Neural Processing Letters (2024) 56:66
https://doi.org/10.1007/s11063-024-11532-0

About Latent Roles in Forecasting Players in Team Sports

Luca Scofano[1] · Alessio Sampieri[1] · Giuseppe Re[2] · Matteo Almanza[2] · Alessandro Panconesi[2] · Fabio Galasso[2]

Accepted: 8 January 2024

Abstract
Forecasting players in sports has grown in popularity due to the potential for a tactical advantage and the applicability of such research to multi-agent interaction systems. Team sports contain a significant social component that influences interactions between teammates and opponents. However, it still needs to be fully exploited. In this work, we hypothesize that each participant has a specific function in each action and that role-based interaction is critical for predicting players' future moves. We create *RolFor*, a novel end-to-end model for Role-based Forecasting. RolFor uses a new module we developed called Ordering Neural Networks (OrderNN) to permute the order of the players such that each player is assigned to a latent role. The latent role is then modeled with a RoleGCN. Thanks to its graph representation, it provides a fully learnable adjacency matrix that captures the relationships between roles and is subsequently used to forecast the players' future trajectories. Extensive experiments on a challenging NBA basketball dataset back up the importance of roles and justify our goal of modeling them using optimizable models. When an oracle provides roles, the proposed RolFor compares favorably to the current state-of-the-art (it ranks first in terms of ADE and second in terms of FDE errors). However, training the end-to-end RolFor incurs the issues of differentiability of permutation methods, which we experimentally review. Finally, this work restates differentiable ranking as a difficult open problem and its great potential in conjunction with graph-based interaction models.

Keywords Multi-agent trajectory forecasting · Graph neural networks · Sport prediction · Differentiable ranking · Interaction pattern processing

1 Introduction

Recent advances in visual recognition and sequence modeling have enabled novel objectives in athletic performance and sport analytics [1–3]. One novel and challenging task is the multi-agent trajectory forecasting (See Fig. 1) of the players as a result of their observed current

✉ Luca Scofano
luca.scofano@uniroma1.it

[1] Department of Computer, Control and Management Engineering, Sapienza University, Rome, Italy

[2] Department of Computer Science, Sapienza University, Rome, Italy

Published online: 23 February 2024

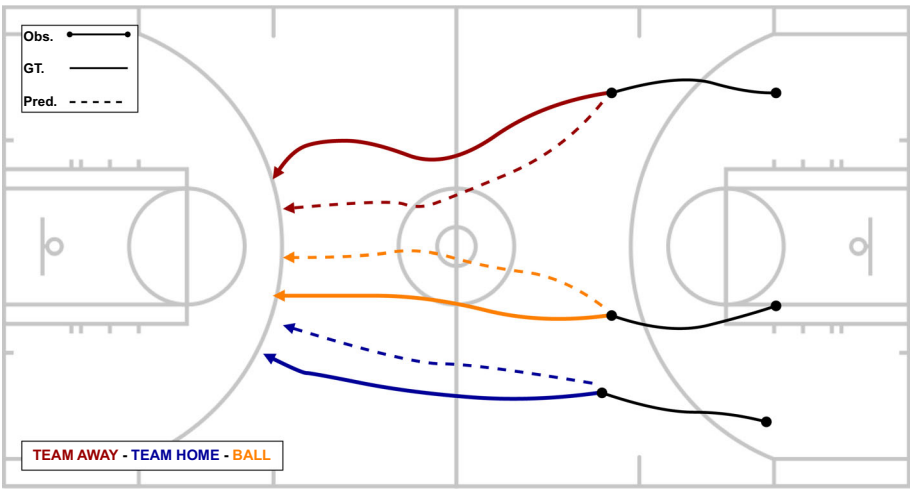

Fig. 1 Example of multi-agent trajectory forecasting. We only plot one player for each team and the basketball for readability reasons

motion [4, 5]. The difficulty is due to tactics, tight interaction of team players, the antagonist behavior of opponents, and the role assigned to each player in each action. Traditional trajectory forecasting techniques [6–10] fall short in performance due to their general formulations and lack of sport-specific dynamics. Furthermore, trajectory forecasting methods must deal with the variable numbers of people in each scene (usually absent in games) and do not consider the presence of two opposing teams, the ball, or the finality in the given sport (e.g. scoring). Most recent literature [4, 5] has started to address some of these objectives, but, to our knowledge, none has modeled the role of players for specific actions.

We propose RoleFor, a novel graph-based encoder-decoder model that performs a robust prediction of the players' future trajectory, utilizing roles to comprehend their interactions. The players' positions and movements on the court often follow pre-defined schemes, so we assume that each player may be assigned a specific role. By proposing a role-based ordering of nodes in the graph, it is possible to establish a player order and learn specific role-specific relationships.

The current best performers in-game forecasting [4, 10] are based on graph convolutional networks (GCN) [11], but they do not consider roles. On the contrary, we model latent roles as nodes in the graph. Our RoleFor model is composed of an ordering and a relational module. The former is an Ordering Network, which identifies latent roles and orders players according to them—we use a well-known sorting approximation [12] to order the latent projections of the players. In the latter, the game dynamics and trajectories are modeled using RoleGCN, based on [13] where the nodes are the newly assigned roles, and the edges are their relations. The adjacency matrix is learned, and each entry corresponds to learning the role-based player interaction.

We assume roles exist, and many characteristics could dictate them—e.g., marking the opponent, possessing the ball, and identifying the attacking and defending teams. However, we assume no prior knowledge about roles. Our goal is to learn latent roles with an end-to-end algorithm, only considering the future trajectory of all players. To test our intuition about roles, we pre-processed the basketball dataset by assigning roles based on different methods (Table 2) and using those in our RoleGCN. We produce SOTA results, confirming

that finding good roles can improve model performances. Nevertheless, we found that current differentiable ordering methods face some limitations of backpropagation when inserted in complex models. In summary, our contributions are:

- We experimentally demonstrate that leveraging roles yields SoTA in trajectory forecasting.
- We propose an Order Neural Network module that creates a latent representation of the player's coordinates and orders them accordingly.
- We build a RoleGCN that learns the relations among roles.
- We empirically demonstrate that the current differentiable ordering approaches have some difficulties with backpropagation—enabling little to no gradients to flow through—when dealing with complex models.

2 Related Work

2.1 Trajectory Forecasting

The forecasting of pedestrian movement has been studied to deal with realistic crowd simulation [14] or to improve vehicle collision avoidance [15]; it was also used to enhance the accuracy of tracking systems [16–18] and to study the intentions of individuals or groups of people [19, 20]. Different models have been proposed to predict such trajectories, like Long Short-Term Memory (LSTM) networks [21] with shared hidden states [6], multi-modal Generative Adversarial Networks (GANs) [9], or inverse reinforcement learning [22]. This group forecasting scenario resembles Game Forecasting, where it is necessary to model the movements of two opposing teams.

2.2 Game Forecasting

Associations such as National Basketball League or the English Premier League have used sophisticated tracking systems that allow teams to gain insight into each game [23]. Variational Autoencoders (VAEs) were used to model real-world basketball actions, showing that the offensive player trajectories are less predictable than the defense [24]. LSTM [25] were employed to predict near-optimal defensive positions for soccer and basketball, respectively, as for predicting the player's movements during the game [5]. Variants of VAEs have also been used [26] to generate trajectories for NBA players. NBA player trajectory forecasting was also studied in [27] and [28], proposing a deep generative model based on VAE, LSTM, and RNN [5, 21, 29] and trained with weak supervision to predict trajectories for an entire team. Nonetheless, we did not encounter work estimating specific latent roles and learning the player interaction on those bases.

2.3 GCN-Based Forecasting

Adopting a graph structure makes it possible to encode information and quantify shared information between nodes. SoA in pose forecasting learns specific terms for the specific joint-to-joint relation [13, 30]. Graphs are also widely used in trajectory forecasting and can be considered fully connected [4], sparse or weighted. These structures distinctly model the interrelationships between nodes, and their combination can be a crucial factor. Also,

Graph attention layers (GAT) are widely used in trajectory forecasting [8, 31] to learn the inter-player dependencies. To model role-based interaction, we use the SoA pose forecasting model [13]. Pose forecasting is relevant since it considers the fixed node cardinalities and the learned interactions. However, players from various matches and teams do not have a fixed order, which is not an issue with pose forecasting. This encourages us to learn and re-order the players based on hidden roles.

2.4 Differentiable Ranking

Sorting and Ranking are two popular operations in information retrieval that, in our case, can be useful in identifying the role of players. When used in composition with other functions, sorting induces non-convexity, rendering model parameter optimization difficult. On the other hand, the ranking operation outputs the positions, or ranks, of the input values in the sorted vector. As a piece-wise constant function, the computation of gradients is way more complex, preventing gradient backpropagation. Several recent works [12, 32] provide an approximation of the above operations to be used in a learnable framework.

3 Methodology

This section formally defines the problem and explains our strategy to tackle it, focusing on the role assignment and encoding methods. First, we briefly explain how the Role-based Forecasting model (RoleFor) performs latent mapping, role assignment, and trajectory prediction. We also focus on the main components: the Order Neural Network (OrderNN), which handles the ordering task, and the RoleGCN, which facilitates the learning process of relationships between roles in a game.

3.1 Problem Formalization

We target to predict the future trajectory of all players, given the observed positions at past time frames. We denote the players by 2D vectors $x_{p,t}$ representing player p at time t. The position of all players at time t are aggregated into a matrix of 2D coordinates $X_t \in \mathbb{R}^{2\times P}$. Motion history of players is denoted by the tensor $X_{in} = [X_1, X_2, \ldots, X_T]$, which is constructed out of the matrices X_t for frames $t = 1, \ldots, T$. The goal is to predict the future K players positions $X_{out} = [X_{T+1}, \ldots, X_{T+K}]$.

3.2 Role-Based Forecasting Model (RolFor)

RoleFor uses two main components, the first one being the OrderNN (Section 3.2.1), which orders players according to their latent roles. We postulate the existence of latent roles that when learned in an end-to-end architecture yield the best trajectory forecasting performance. From the OrderNN, we will consider R, the role vector, instead of P, the position vector. Notice that R and P have equal dimensions. The graph is now defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the nodes indicate the roles of each player and the edges capture the interaction among roles during the game. The graph $\mathcal{G}$ has $|\mathcal{V}| = T \times R$ nodes, which represent all R roles across T observed time frames. Edges in $\mathcal{E}$ are represented by a Spatio-Temporal adjacency matrix $A^{st} \in \mathbb{R}^{RT\times RT}$, relating the interactions of all roles at all times. Note that A^{st} is learned,

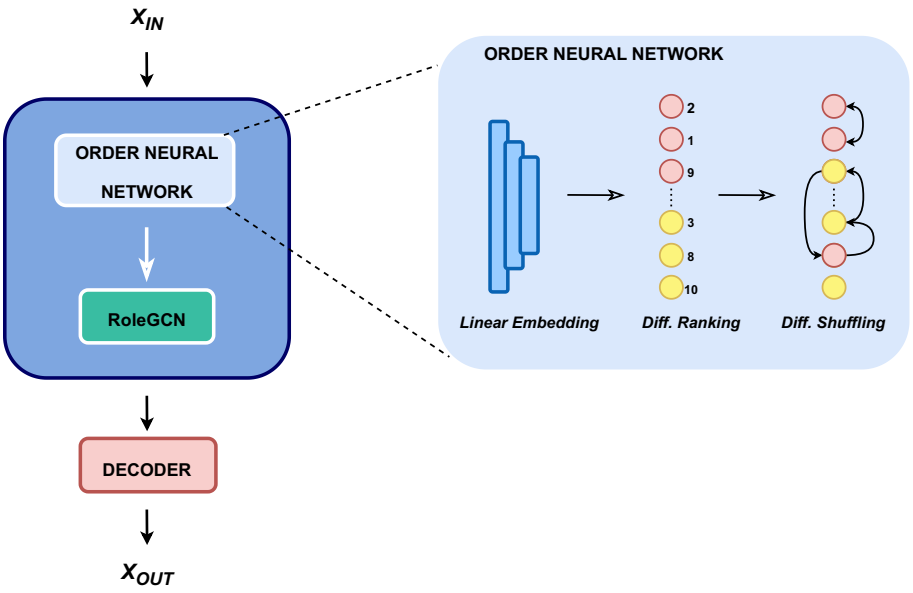

Fig. 2 Architecture of RoleFor and a zoom into Order Neural Network

i.e., the model learns how players with different roles interact by learning how latent roles interact over time.

3.2.1 Order Neural Network

The Order Neural Network (Fig. 2) takes in input the initial coordinates X_{in} and maps them into a latent space. Additionally, it orders the latent vector into optimal roles X_{role_in}, thanks to the use of a differentiable ranking method [12], which has the same dimensionality of X_{in}. Note that roles get the corresponding position coordinates over subsequent time frames, so each role is now characterized by a spatio-temporal trajectory. A straightforward example of a role assignment involves sorting players in ascending order based on their Euclidean distance from the ball. This method is also used as a valuable proxy task, which we use for ablation studies (see Sect. 4 Table 2). However, since RolFor is trained end-to-end, OrderNN is free to learn the ideal ordering that yields the best forecasting performance.

3.3 The Differentiable Ranking Method

SoftRank, [12] is a recent differentiable implementation of the classic sorting and ranking algorithm, empirically shown to achieve accurate approximation for both tasks. It is designed by constructing differentiable operators as projections onto the permutahedron, i.e., the convex hull of permutations, and using a reduction to isotonic optimization. The key takeaway of the method is to cast sorting and ranking operations as linear programs over the permutahedron. More precisely, it formulates the argsort and ranking operations as optimization problems over the set of permutations Σ. SoftRank also relies on a regularization parameter ε, which creates a trade-off between the differentiability of the algorithm and the optimum's accuracy. The greater the regularization factor ($\varepsilon \to \infty$), the further the approximation will

be from the permutation vertices, and the smoother the loss function gradient will be. And vice versa, by picking an $\varepsilon \to 0$, the algorithm will yield more accurate permutations with a lower degree of differentiability. After learning the ranking, we order the players according to it by employing a differentiable re-shuffling module. The outputs of SoftRank are noted as $\{s_i\}_{i=1}^{n}$ where n is the number of rankings considered. At this point, we use a so-called *base* matrix $\mathbf{B}$ with the number of rows and columns equal to the number of rankings. $\mathbf{B}$ will be used to store the real rankings $\{p_i\}_{i=0}^{n}$. We then compute a $\{\Delta_i\}_{i=1}^{n}$ matrix, which represents $\Delta_i = p_i - s_i$ for each position $\{p_i\}_{i=0}^{n}$. The matrix Δ is used as the input as a rescaling function. The re-shuffle process is a weighted combination: it yields a real shuffling when the approximated rankings are integer and a differentiable shuffling instead when the ranking is fractional. $M_i = e^{\left(\frac{-\Delta}{scale}\right)^2}$ can be considered an array of weights for each position, with values closer to 1 being the predicted positions of each player. Finally, this will be used to recall the initial coordinates in an ordered manner:

$$P_i = \begin{cases} x_i' = \sum_{j=1}^{n} M_j \cdot x_j \\ y_i' = \sum_{j=1}^{n} M_j \cdot y_j \end{cases} \tag{1}$$

3.3.1 RoleGCN

Once the latent roles are inferred, the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represents each node $i \in \mathcal{V}$ as the player's role while the edges $(i, r) \in \mathcal{E}$ connect all the roles and describe their mutual interaction. RoleGCN (Fig. 2) will capture the underlying graph's relationships, both between different nodes on the court in the same time frame and between one node and itself over different time-frames. GCN [11] is a graph-based operation that works with nodes and edges. For nodes, it aims to learn an embedding containing information about the node itself and its neighborhood for each node in the graph. Thus, the learned adjacency matrices yield a quantitative description of the interplay among roles. The space-time cross-talk is realized by factoring the space-time adjacency matrix (as in [13]) into the product of separate spatial and temporal adjacency matrices $A^{st} = A^S A^t$. A separable space-time graph convolutional layer l is written as follows:

$$H^{(l+1)} = \sigma(A^{s-(l)} A^{t-(l)} \mathcal{H}^{(l)} W^{(l)}) \tag{2}$$

It is similar to a classic GCN convolutional layer, where $A^{s-(l)} A^{t-(l)}$ is the factorized matrix $A^{st-(l)}$ of a GCN [11] layer. The critical difference is better efficiency and allows full learnability of the former.

3.3.2 Decoder

First, we de-shuffle the permuted roles according to the inverse of $\mathbf{B}$ to return to the original coordinates' position. The decoding is done with multiple temporal convolutional (TCN) layers [33] used to predict the following frames. We adopt TCN due to its performance and robustness.

4 Experimental Evaluation

In this section, we introduce the NBA benchmark dataset and metrics, the trajectory forecasting results and investigate why learning E2E roles is challenging.

4.1 Dataset

For our experiments, we use NBA SportVU [24]. It contains players and ball trajectories for 631 games from the 2015–2016 NBA season. Similar to previous work [26], we focus on just two teams and consider all their games. We obtain a dataset of 95, 002, 12-second sequences of players and ball overhead-view trajectories from 1247 games. Each sequence is sampled at 25 Hz, has the same team on offense for the entire duration and ends in a shot, turnover, or foul. As in [24], the data is randomly split into train, validation, and test sets with respectively 60, 708, 15, 244, and 19, 050 sequences.

4.2 Trajectory Forecasting Metrics

We use as metrics ADE (Average Displacement Error) and FDE (Final Displacement Error), as usual in literature [4–6, 9, 24]. They are used to measure the error of the whole trajectory sequence and the final endpoints for each player. Respectively:

$$ADE = \left\| \hat{T}_c - T_c \right\|_2^2 \tag{3}$$

$$FDE = \left\| \hat{E}_f - E_f \right\|_2^2 \tag{4}$$

Each observation has *five* frames, which correspond to 2.0 s in a basketball scenario. The goal is to forecast the successive *ten* frames (4.0 s). In Eq. 3, $\hat{T}_c$ represents the prediction for all future trajectories over the $c = 1, \ldots, 10$ subsequent frames, and T_c is the ground truth. The same nomenclature is used in Eq. 4, where E is the matrix for the endpoints and $c = 1$ since we are only considering the last frame.

4.3 Trajectory Forecasting Results

So, do roles exist, and does learning the role interaction yield state-of-the-art performance? We answer this question by considering the most straightforward ordering: Euclidean distance of players from the ball. In Table 1, we report state-of-the-art techniques compared to the RolFor model, with the Euclidean distance ordering of players from the ball. [4] proposes multiple predictions via latent interaction graphs among multiple interactive agents. [9], similarly, is also a multi-modal model incorporating the social aspects of the players as well. [8] is based on a sequence-to-sequence architecture to predict the future trajectories of players. Lastly, [10] substitutes the need for aggregation by modeling the interactions as a graph. Similar to [30], it needs a pre-defined graph, allowing the leaning procedure only on the given edges. RolFor in Table 1 yields the SoA forecasting performance in terms of ADE, 5.55 ms, second best in terms of FDE, 9.99 ms. It sorts players according to their Euclidean distance from the ball, arranging them into a sequence of attackers (players detaining the ball in the considered action), alternating with defenders (not detaining the ball). Each attacker is followed by its marker, which RolFor considers the closest to it in terms of Euclidean distance. As for all other reported SoA algorithms, RolFor considers that the teams are known. Finally, "Oracular Permutation" means that RolFor uses distances at the last future step, i.e., step 10 in the future. In contrast, any other reported algorithm uses only the observed five frames. We will investigate this more thoroughly in the next section. A neural network can learn the Euclidean distance, and softRank [12] should be able to sort the players according to it. Replacing the hand-defined distance computation with a Neural Network should be as

Table 1 Comparison of our model with SoTA models

Model	ADE	FDE
EvolveGraph [4]	5.73	8.65
Social-STGCNN [10]	6.42	10.04
STGAT [8]	7.06	12.54
SGAN [9]	5.88	10.36
RolFor + Oracular Permutation	**5.55**	9.99

Table 2 Results for different types of ordering

Ball Dist.	Obs.	Future	Mark	ADE	FDE
–	–	–	–	6.34	11.5
✓	–	–	–	6.31	11.1
✓	✓	–	✓	6.16	11.28
✓	–	✓	✓	5.55	9.99

effective. We expect that a model with a sorting unit that learns sorting E2E in relation to the final forecasting goal should be capable of doing better than this, assuming all modules were effectively differentiable.

4.3.1 Further Experiments on Euclidean Ordering

We delve deeper into the results of RolFor in Table 1 and analyze the importance of each hand-defined Euclidean distance term in Table 2.

No Ordering Vs. Simple Ordering The first forecasting result in the table neglects the player ordering and learns interaction terms between players, arranged in random order. It yields 6.34/11.5 ADE/FDE meters errors. Simple ordering stands for arranging all players in a list, according to their distance from the ball, at the last (5th) observed frame. This uncomplicated ordering is only negligibly better than no order. A GCN model may deal with players in random order well and only benefits from ordering if it is informative.

Distance from the Ball and Marking Results in the third row of the Table 2 add marking to the ball distance ordering. Each player in the attacker team is matched with one from the defender team according to Euclidean distance. Performance improves in ADE, from 6.31 to 6.16 ms, and slightly degrades in terms of FDE, from 11.1 to 11.28. Overall All distances are computed at the last observed frame. Furthermore, all distances are plain Euclidean distances, which a simple Neural Network may replicate or improve with E2E learning.

Distance from the Ball and Marking at Future Frames The last row of Table 2 considers the furthest future frame position for all distance computations. It should be noted that the model makes no assumptions about future locations. Future information is simply utilized to place players in order. This motivates us to replace the hand-defined ordering with an E2E-trained module, which we will do in the following section.

4.4 End to End Model with Latent Roles

In this section, we leverage the full RolFor model, E2E trained. Here the first module, OrderNN, sorts players into their roles in the action, then the RoleGCN module reasons on their role-based interaction. Sorting into roles has benefited forecasting in Sec. 4.3.1.

Table 3 Different training configurations for RolFor

Model	Configuration	ADE	FDE
RolFor	*E2E*	12.12	15.02
RolFor	*E2E-finetune*	12.08	14.97
RolFor	*EuclDistEst*	7.50	12.58
RolFor	*Best non-or. dist.*	6.16	11.28

Table 4 Analysis of simulated errors in ordering

Model	ADE	FDE
Oracular ordering	5.55	9.99
Light swap	6.55	12.10
Light insert	6.55	12.10
Light swap + light insert	6.59	12.08
Heavy swap + heavy insert	6.71	12.25

Here we assume that roles are latent variables, which the OrderNN estimates, E2E, based on the best forecasting performance. In Table 3, we compare the hand-defined baseline (ball and marking distance on the last observed frame, scoring 6.16/11.28) against E2E model variants. *E2E* is learning to order, encode the role-role interaction, and forecast based on the encoder. This model is performing poorly at 12.12./15.02 ADE/FDE. Is this because the OrderNN is incapable of ordering, or is it because the OrderNN is not fully differentiable? Moreover, the *EuclDistEst* variant attempts to answer part of this question. Here we used a pre-trained Neural Network module to approximate the Euclidean distance based on the player's performance. We then use the pre-trained module to sort players according to the ball. If the Euclidean distance estimator model were perfect, performance would be 6.31/11.1 (ADE/FDE), cf. Table 2. *EuclDistEst* yields, however, 7.50/12.58. We attribute this mismatch to the residual errors in the Euclidean distance estimation, which, as it seems, matters. More surprisingly, *E2E-finetune* starts from the *EuclDistEst* variant, and it fine-tunes it, E2E. The error increases to 12.08/14.97, so the model neglects the initialization and reverts to the *E2E* performance. We attribute the discrepancy between *EuclDistEst* and *E2E* to the challenges in the SoftRank differentiability, as we further analyze in the next section.

4.4.1 Analysis of the Order Neural Network

Here we focus on confirming our claims on the issues of the differentiability of Softrank. We set to order the players according to their ascending distance from the ball, at a specific frame, given their 2D coordinates. It allows us to test the first RolFor module, OrderNN, in isolation, cf.5. In Table 5, we compare *OrderNN E2E* against *OrderNN EuclDistEst*. The first E2E trains the order of players and re-shuffles them. The second supervises the network by tasking it to learn the Euclidean distance between the players and the ball and then sort the distances according to SoftRank. We measure the ordering accuracy p_{ord} as the percentage of players the models place in the correct order. In other words, we reproduce the top-k classification experiment as [32]. The authors propose a loss for top-k classification between a ground truth class $ord \in [n]$ and a vector of soft ranks $\hat{ord} \in \mathbb{R}^n$, which is higher if the predicted soft ranks correctly place y in the top-k elements.

Table 5 OrderNN *E2E* against OrderNN *EuclDistEst* top-k accuracy

Model	top-k	Accuracy (%)
OrderNN *E2E*	10	1.00
OrderNN *EuclDistEst*	10	71.00
OrderNN *EuclDistEst*	5	77.00
OrderNN *EuclDistEst*	3	82.00
OrderNN *EuclDistEst*	1	92.00

Observe from Table 5 that learning Euclidean distances from 2D positions is an easier task for a deep neural network since SoftRank yields 71% at the *top-10* ordering accuracy p_{ord}. It is also interesting to notice that when changing the *top-k* ordering accuracy into 5, 3, 1 we get similar results to [12]. By contrast, learning the ordering E2E from the 2D coordinates yields surprisingly low performance. Note in the table that *OrderNN E2E* achieves a *top-10* ordering accuracy of only 1%.

4.5 Robustness of RolFor to Ordering Errors

How much does misordering impact forecasting? We measure ADE and FDE forecasting errors when randomly altering the order provided by our best performing oracle RolFor (5.55/9.99 ADE/FDE, Table 2). In more detail, we consider the swap of two players *Light Swap*, which can occur if the distance between them is relatively small. A more significant error can also occur, e.g., one role is not identified correctly and a player is inserted at the wrong position, making the whole order slip. We name this *Light Insert*. In Table 4, we consider the two potential sources of errors by randomly simulating one or both. The results are coherent with what we said previously Table 3, where the RolFor *EuclDistEst* has a *top-10* ordering accuracy of 71% yielding 7.50/12.58. At the same time, a Light Swap/Insert gives 6.55/12.10 in ADE/FDE and 80% *top-10* ordering accuracy. This last Table 4 highlights the importance of roles and their impact on the final trajectory accuracy.

5 Conclusions

Our goal was to show that roles and social relations in sports are quantifiable and can be effectively used to improve the current SoA models in game forecasting. We demonstrate that roles exist by testing different permutations over players. Then, we encode the player's coordinates into a latent space and use the encoding to find an optimal latent role ordering. The model employed to perform trajectory forecasting is called RolFor (Role Forecasting) and considers the input nodes of a graph indicating roles in a game. This single-graph framework favors the relation between roles and time, allowing better learning of the fully-trainable adjacency matrices for role-role and time-time interactions. The adoption of CNNs and the graph structure of the input allows the requirement of parameters to be only a fraction of the ones used in Transformers, GANs, and VAEs. Our observations emphasize the significant opportunity for future work to develop fully differentiable ordering modules to enable learning latent role-based interactions in graph-based models, also applicable to social networks and multi-agent systems.

Acknowledgements This work was partially supported by the MUR PNRR project FAIR (PE00000013).

Author Contributions LS: Conceptualization, Investigation, Project administration, Methodology, Writing, Visualization. AS: Data curation, Investigation, Methodology, Writing. GR: Data curation, Conceptualization, Formal analysis, Supervision, Writing-review. MA: Data curation, Conceptualization, Formal analysis, Writing. AP: Conceptualization, Writing-review, Supervision. FG: Conceptualization, Writing-review, Supervision.

Funding Open access funding provided by Università degli Studi di Roma La Sapienza within the CRUI-CARE Agreement.

Declarations

Conflict of interest All authors declare that they have no conflict of interest.

References

1. Rein R, Memmert D (2016) Big data and tactical analysis in elite soccer: future challenges and opportunities for sports science. SpringerPlus 5(1):1–13
2. Merhej C, Beal RJ, Matthews T, Ramchurn S (2021) What happened next? using deep learning to value defensive actions in football event-data. In: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, pp 3394–3403
3. Morgulev E, Azar OH, Lidor R (2018) Sports analytics and the big-data era. Int J Data Sci Anal 5(4):213–222
4. Li J, Yang F, Tomizuka M, Choi C (2020) Evolvegraph: multi-agent trajectory prediction with dynamic relational reasoning. https://doi.org/10.48550/arXiv.2003.13924
5. Hauri S, Djuric N, Radosavljevic V, Vucetic S (2021) Multi-modal trajectory prediction of nba players. In: Winter conference on applications of computer vision (WACV). https://doi.org/10.48550/arXiv.2008.07870
6. Alahi A, Goel K, Ramanathan V, Robicquet A, Fei-Fei L, Savarese S (2016) Social lstm: human trajectory prediction in crowded spaces. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 961–971. https://doi.org/10.1109/CVPR.2016.110
7. Giuliari F, Hasan I, Cristani M, Galasso F (2020) Transformer networks for trajectory forecasting. In: In the international conference on pattern recognition (ICPR). https://doi.org/10.48550/arXiv.2003.08111
8. Huang Y, Bi H, Li Z, Mao T, Wang Z (2019) Stgat: modeling spatial-temporal interactions for human trajectory prediction. In: ICCV
9. Gupta A, Johnson J, Fei-Fei L, Savarese S, Alahi A (2018) Social GAN: socially acceptable trajectories with generative adversarial networks. In: 2018 IEEE/CVF conference on computer vision and pattern recognition, pp 2255–2264. https://doi.org/10.1109/CVPR.2018.00240
10. Mohamed A, Qian K, Elhoseiny M, Claudel C (2020) Social-stgcnn: a social spatio-temporal graph convolutional neural network for human trajectory prediction. In: CVPR
11. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: ICLR
12. Blondel M, Teboul O, Berthet Q, Djolonga J (2020) Fast differentiable sorting and ranking. In: International conference on leadership and management (ICLM). https://doi.org/10.48550/arXiv.2002.08871
13. Sofianos T, Sampieri A, Franco L, Galasso F (2021) Space-time-separable graph convolutional network for pose forecasting. In: International conference on computer vision (ICCV). https://doi.org/10.48550/arXiv.2110.04573

14. Pelechano N, Allbeck JM, Badler NI (2007) Controlling individual agents in high-density crowd simulation. In: SCA '07
15. Bhattacharyya A, Fritz M, Schiele B (2018) Long-term on-board prediction of people in traffic scenes under uncertainty. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4194–4202
16. Choi W, Savarese S (2012) A unified framework for multi-target tracking and collective activity recognition, pp 215–230
17. Pellegrini S, Ess A, Gool LV (2010) Improving data association by joint modeling of pedestrian trajectories and groupings. In: ECCV
18. Yamaguchi K, Berg AC, Ortiz LE, Berg TL (2011) Who are you with and where are you going? In: CVPR 2011, pp 1345–1352. https://doi.org/10.1109/CVPR.2011.5995468
19. Lan T, Wang Y, Yang W, Robinovitch SN, Mori G (2012) Discriminative latent models for recognizing contextual group activities. EEE Trans Pattern Anal Mach Intell 34:1549–1562
20. Xie D, Shu T, Todorovic S, Zhu S-C (2018) Learning and inferring "dark matter" and predicting human intents and trajectories in videos, vol 40, pp 1639–1652. https://doi.org/10.1109/TPAMI.2017.2728788
21. Hochreiter S, Schmidhuber J (1997) Long short-term memory. In: Neural computation, pp 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
22. Kitani KM, Ziebart BD, Bagnell JA, Hebert M (2012) Activity forecasting. In: In European conference on computer vision ECCV, pp 201–214
23. Carling C, Bloomfield J, Nelsen L, Reilly T (2008) The role of motion analysis in elite soccer. Sports Med 38(10):839–862
24. Felsen P, Lucey P, Ganguly S (2018) Where will they go? Predicting finegrained adversarial multi-agent motion using conditional variational autoencoders. In: Proceedings of the European conference on computer vision (ECCV), pp 732–747
25. Seidl T, Cherukumudi A, Hartnett AT, Carr P, Lucey P (2010) Bhostgusters: realtime interactive play sketching with synthesized nba defenses. In: MIT Sloan sports analytics conference
26. Sun C, Karlsson P, Wu J, Tenenbaum J, Murphy K (2018) Stochastic prediction of multi-agent interactions from partial observations. https://doi.org/10.48550/arXiv.1902.09641
27. Zhan E, Zheng S, Yue Y, Sha L, Lucey P (2018) Generating multi-agent trajectories using programmatic weak supervision. https://doi.org/10.48550/arXiv.1803.07612
28. Zheng S, Yue Y, Lucey P (2017) Generating long-term trajectories using deep hierarchical networks. https://doi.org/10.48550/arXiv.1706.07138
29. Jain A, Zamir A, Savarese S, Saxena A (2016) Structural-rnn: Deep learning on spatio-temporal graph. In: The IEEE computer vision and pattern recognition (CVPR). https://doi.org/10.48550/arXiv.1511.05298
30. Yan S, Xiong Y, Lin D (2018) Spatial temporal graph convolutional networks for skeleton-based action recognition. In: AAAI
31. Li L, Yao J, Wenliang L, He T, Xiao T, Yan J, Wipf D, Zhang Z (2021) Grin: generative relation and intention network for multi-agent trajectory prediction. Adv Neural Inf Process Syst 34 (2021)
32. Cuturi M, Teboul O, Vert J-P (2019) Differentiable ranking and sorting using optimal transport. In: Wallach H, Larochelle H, Beygelzimer A, d' Alché-Buc F, Fox E, Garnett R (eds) Advances in Neural Information Processing Systems, vol 32. Curran Associates, Inc.
33. Holden D, Saito J, Komura T, Joyce T (2015) Learning motion manifolds with convolutional autoencoders. In: SIGGRAPH Asia. https://doi.org/10.1145/2820903.2820918