohn, K., Yan, X., Lee, H.: Learning structured output representation using deep conditional generative models. Advances in Neural Information Processing Systems (2015), https://dl.acm.org/doi/10.5555/2969442.2969628
20. Stöckl, M., Seidl, T., Marley, D., Power, P.: Making offensive play predictable - using a graph convolutional network to understand defensive performance in soccer. MIT Sloan Conference 2021 https://www.statsperform.com/wp-content/uploads/2021/04/Making-Offensive-Play-Predictable.pdf
21. Syed1, A., Morris, B.T.: Semantic scene upgrades for trajectory prediction. Machine Vision and Applications (2023), https://doi.org/10.1007/s00138-022-01357-z
22. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2017), https://arxiv.org/abs/1706.03762

23. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks (2017), https://arxiv.org/abs/1710.10903
24. Walker, J., Marino, K., Gupta, A., Hebert, M.: The pose knows: Video forecasting by generating pose futures. Proceedings of the IEEE international con- ference on computer vision (2017), https://arxiv.org/pdf/1705.00053
25. Wang, Z., Veličković, P., Hennes, D., Tomašev, N., Prince, L., Kaisers, M., Bachrach, Y., Elie, R., Wenliang, L.K., Piccinini, F., et al.: Tacticai: an ai assistant for football tactics. Nature Communications **15**(1), 1–13 (2024)
26. Yeh, R.A., Schwing, A.G., Huang, J., Murphy, K.: Diverse generation for multi-agent sports games. IEEE/CVF (2019)
27. Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., Li, H.: T-gcn: A temporal graph convolutional network for traffic prediction. IEEE Transactions on Intelligent Transportation Systems-2019 (2018), https://arxiv.org/pdf/1811.05320