Trajectory Imputation in Multi-Agent Sports with Derivative-Accumulating Self-Ensemble

Han-Jun Choi[1][*] Hyunsung Kim[2,3][*], Minho Lee[4], Minchul Jeong[5],
Changjo Kim[3], Jinsung Yoon[3], and Sang-Ki Ko[6]✉

[1] KETI, Seongnam, South Korea
`hanjun_c@keti.re.kr`
[2] KAIST, Daejeon, South Korea
`hyunsung.kim@kaist.ac.kr`
[3] Fitogether Inc., Seoul, South Korea
`{hyunsung.kim,changjo.kim,jinsung.yoon}@fitogether.com`
[4] Saarland University, Saarbrücken, Germany
`minho.lee@uni-saarland.de`
[5] Weflo Inc., Seoul, South Korea
`mcjeong@weflo.ai`
[6] University of Seoul, Seoul, South Korea
`sangkiko@uos.ac.kr`

Abstract. Multi-agent trajectory data collected from domains such as team sports often suffer from missing values due to various factors. While many imputation methods have been proposed for spatiotemporal data, they are ill-suited for multi-agent sports, where player movements are highly dynamic and interactions evolve over time. To address these challenges, we propose MIDAS (**M**ulti-agent **I**mputer with **D**erivative-**A**ccumulating **S**elf-ensemble), a data-efficient framework that imputes multi-agent trajectories with high accuracy and physical plausibility. It jointly predicts positions, velocities, and accelerations via a Set Transformer-based neural network and refines them by recursively accumulating predicted velocity and acceleration values. These predictions are then combined using a learnable weighted ensemble to produce final imputed trajectories. Experiments on three sports datasets show that MIDAS significantly outperforms existing baselines, with particularly large margins in limited-data settings. We also demonstrate its utility in downstream tasks such as estimating total distance and pass success probability. The source code is available at
`https://github.com/gkswns95/midas.git`.

Keywords: Sports Analytics · Multi-Agent System · Trajectory Imputation · Deep Learning under Physical Constraints · Weighted Ensemble

1 Introduction

Many spatiotemporal domains, such as transportation, robotics, surveillance, and sports, handle multi-agent trajectory data. While advances in computer vision

---

[*] These authors contributed equally to the paper.

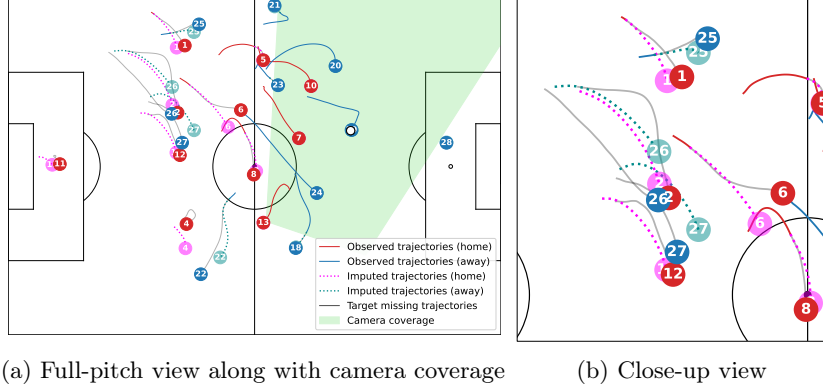(a) Full-pitch view along with camera coverage   (b) Close-up view

Fig. 1: Example of imputing unobserved player trajectories from tracking data obtained in a soccer broadcast.

and sensing technologies have facilitated large-scale trajectory data collection, acquiring complete data remains challenging due to various factors such as signal loss in wearable devices and limitations of the camera's field of view (Fig. 1). This prevalence of missing values calls for the development of effective imputation techniques that can accurately reconstruct missing trajectories.

Though various imputation methods have been proposed for spatiotemporal data, applying them to multi-agent sports remains challenging due to their dynamic nature. In particular, while many of them [1,5,16,17,18,24] have demonstrated their effectiveness in multi-sensor data, including traffic flow or air quality datasets, they do not account for dynamic interactions between agents. In such fixed-sensor networks, spatial relationships typically remain constant over time. In contrast, multi-agent domains such as team sports involve players with continuously changing positions, requiring a suitable architecture for explicitly modeling these dynamic inter-agent relationships while maintaining permutation-equivariance with respect to the agents.

Furthermore, player motion is governed by biomechanical constraints that impose physical limits on speed, acceleration, and directional change. To accurately model such constraints and generate physically plausible trajectories, imputation models must learn nuanced patterns in movement dynamics. This typically requires a large volume of high-quality training data. However, this poses a significant limitation in the sports domain, where player tracking data is often treated as confidential due to the competitive nature of professional leagues.

Addressing these challenges, this paper proposes MIDAS (**M**ulti-agent **I**mputer with **D**erivative-**A**ccumulating **S**elf-ensemble), a framework for multi-agent sports that imputes missing trajectories with high accuracy and data efficiency by explicitly enforcing the physical constraints that real player trajectories should satisfy. First, a neural network equipped with Set Transformers [13] and player-wise bidirectional LSTMs [9] predicts the positions, velocities, and accelerations of

missing players. In addition to this *initial prediction* (IP), it generates alternative estimates through *derivative-accumulating prediction* (DAP), which recursively accumulates the predicted velocity and acceleration values from the nearest observed positions in both forward and backward directions. Finally, the learnable soft voting mechanism combines these three predictions, namely IP and forward/backward DAPs, to produce the final imputed trajectories.

The proposed MIDAS overcomes the aforementioned challenges in two aspects. First, by employing the Set Transformer, it models dynamic inter-agent relationships while ensuring permutation-equivariance with respect to agents. Second, MIDAS jointly predicts positions, velocities, and accelerations, and integrates them through a derivative-accumulating self-ensemble mechanism that explicitly enforces their physical consistency. This design enables the model to capture the underlying patterns of player motion, leading to improved data efficiency.

Experiments across three sports datasets demonstrate that MIDAS consistently outperforms existing baselines in terms of both positional accuracy and physical plausibility of imputed trajectories, with particularly large margins in low-data settings. In addition, we showcase real-world applications of MIDAS, including the approximation of physical (e.g., total distance covered) and contextual (e.g., pass success probability) metrics based on imputed trajectories.

2   Related Work

Numerous methods have been proposed for time series and spatiotemporal data imputation. Early models such as BRITS [1], GRU-D [4], and MRNN [28] relied on recurrent models, but suffered from compounding errors due to dependence on their previous predictions. Non-autoregressive frameworks such as NAOMI [16], CSDI [24], and SAITS [7] mitigated this issue by enabling parallel imputation across time steps, improving robustness and efficiency. More recently, methods like TIDER [15] and TimesNet [25] were developed to capture temporal patterns such as seasonality or local biases. Other recent approaches further leverage graph structures (GRIN [6], SPIN [17], and NRTSI [21]), information bottlenecks (TimeCIB [5]), or low-rank priors (ImputeFormer [18]) to enhance imputation performance. Although they perform well on static multi-sensor systems, they are not designed to handle dynamic interactions in multi-agent scenarios.

Meanwhile, several frameworks have proposed dedicated architectures for multi-agent imputation to capture shifting spatial relationships of agents. Notable examples include Graph Imputer [19], GC-VRNN [26], and TranSPORTmer [2], which employ dynamic graph or set attention [13] architectures to model time-varying player interactions. More recently, methods tailored for sports data with advanced architectures have been explored, such as U2Diff [3] and Event2Tracking [10]. However, the former suffers from high computational costs and low data efficiency inherent to diffusion models, while the latter requires manually annotated event data, which are not always available. Interaction modeling has been widely studied in future trajectory forecasting [8,12,14,23,27,29], but these methods are not directly applicable to missing trajectory imputation.

3   Proposed Framework

Our study about multi-agent trajectory imputation assumes a scenario where the missing time intervals of players could differ from one another. To elaborate, let the trajectories of K players be $X_{1:T} = \{\mathbf{x}_{1:T}^k\}_{k=1}^K$, where each player k's input features $\mathbf{x}_t^k$ at each time t consist of their (x, y) position $\mathbf{p}_t^k = (p_{t,x}^k, p_{t,y}^k)$, velocity $\mathbf{v}_t^k = (v_{t,x}^k, v_{t,y}^k)$, and acceleration $\mathbf{a}_t^k = (a_{t,x}^k, a_{t,y}^k)$. Here, the velocity and acceleration are calculated from the position values by the following approximations:

$$\mathbf{v}_t^k \approx \frac{\mathbf{p}_t^k - \mathbf{p}_{t-1}^k}{\Delta t}, \quad \mathbf{a}_t^k \approx \frac{\mathbf{v}_{t+1}^k - \mathbf{v}_t^k}{\Delta t} \tag{1}$$

where Δt is the difference between adjacent time steps.

In our scenario, each $\mathbf{x}_{1:T}^k$ has missing parts identified by a masking sequence $\mathbf{m}_{1:T}^k = (m_1^k, \ldots, m_T^k)$ where $m_t^k = 1$ if $\mathbf{x}_t^k$ is *observed* and 0 if it is *missing*. Then, an imputation model aims to take the incomplete data $\{\mathbf{m}_{1:T}^k \odot \mathbf{x}_{1:T}^k\}_{k=1}^K$ as input and produce imputed trajectories $\{\hat{\mathbf{x}}_{1:T}^k\}_{k=1}^K$. Combining these with the observed fragments results in complete trajectories, i.e.,

$$\tilde{\mathbf{x}}_{1:T}^k = \mathbf{m}_{1:T}^k \odot \mathbf{x}_{1:T}^k + (\mathbb{1}_T - \mathbf{m}_{1:T}^k) \odot \hat{\mathbf{x}}_{1:T}^k, \quad k = 1, \ldots, K. \tag{2}$$

The novelty of the proposed framework lies in the mechanism of enhancing the model performance by combining positions directly predicted by a neural network and those resulting from accumulating predicted derivatives (i.e., velocity and acceleration values). To elaborate on the details of the proposed mechanism, the remainder of this section consists of the following four parts: Section 3.1 describes the neural network for initial prediction, Section 3.2 introduces the derivative accumulation process for alternative predictions, Section 3.3 describes the weighted ensemble mechanism to combine multiple predictions resulting from the previous sections, and Section 3.4 explains the loss function for model training. See Fig. 2 illustrating the overall architecture of our framework.

3.1   Neural Network-Based Initial Prediction

This section describes the neural network architecture that makes *initial prediction* (IP) of imputed trajectories. It takes partially observed trajectories $\{\mathbf{m}_{1:T}^k \odot \mathbf{x}_{1:T}^k\}_{k=1}^K$ as an input and predicts each player k's full trajectory

$$\hat{\mathbf{x}}_{1:T}^{k,i} = \{(\hat{p}_{t,x}^{k,i}, \hat{p}_{t,y}^{k,i}, \hat{v}_{t,x}^{k,i}, \hat{v}_{t,y}^{k,i}, \hat{a}_{t,x}^{k,i}, \hat{a}_{t,y}^{k,i})\}_{t=1}^T, \tag{3}$$

where the superscript i stands for "initial".

Since there is generally no inherent order among players in team sports, modeling their movements requires ensuring *permutation-equivariance*. That is, permuting the input order should not affect each player's output, except for applying the same permutation in the output order. Following a recent study on the ball trajectory inference in team sports [11], we employ Set Transformer [13] to ensure the permutation-equivariance of outputs.
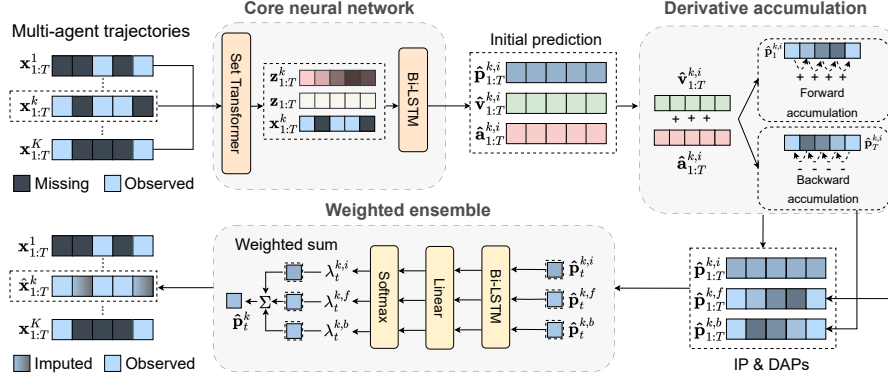
Fig. 2: Overview of the proposed framework.

To be specific, we obtain permutation-equivariant player-wise embeddings $\{\mathbf{z}_t^k\}_{k=1}^{K}$ from the encoder of a Set Transformer and a single permutation-invariant embedding $\mathbf{z}_t$ from a full Set Transformer for each time step t:

$$(\mathbf{z}_t^1, \ldots, \mathbf{z}_t^K) = \text{ST-Encoder}(m_t^1\mathbf{x}_t^1, \ldots, m_t^K\mathbf{x}_t^K),$$
$$\mathbf{z}_t = \text{SetTransformer}(m_t^1\mathbf{x}_t^1, \ldots, m_t^K\mathbf{x}_t^K). \tag{4}$$

Frame-by-frame application of the Set Transformer to the input features yields embeddings $\{(\mathbf{z}_t^1, \ldots, \mathbf{z}_t^K, \mathbf{z}_t)\}_{t=1}^{T}$. Then, bidirectional LSTMs [9] sharing weights across players extract the sequential information from the concatenated sequence $\{(\mathbf{x}_t^k, \mathbf{z}_t^k, \mathbf{z}_t)\}_{t=1}^{T}$ per player k by updating joint hidden states:

$$\mathbf{h}_t^{k,f} = \text{LSTM}^f(\mathbf{x}_t^k, \mathbf{z}_t^k, \mathbf{z}_t; \mathbf{h}_{t-1}^{k,f}), \quad \mathbf{h}_t^{k,b} = \text{LSTM}^b(\mathbf{x}_t^k, \mathbf{z}_t^k, \mathbf{z}_t; \mathbf{h}_{t+1}^{k,b}) \tag{5}$$

Lastly, a fully-connected layer decodes the joint hidden state to output a prediction $\hat{\mathbf{x}}_t^{k,i} = \text{FC}(\mathbf{h}_t^{k,f}, \mathbf{h}_t^{k,b})$ at each time t. In later sections, we combine it with alternative predictions to get a more accurate final prediction.

3.2   Derivative-Accumulating Prediction

In this section, we start from the fact that players' acceleration values highly vary over time since they are directly related to their stochastic intents. In contrast, velocities are more correlated across neighboring time steps, and positions exhibit even stronger autocorrelation than their derivatives. This implies that accurately predicting acceleration values can lead to more stable and accurate position estimates, provided the physical relationships in Eq. (1) are maintained. However, since the model introduced in Section 3.1 does not enforce these relationships in its outputs $\{\hat{\mathbf{x}}_{1:T}^{k,i}\}_{k=1}^{K}$, accurate prediction of the derivatives does not necessarily lead to improved position accuracy.

Taking this into account, we make alternative *derivative-accumulating predictions* (DAP), which enforces the physical relationships in Eq. (1) for improved stability. Specifically, given a missing segment (t_s, t_e) for a player k, we recursively predict positions inside the segment by accumulating velocities and accelerations in either direction using the following equation derived from Eq. (1):

$$\mathbf{p}_{t+1}^k \approx \mathbf{p}_t^k + \mathbf{v}_{t+1}^k \Delta t, \quad \mathbf{v}_{t+1}^k \approx \mathbf{v}_t^k + \mathbf{a}_t^k \Delta t. \tag{6}$$

That is, along the forward direction, we start from the observed position $\mathbf{p}_{t_s}^k$ by setting $\hat{\mathbf{p}}_{t_s}^{k,f} = \mathbf{p}_{t_s}^k$ and recursively add predicted velocities and accelerations to obtain *forward predictions* $\hat{\mathbf{p}}_t^{k,f}$ for $t \in (t_s, t_e)$ as follows:

$$\hat{\mathbf{p}}_t^{k,f} \approx \hat{\mathbf{p}}_{t-1}^{k,f} + \hat{\mathbf{v}}_t^{k,i} \Delta t \approx \hat{\mathbf{p}}_{t-1}^{k,f} + (\hat{\mathbf{v}}_{t-1}^{k,i} + \hat{\mathbf{a}}_{t-1}^{k,i} \Delta t) \Delta t \tag{7}$$

Likewise, we start from the observed position $\mathbf{p}_{t_e}^k$ at the opposite endpoint and recursively subtract the predicted derivatives to obtain *backward predictions* $\hat{\mathbf{p}}_t^{k,b}$.

Adopting DAPs instead of initial prediction carries several advantages. First, since the loss between these DAPs and the ground truth penalizes unstable predictions of the velocity and acceleration, minimizing it improves the smoothness of the predicted derivatives. Considering that existing position-oriented imputation models suffer from fluctuating trajectories, these smooth derivatives have a clear advantage in that they result in more plausible positional predictions. Furthermore, enforcing the relationships between the physical quantities imposes an additional inductive bias on the model, making it more data-efficient.

3.3   Weighted Ensemble of Multiple Predictions

Although DAP introduced in Section 3.2 has clear advantages over IP resulting from Section 3.1, it also has a potential drawback known as the *error compounding problem*. Because DAP only relies on the observation at an endpoint as an anchor and the predicted derivatives that are accumulated on the anchor, prediction errors tend to grow as the number of iterations in Eq. (7) (or its backward counterpart) increases. In contrast, IP is robust to this problem since it is less sensitive to estimates at certain time steps.

To balance this trade-off, we take a hybrid approach that combines the strengths of both IP and DAP. Rather than exclusively relying on one prediction, it performs a soft voting ensemble by computing a weighted sum of three predictions, IP and forward/backward DAPs. These weights are dynamically learned through an additional player-wise Bi-LSTM, which adapts the contribution of each prediction at each time step.

More specifically, for each player k and time step t, we feed the three predictions $\hat{\mathbf{x}}_t^{k,i}, \hat{\mathbf{p}}_t^{k,f}, \hat{\mathbf{p}}_t^{k,b}$ along with the context embeddings $\mathbf{z}_t^k, \mathbf{z}_t$ from Eq. (4) into a Bi-LSTM that updates its hidden states:

$$\tilde{\mathbf{h}}_t^{k,f} = \text{LSTM}^f(\hat{\mathbf{x}}_t^{k,i}, \hat{\mathbf{p}}_t^{k,f}, \hat{\mathbf{p}}_t^{k,b}, \mathbf{z}_t^k, \mathbf{z}_t, \gamma_t; \tilde{\mathbf{h}}_{t-1}^{k,f}), \tag{8}$$

$$\tilde{\mathbf{h}}_t^{k,b} = \text{LSTM}^b(\hat{\mathbf{x}}_t^{k,i}, \hat{\mathbf{p}}_t^{k,f}, \hat{\mathbf{p}}_t^{k,b}, \mathbf{z}_t^k, \mathbf{z}_t, \gamma_t; \tilde{\mathbf{h}}_{t+1}^{k,b}). \tag{9}$$

where $\gamma_t = \exp(-\max\{0, \mathbf{W}_\gamma \delta_t + \mathbf{b}_\gamma\})$ is the temporal decay factor introduced in BRITS [1], indicating the distance of t from observed endpoints. We define $\delta_t = (t - t_s, t_e - t)$ to provide symmetric time gaps for weighting the bidirectional DAPs. Then, a fully-connected layer with a softmax activation returns

$$(\lambda_t^{k,i}, \lambda_t^{k,f}, \lambda_t^{k,b}) = \text{Softmax}(\text{FC}(\tilde{\mathbf{h}}_t^{k,f}, \tilde{\mathbf{h}}_t^{k,b})) \tag{10}$$

that add up to 1. Based on these weights, the model yields a final prediction

$$\hat{\mathbf{p}}_t^k = \lambda_t^{k,i} \hat{\mathbf{p}}_t^{k,i} + \lambda_t^{k,f} \hat{\mathbf{p}}_t^{k,f} + \lambda_t^{k,b} \hat{\mathbf{p}}_t^{k,b}. \tag{11}$$

Combining this final prediction with the observed fragments by Eq. (2) results in complete trajectories across the entire period:

$$\tilde{\mathbf{x}}_{1:T}^k = \mathbf{m}_{1:T}^k \odot \mathbf{x}_{1:T}^k + (\mathbb{1}_T - \mathbf{m}_{1:T}^k) \odot \hat{\mathbf{x}}_{1:T}^k, \tag{12}$$

where $\hat{\mathbf{x}}_t^k = (\hat{p}_{t,x}^k, \hat{p}_{t,y}^k, \hat{v}_{t,x}^{k,i}, \hat{v}_{t,y}^{k,i}, \hat{a}_{t,x}^{k,i}, \hat{a}_{t,y}^{k,i})$.

Figure 3 illustrates how MIDAS dynamically adjusts ensemble weights depending on the characteristics of the missing segment. For Player 2, who has a short and stable missing trajectory, the model assigns negligible weight to IP, relying more on DAPs for imputation. In contrast, Player 3's longer and more variable missing trajectory leads to higher IP weights, especially in the middle where DAP errors may accumulate. This highlights MIDAS's ability to improve the final prediction by adaptively combining its components.

3.4   Loss Function

In our framework, improving the accuracy of the prediction at each stage contributes to a more reliable ensemble output. Therefore, we minimize not only the loss between the final ensemble prediction and the true trajectories but also the loss of each auxiliary prediction. Specifically, we compute the mean absolute errors (MAEs) for the IP and DAPs as well as the ensemble prediction, respectively, and train the entire architecture by minimizing the sum of these MAEs. Formally, for the MAE losses $\mathcal{L}^i$ of the IP, $\mathcal{L}^f$ and $\mathcal{L}^b$ for the bidirectional DAPs, and $\mathcal{L}^h$ for the ensemble prediction, the model is trained by minimizing

$$\mathcal{L}^{\text{MIDAS}} = \mathcal{L}^i + \mathcal{L}^f + \mathcal{L}^b + \mathcal{L}^h. \tag{13}$$
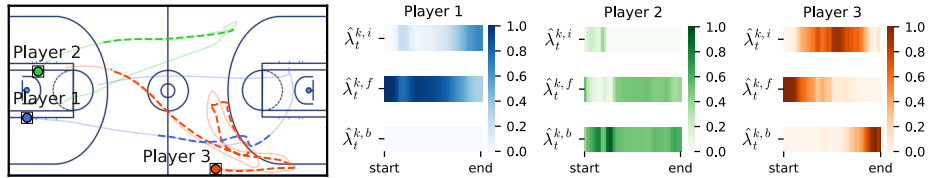

Fig. 3: Example of ensemble weights for individual imputed trajectories, where circles indicate each player's initial position.

Table 1: Details on the three sports datasets.

| | Soccer | | Basketball | | A. Football | |
Split	Matches	Frames	Matches	Frames	Matches	Frames
Training	2	65,014	70	1,621,835	—	425,000
Validation	0.5	20,104	10	216,118	—	52,150
Test	0.5	21,242	20	468,885	—	—

4   Experiments

In this section, we conduct experiments on multiple sports datasets to evaluate the performance of MIDAS and its generalizability across different sports.

4.1   Data Preparation

In the experiments, we independently trained and evaluated models on three public datasets collected from popular team sports: soccer, basketball, and American football. The soccer dataset is provided by Metrica Sports[7] and contains tracking data for 22 players collected across three matches. For basketball, we use the first 100 matches of SportsVU NBA dataset[8], containing trajectories of 10 players per match. The American football dataset is from the Kaggle competition[9] and is based on NFL's Next Gen Stats. We adopt its preprocessed version[10] provided by the NRTSI paper [21], which contains 9,543 five-second time series of six offensive players. The original sampling rates of the three datasets are 25Hz, 25Hz, and 10Hz, respectively, but we downsample all datasets to 10Hz for consistency. As model inputs, we use 200 frames (20 seconds) per sequence for soccer and basketball and 50 frames (5 seconds) for American football.

4.2   Missing Scenarios

To evaluate the model performance on various missing patterns, we consider the following three scenarios that may occur during data acquisition processes:

1. Uniform missing: All players have missing values at the same time interval. Note that among the baselines, NAOMI [16] and NRTSI [21] are designed to only handle this scenario and are not capable of the following other scenarios.
2. Agent-wise missing: Individual players have different missing intervals.
3. Broadcasting camera: A virtual camera follows the ball and only captures the players inside the camera view, resulting in missing values for the remaining players as shown in Fig. 1. Following Graph Imputer [19], we conduct experiments only on the soccer dataset for this scenario.

---

[7] https://github.com/metrica-sports/sample-data

[8] https://github.com/linouk23/NBA-Player-Movements

[9] https://www.kaggle.com/competitions/nfl-big-data-bowl-2021

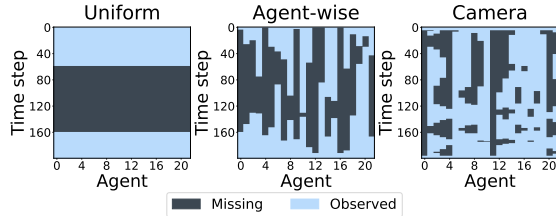[10] https://github.com/lupalab/NRTSI/tree/main/codes_stochastic

Fig. 4: Masking matrix examples for three different missing scenarios.

Fig. 4 illustrates examples of masking matrices from the soccer dataset used in each missing scenario. Since our task is to impute trajectories given observed data before and after the missing intervals, we ensure that the first and last five frames of each sequence are always observable, following previous work [16,19,21]. During training, we apply a dynamic missing rate ranging from 0.1 to 0.9, while testing is conducted with a fixed missing rate of 0.5 to encourage the model to generalize across various missing rate scenarios. Additional evaluations with different missing rates are presented as the ablation study in Section 4.5.

4.3   Baseline Models and Evaluation Metrics

In the experiments, we compare the imputation performance of MIDAS with several baselines, including naive methods such as linear interpolation (LI) and cubic spline (CS), as well as deep learning models such as BRITS [1], NAOMI [16], NRTSI [21], CSDI [24], Graph Imputer (GI) [19], and ImputeFormer (IF) [18]. To evaluate the impact of our derivative-accumulating self-ensemble mechanism, we also implement a baseline that only uses the initial prediction (IP) described in Section 3.1 and is trained with the loss $\mathcal{L}^i$ from Section 3.4. For models such as BRITS, NAOMI, and CSDI that do not preserve permutation-equivariance of input players, we sort trajectories by the sum of their average x and y coordinates to ensure permutation robustness. We compare these baselines using two evaluation metrics: (1) *position error* (PE) indicating the average Euclidean distance between the true and predicted positions and (2) *step change error* (SCE) [16,21] defined as the average absolute difference between the variance of the true and predicted velocities to assess the trajectories' physical plausibility.

4.4   Main Experimental Results

The resulting Table 2 shows that the proposed MIDAS consistently outperforms other baselines in both positional accuracy (PE) and physical plausibility (SCE). Linear interpolation (LI) and cubic spline (CS) offer simple yet competitive results in some scenarios. Notably, deep learning baselines such as BRITS, NAOMI, and CSDI, which are not designed for multi-agent domains, often fail to exceed these naive baselines. Similarly, Graph Imputer (GI) performs worse than LI and CS in many cases, as previously reported in its original article [19].

Table 2: Performance of imputation methods on different datasets and scenarios.

Scenario	Metric	LI	CS	BRITS	NAOMI	NRTSI	CSDI	GI	IF	IP	MIDAS
Soccer											
Uniform	PE	3.8406	2.2085	7.4859	4.5343	3.1791	3.4295	4.6511	2.0898	1.4563	**1.3205**
	SCE	0.1299	0.0867	3.9089	3.9793	0.0854	0.1586	0.1191	0.0815	0.1488	**0.0516**
Agent-wise	PE	5.0752	11.4647	5.7266	—	—	4.0279	5.6011	2.5798	2.0755	**1.9832**
	SCE	0.1631	0.2939	2.9627	—	—	0.1305	0.1508	0.0976	0.1057	**0.0535**
Camera	PE	3.1083	1.9209	7.4208	—	—	3.5181	3.6512	2.2151	1.4879	**1.2296**
	SCE	0.0993	0.0547	4.1967	—	—	0.2132	0.0934	0.3149	0.1554	**0.0374**
Basketball											
Uniform	PE	3.3481	2.3114	2.9085	1.5254	2.5291	2.2558	2.8305	1.3622	0.9801	**0.9727**
	SCE	0.1483	0.1025	1.0521	0.3230	0.0734	0.0631	0.1066	0.0531	**0.0432**	0.0438
Agent-wise	PE	4.4992	10.3857	2.4238	—	—	2.3471	2.5859	**1.3345**	1.3832	1.3862
	SCE	0.1787	0.2715	0.5397	—	—	0.0563	0.0700	0.0485	**0.0373**	0.0381
American Football											
Uniform	PE	0.8897	0.7448	1.7990	0.9692	0.5158	0.5558	0.8899	0.3673	0.2073	**0.1542**
	SCE	1.1063	0.9463	10.9459	2.3112	0.2989	0.4905	1.1023	0.2858	0.1990	**0.1126**
Agent-wise	PE	1.5128	1.2041	1.7527	—	—	0.6182	1.5128	0.3944	0.2383	**0.2104**
	SCE	1.0641	0.5306	10.6807	—	—	0.4288	1.0631	0.1869	0.1180	**0.0967**

Table 3: Performance of the top-3 methods on the basketball dataset when trained with full data (70 games) versus limited data (3 games).

Scenario	Metric	Limited Data			Full Data		
		IF	IP	MIDAS	IF	IP	MIDAS
Uniform	PE	1.6741	1.1868	**1.1438**	1.3622	0.9801	**0.9727**
	SCE	0.0628	0.1483	**0.0493**	0.0531	**0.0432**	0.0438
Agent-wise	PE	1.8876	1.6414	**1.5994**	**1.3345**	1.3832	1.3862
	SCE	0.0645	0.0452	**0.0439**	0.0485	**0.0373**	0.0381

By contrast, our model for initial prediction (IP) already surpasses most baselines, demonstrating its strength in modeling dynamic multi-agent interactions. Furthermore, comparing IP and MIDAS highlights the effectiveness of our self-ensemble mechanism. By effectively leveraging the complementary strengths of initial predictions and alternative derivative-accumulating predictions, it leads to superior performance even compared to state-of-the-art methods such as ImputeFormer (IF) in most scenarios.

Such observations are also evident in Fig. 5, where LI, NRTSI, and CSDI often generate unrealistic trajectories, either overly linear or changing direction too frequently, resulting in erratic and implausible motion patterns. In contrast, IF and MIDAS produce trajectories that closely resemble true player movements, as indicated by the relatively low PEs over the missing interval.

Meanwhile, an interesting observation emerges in the basketball dataset, where IP and IF show performance comparable to or slightly better than MIDAS in some cases. We attribute this to the larger amount of training data available
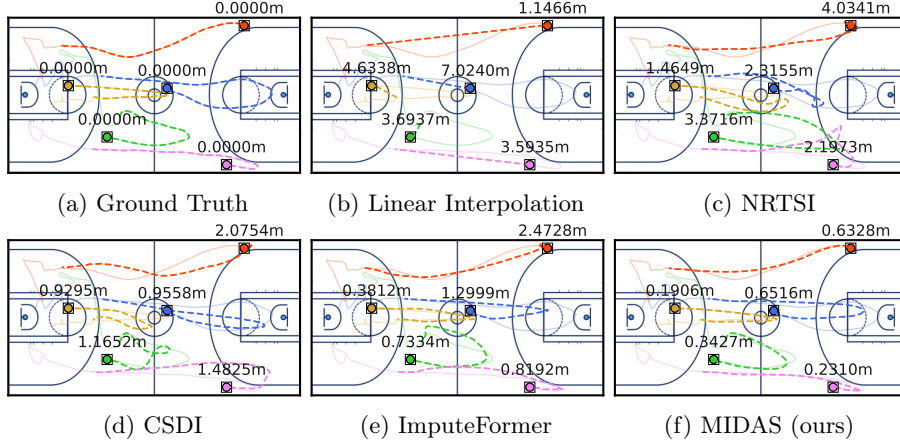
Fig. 5: Ground truth (solid) and imputed trajectories (dashed) for a uniform missing scenario in basketball. Circles indicate the players' initial positions, and the annotated values represent average position errors over the missing interval.

(70 games), which may reduce the relative advantage of MIDAS. In contrast, for soccer and American football, where training data are more limited, MIDAS's derivative-based self-ensemble mechanism demonstrates its data efficiency by significantly outperforming other methods.

To investigate whether this data efficiency holds for the basketball dataset, we measure model performance on an additional setting where models are trained using only 3 games (comparable in scale to the training data used for the soccer dataset). As shown in Table 3, all models experience a performance drop when trained on limited data. However, the degradation is much less severe for MIDAS, which eventually outperforms both IF and IP under this setting. This observation underscores the practical utility of MIDAS in real-world sports analytics, where obtaining large amounts of complete tracking data is often difficult due to the competitive and commercial nature of professional sports.

4.5   Ablation Studies

In addition to the main experiments, we conducted two ablation studies to further investigate the model's behavior across varying difficulties and the impact of its features and components on the performance.

Model Behavior Across Varying Difficulties. To better understand how MIDAS responds to varying levels of difficulty, we conduct an ablation study analyzing the model's behavior across different missing lengths. For the agent-wise missing scenario in the soccer and basketball datasets, we divide trajectory segments in the test set into three groups based on missing length.

As shown in Table 4, the position error increases with longer missing segments, and the advantage of DAP over IP diminishes. When the missing length is short,

the DAP components outperform the initial prediction (IP) by a wide margin, resulting in substantial performance gains through the ensemble. However, as the missing length increases, the uncertainty of player motion makes DAP less reliable, reducing its advantage. This trend is reflected in the ensemble weights as well: $\hat{\lambda}_t^{k,i}$ increases and $\hat{\lambda}_t^{k,f}$ decreases as the missing length grows.

In addition, the advantage of DAP over IP varies by sport. In soccer, DAP maintains its superiority over IP even for long segments, and the IP component receives almost zero weight in the ensemble, indicating that MIDAS relies entirely on DAP. In contrast, in basketball, IP surpasses DAP for medium and long segments. This is reflected in noticeably higher $\hat{\lambda}_t^{k,i}$, suggesting its greater influence in ensemble prediction. This difference likely stems from the nature of basketball, where the variability of player motion is higher relative to the small court size, making long-range accumulation of derivatives more error-prone.

Impact of Derivative Features. To empirically justify the use of derivative features in IP and DAP, we compared the performance of MIDAS on the broadcasting camera scenario in soccer with those trained without accelerations (i.e., using positions and velocities) and even without velocities (i.e., only using positions) of observed trajectories. Furthermore, we configured IP to predict only positions and velocities, and DAP to estimate positions only based on predicted velocities.

Table 4: Position errors and ensemble weights of each MIDAS component in Eq. (11) across different missing lengths within 200-frame (20 s) trajectory segments. Segments are grouped into the short, medium, and long thirds based on missing length, where the mean and standard deviation of the number of missing frames in each group are reported in the table.

Sports	Category	Missing Frames	$\hat{\mathbf{p}}_t^{k,i}$ ($\hat{\lambda}_t^{k,i}$)	$\hat{\mathbf{p}}_t^{k,f}$ ($\hat{\lambda}_t^{k,f}$)	$\hat{\mathbf{p}}_t^{k,b}$ ($\hat{\lambda}_t^{k,b}$)	$\hat{\mathbf{p}}_t^k$
Soccer	Short	33.30 ± 15.72	0.1379 (0.0001)	0.0742 (0.6501)	0.0783 (0.3498)	**0.0504**
	Medium	90.42 ± 19.17	0.7939 (0.0003)	0.7492 (0.6110)	0.7624 (0.3887)	**0.7004**
	Long	173.41 ± 21.70	2.7359 (0.0002)	2.7017 (0.5034)	2.7186 (0.4964)	**2.6082**
Basketball	Short	33.86 ± 16.11	0.0444 (0.1596)	0.0424 (0.5216)	0.0426 (0.3187)	**0.0376**
	Medium	90.65 ± 19.02	0.5395 (0.1783)	0.5483 (0.4809)	0.5482 (0.3406)	**0.5312**
	Long	172.60 ± 21.20	1.8104 (0.1801)	1.8288 (0.4313)	1.8258 (0.3985)	**1.8005**

Table 5: Position errors of each MIDAS component in Eq. (11), when trained using different subsets of features.

IP Features	DAP Features	$\hat{\mathbf{p}}_t^{k,i}$	$\hat{\mathbf{p}}_t^{k,f}$	$\hat{\mathbf{p}}_t^{k,b}$	$\hat{\mathbf{p}}_t^k$
Position	Velocity	4.0008	2.8203	2.8261	2.5549
Position	Vel. & accel.	2.9119	2.6973	2.6858	2.4954
Pos. & vel.	Velocity	1.6600	1.5742	1.5590	1.4025
Pos. & vel.	Vel. & accel.	1.6235	1.5731	1.5670	1.4013
All features	Velocity	1.6713	1.4985	1.5595	1.2644
All features	Vel. & accel.	**1.4963**	**1.4122**	**1.3982**	**1.2296**

We then compared these "velocity-only" DAP models with their counterparts that also incorporate predicted accelerations as in Eq. (7).

Table 5 shows that providing velocities and accelerations as input features for IP clearly improves the imputation performance. We attribute this to the neural network architecture for IP, where Set Transformers independently encode multi-agent contexts at each time step via Set Transformers, and player-wise Bi-LSTMs link temporal information. Providing derivatives as input allows the Set Transformers to utilize information of adjoining time steps, resulting in more comprehensive context embeddings compared to using positions alone.

In addition, we observe consistent performance improvements when predicted accelerations are incorporated into DAP alongside velocities (i.e., rows 2, 4, and 6 versus 1, 3, and 5, respectively). We attribute this to the fact that including accelerations enables DAP to capture the finer dynamics of player motion, mitigating the drift caused by error compounding in velocity-only DAP.

4.6 Time Complexity Analysis

While achieving high imputation accuracy is important, real-world deployment requires efficient processing without excessive computational overhead. To assess the feasibility of real-time use, we analyze both the theoretical and empirical time complexity of the proposed MIDAS framework.

From a theoretical standpoint, we consider three key parameters: the number of time steps T, the hidden dimension m, and the number of players K. It takes $O(TK^2m)$ for Set Transformer used in the initial prediction, $O(TKm^2)$ for Bi-LSTMs employed in the initial prediction and weighted ensemble. In total, $O(TK^2m + TKm^2) \approx O(TKm^2)$ as $m \gg K$. Since Bi-LSTM adopts a recurrent structure, the entire computation process proceeds over $O(T)$ steps, proportional to the length of the sequence.

In addition to this analysis, we empirically evaluated the inference time of MIDAS and several baseline models using an NVIDIA TITAN RTX GPU with 24GB memory. As summarized in Table 6, although MIDAS is slightly slower than ImputeFormer [18], it can process an entire 40-minute game (half-match for

Table 6: Empirical analysis of time complexity on soccer and basketball data. Each row corresponds to an imputation method, with the number of parameters, average inference time per 20-second window, and total processing time for half a soccer match and a full basketball match in the test set described in Table 1.

Method	Soccer			Basketball		
	Params	Inference	Total	Params	Inference	Total
NRTSI	84,090,968	893.07 ms	84.40 s	84,017,192	906.74 ms	100.55 s
CSDI	414,209	394.44 ms	39.68 s	413,825	324.70 ms	37.68 s
ImputeFormer	1,290,978	6.19 ms	4.92 s	1,060,578	5.68 ms	3.85 s
MIDAS	3,945,579	24.32 ms	8.22 s	3,945,579	21.58 ms	5.71 s

soccer and full game for basketball) within 10 seconds. This demonstrates that MIDAS operates well within real-time constraints, enabling its potential use in practical scenarios such as live match analysis or broadcasting augmentation.

5   Applications

In Section 4, we evaluate model performance primarily based on metrics related to missing periods in sports data. In practice, however, what is more important in this domain is the quality of the trajectories for the entire period, as they can be utilized in diverse domain-specific downstream tasks. As examples, we present two promising applications in the soccer domain: approximating physical statistics and pass success probability from incomplete tracking data.

5.1   Approximation of Physical Statistics for Load Management

In this section, we explore how accurately our method can estimate statistics for a given period when imputed trajectories are combined with known observations. Specifically, we compare the *total distance* covered by a player and the *number of sprints* estimated by each method, as they are widely used as indicators for players' physical performance or fitness. We first compute velocities from the observed/imputed positions based on Eq. 1 and obtain speed values by calculating the norms of these velocity vectors. To make the best estimation from given positional predictions, we remove outliers whose speed is larger $12\,\mathrm{m\,s^{-1}}$ or whose norm of the acceleration exceeds $8\,\mathrm{m\,s^{-2}}$ and replace the values by linear interpolation. Also, we smoothen the resulting speed signal by applying a Savitzky-Golay filter [20]. After preprocessing speed signals, we compute the distance covered by each player by summing the speed values multiplied by $\Delta t = 0.1\mathrm{s}$. For the latter, if a player runs faster than $6\,\mathrm{m\,s^{-1}}$ for consecutive frames, we detect his/her movement during the frames as *sprint* and count the number of such sprints the player made during the given period.

Table 7: Players' physical statistics estimated by different models and their mean absolute percentage errors (MAPE) for the soccer test data.

Method	Distance (m)		Sprints	
	Mean	MAPE	Mean	MAPE
Ground Truth	11,093.5	—	41.49	—
Linear Interp.	10,167.8	8.46%	38.89	6.32%
Cubic Spline	10,686.3	3.73%	38.85	6.73%
BRITS	10,979.2	2.76%	59.89	53.62%
CSDI	11,343.0	2.77%	44.20	14.71%
Graph Imputer	8,972.1	19.15%	37.85	9.80%
ImputeFormer	11,441.7	3.22%	50.25	26.29%
MIDAS (ours)	**10,922.4**	**1.58%**	**40.71**	**4.95%**

For evaluation, we use the soccer test data consisting of the half of a match and assume the broadcasting camera scenario. Since players played for different time periods, we normalize each player's statistics by 90 minutes and calculate the averages of such normalized values estimated by MIDAS and other baselines, respectively. Note that players who ran fewer than two sprints during the half were excluded from every evaluation.

According to Table 7, MIDAS provides accurate estimates close to the ground truth. Especially considering that almost all baselines either suffer from inaccurate distance measures (linear interpolation, cubic spline, and Graph Imputer) or overestimate speed spikes (BRITS, CSDI, and ImputeFormer), it is obvious that our model takes clear advantage of smooth prediction of velocities. In a nutshell, our framework is practical in that it can provide reliable statistics with incomplete tracking data, which originally require complete player trajectories.

5.2   Approximation of Pass Success Probability for Spatial Analysis

One representative example of leveraging player tracking data for match analysis is Pitch Control [22], which estimates the probability that a pass to each location on the pitch would be successful. Such pass success probabilities for different destinations are typically visualized as a heat map overlaid on the pitch, so that domain experts can evaluate players' positioning and decision-making for both actual and hypothetical passes to different locations on the pitch.

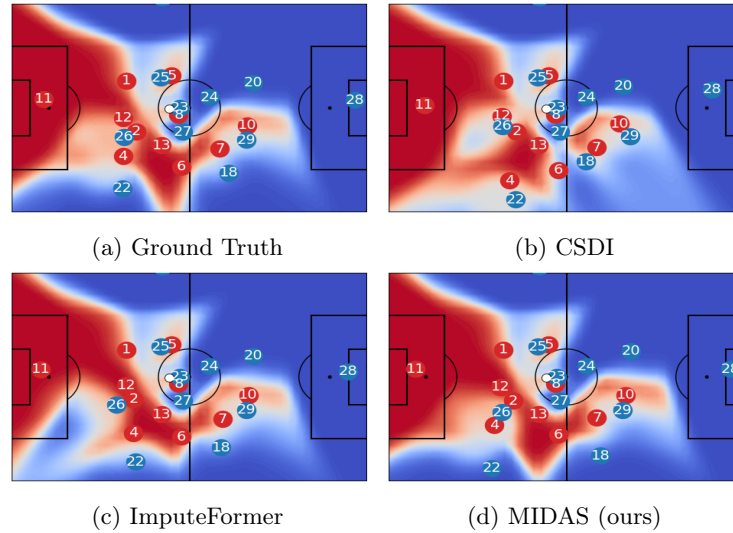(a) Ground Truth

(b) CSDI

(c) ImputeFormer

(d) MIDAS (ours)

Fig. 6: Pass success probability maps based on true and imputed player positions resulting from different methods in a partially-observable camera setting. Darker red and blue regions indicate a higher probability of the left and right teams gaining possession, respectively, if the ball is passed to those areas.

Thus, to demonstrate the applicability of imputation models in actual match analysis, we compare the Pitch Control maps generated using the player positions imputed by different methods. In the situation in Fig. 6 as an example, player 23 of the blue team has the ball, while his teammates 22 and 26 are making forward runs towards the open spaces on the left flank and in the center behind the red defensive line, respectively. The ground truth map (Fig. 6(a)) indicates that the blue team has a slightly higher control probability in both areas, reflecting reasonable pass success opportunities to these players, who are onside. However, inaccurate imputations by CSDI (Fig. 6(b)) and ImputeFormer (Fig. 6(c)) lead to issues such as players being mispredicted in offside positions or pass success probabilities being over- or underestimated in critical areas. In contrast, MIDAS (Fig. 6(d)) produces predictions that closely follow the actual player dynamics, resulting in probability maps that more accurately reflect the true game situation. This example instantiates how our framework facilitates more reliable downstream analysis by providing accurate imputation results.

6   Conclusions

This paper proposes MIDAS, a framework for imputing missing values in multi-agent trajectories with high accuracy and physical plausibility. MIDAS combines a permutation-equivariant neural network for initial trajectory prediction with a self-ensemble mechanism that incorporates derivative-based alternative predictions to refine imputation results and enforce physical consistency. Experiments on three team sports datasets under various missing scenarios demonstrate the effectiveness of our approach in generating trajectories with higher positional accuracy and improved reality than existing baselines. While this study focuses on team sports, we believe the proposed framework is applicable to other spatiotemporal domains. Future work will explore extending MIDAS to additional domains such as autonomous driving and crowd simulation, as well as its application to downstream tasks that require complete and reliable trajectory data.

Acknowledgments

References

1. Cao, W., Wang, D., Li, J., Zhou, H., Li, L., Li, Y.: BRITS: Bidirectional recurrent imputation for time series. In: Advances in Neural Information Processing Systems 31 (2018)

2. Capellera, G., Ferraz, L., Rubio, A., Agudo, A., Moreno-Noguer, F.: TranSPORT-mer: A holistic approach to trajectory understanding in multi-agent sports. In: Proceedings of the 17th Asian Conference on Computer Vision (2024)

3. Capellera, G., Rubio, A., Ferraz, L., Agudo, A.: Unified uncertainty-aware diffusion for multi-agent trajectory modeling. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2025)

4. Che, Z., Purushotham, S., Cho, K., Sontag, D.A., Liu, Y.: Recurrent neural networks for multivariate time series with missing values. Scientific Reports **8** (2018)

5. Choi, M., Lee, C.: Conditional information bottleneck approach for time series imputation. In: Proceedings of the 12th International Conference on Learning Representations (2023)

6. Cini, A., Marisca, I., Alippi, C.: Multivariate time series imputation by graph neural networks. In: Proceedings of the 10th International Conference on Learning Representations (2022)

7. Du, W., Côté, D., Liu, Y.: SAITS: Self-attention-based imputation for time series. Expert Systems with Applications **219** (2023)

8. Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., Alahi, A.: Social GAN: Socially acceptable trajectories with Generative Adversarial Networks. In: IEEE Conference on Computer Vision and Pattern Recognition (2018)

9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)

10. Hughes, H., Horton, M., Wei, X., Gammulle, H., Fookes, C., Sridharan, S., Lucey, P.: Event2Tracking: Reconstructing multi-agent soccer trajectories using long-term multimodal context. In: Proceedings of the 39th AAAI Conference on Artificial Intelligence (2025)

11. Kim, H., Choi, H., Kim, C., Yoon, J., Ko, S.: Ball trajectory inference from multi-agent sports contexts using set transformer and hierarchical bi-lstm. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (2023)

12. Kipf, T.N., Fetaya, E., Wang, K., Welling, M., Zemel, R.S.: Neural relational inference for interacting systems. In: Proceedings of the 35th International Conference on Machine Learning (2018)

13. Lee, J., Lee, Y., Kim, J., Kosiorek, A.R., Choi, S., Teh, Y.W.: Set Transformer: A framework for attention-based permutation-invariant neural networks. In: Proceedings of the 36th International Conference on Machine Learning (2019)

14. Li, J., Yang, F., Tomizuka, M., Choi, C.: EvolveGraph: Multi-agent trajectory prediction with dynamic relational reasoning. In: Advances in Neural Information Processing Systems 33 (2020)

15. Liu, S., Li, X., Cong, G., Chen, Y., Jiang, Y.: Multivariate time-series imputation with disentangled temporal representations. In: The 11th International Conference on Learning Representations (2023)

16. Liu, Y., Yu, R., Zheng, S., Zhan, E., Yue, Y.: NAOMI: Non-autoregressive multiresolution sequence imputation. In: Advances in Neural Information Processing Systems 32 (2019)

17. Marisca, I., Cini, A., Alippi, C.: Learning to reconstruct missing data from spatiotemporal graphs with sparse observations. In: Advances in Neural Information Processing Systems 35 (2022)

18. Nie, T., Qin, G., Ma, W., Mei, Y., Sun, J.: Imputeformer: Low rankness-induced transformers for generalizable spatiotemporal imputation. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (2024)

19. Omidshafiei, S., Hennes, D., Garnelo, M., Wang, Z., Recasens, A., Tarassov, E., Yang, Y., Elie, R., Connor, J., Muller, P., Mackraz, N., Cao, K., Moreno, P., Sprechmann, P., Hassabis, D., Graham, I., Spearman, W., Heess, N., Tuyls, K.: Multiagent off-screen behavior prediction in football. Scientific Reports **12** (2022)
20. Savitzky, A., Golay, M.J.E.: Smoothing and differentiation of data by simplified least squares procedures. Analytical Chemistry **36**(8) (1964)
21. Shan, S., Li, Y., Oliva, J.B.: NRTSI: Non-recurrent time series imputation. In: IEEE International Conference on Acoustics, Speech and Signal Processing (2023)
22. Spearman, W., Basye, A., Dick, G., Hotovy, R., Pop, P.: Physics-based modeling of pass probabilities in soccer. In: MIT Sloan Sports Analytics Conference (2017)
23. Sun, F., Kauvar, I., Zhang, R., Li, J., Kochenderfer, M.J., Wu, J., Haber, N.: Interaction modeling with multiplex attention. In: Advances in Neural Information Processing Systems 35 (2022)
24. Tashiro, Y., Song, J., Song, Y., Ermon, S.: CSDI: Conditional score-based diffusion models for probabilistic time series imputation. In: Advances in Neural Information Processing Systems 34 (2021)
25. Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., Long, M.: TimesNet: Temporal 2D-variation modeling for general time series analysis. In: Proceedings of the 11th International Conference on Learning Representations (2023)
26. Xu, Y., Bazarjani, A., Chi, H., Choi, C., Fu, Y.: Uncovering the missing pattern: Unified framework towards trajectory imputation and prediction. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2023)
27. Yeh, R.A., Schwing, A.G., Huang, J., Murphy, K.: Diverse generation for multi-agent sports games. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019)
28. Yoon, J., Zame, W.R., van der Schaar, M.: Estimating missing data in temporal data streams using multi-directional recurrent neural networks. IEEE Transactions on Biomedical Engineering **66**(5), 1477–1490 (2019)
29. Yuan, Y., Weng, X., Ou, Y., Kitani, K.: AgentFormer: Agent-aware transformers for socio-temporal multi-agent forecasting. In: IEEE/CVF International Conference on Computer Vision (2021)