scientific reports

OPEN

Multiagent off-screen behavior prediction in football

Shayegan Omidshafiei[1]✉, Daniel Hennes[1], Marta Garnelo[1], Zhe Wang[1], Adria Recasens[1], Eugene Tarassov[1], Yi Yang[1], Romuald Elie[1], Jerome T. Connor[1], Paul Muller[1], Natalie Mackraz[1], Kris Cao[1], Pol Moreno[1], Pablo Sprechmann[1], Demis Hassabis[1], Ian Graham[2], William Spearman[2], Nicolas Heess[1] & Karl Tuyls[1]✉

In multiagent worlds, several decision-making individuals interact while adhering to the dynamics constraints imposed by the environment. These interactions, combined with the potential stochasticity of the agents' dynamic behaviors, make such systems complex and interesting to study from a decision-making perspective. Significant research has been conducted on learning models for forward-direction estimation of agent behaviors, for example, pedestrian predictions used for collision-avoidance in self-driving cars. In many settings, only sporadic observations of agents may be available in a given trajectory sequence. In football, subsets of players may come in and out of view of broadcast video footage, while unobserved players continue to interact off-screen. In this paper, we study the problem of multiagent time-series imputation in the context of human football play, where available past and future observations of subsets of agents are used to estimate missing observations for other agents. Our approach, called the *Graph Imputer*, uses past and future information in combination with graph networks and variational autoencoders to enable learning of a distribution of imputed trajectories. We demonstrate our approach on multiagent settings involving players that are partially-observable, using the Graph Imputer to predict the behaviors of off-screen players. To quantitatively evaluate the approach, we conduct experiments on football matches with ground truth trajectory data, using a camera module to simulate the off-screen player state estimation setting. We subsequently use our approach for downstream football analytics under partial observability using the well-established framework of pitch control, which traditionally relies on fully observed data. We illustrate that our method outperforms several state-of-the-art approaches, including those hand-crafted for football, across all considered metrics.

Predictive modeling of multiagent behaviors has been a topic of considerable interest in machine learning[1–3], financial economics[4–6], robotics[7–9], and sports analytics[10–13]. In such systems, decision-making agents interact within a shared environment, following an underlying dynamical process that may be stochastic and often infeasible to characterize analytically due to the complex interactions involved. Learning a dynamical model of such systems enables both the understanding and evaluation of agents' behaviors. Ideally, methods that learn models of such coupled dynamical systems should enable the prediction of future behaviors, the retrodiction of past behaviors, and ultimately the imputation (i.e., filling-in) of partially-occluded data, while respecting any constraints imposed by available observations. In this paper, we introduce such a method for multiagent time-series imputation under temporal occlusion, focusing specifically on the setting of prediction of human football players. In the football domain, off-screen player predictions are crucial for enabling the application of downstream analysis techniques such as pitch control[14], which rely on information about positions of *all* players in the game.

Football is an especially interesting testbed for the multiagent imputation problem as it involves dynamic interaction of several individuals and stochasticity due to the human decisions involved. A large corpus of prior works have targeted learning models for forward-prediction of multiagent trajectories[10–13,15–19]. In these works, a stream of observations for all involved entities (e.g., all players and the ball) is assumed to be available for some number of timesteps, after which the states of a subset of entities are predicted. However, the availability of full tracking information is a restrictive assumption (requiring the use of proprietary sensors from third-party providers). In many situations, only partial information about the state of a game is available (e.g., positions of only the players visible on broadcast camera), thus requiring imputation of missing data. In contrast to prior works, we target this under-explored multiagent imputation regime, wherein we assume available observations of on-screen players (e.g., as obtained from a vision-based tracking system), and seek to predict the unobserved

[1]DeepMind, London, UK. [2]Liverpool Football Club, Liverpool, UK. ✉email: somidshafiei@google.com; karltuyls@deepmind.com

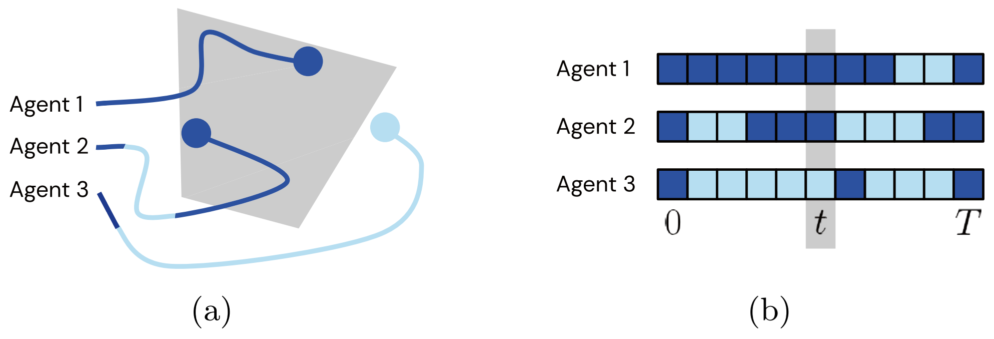

Figure 1. Stylized visualization of the multiagent time-series imputation setting. (**a**) Agent trajectories up to and including time t. Dark blue indicates trajectory portions that are observed (with light indicating otherwise); the camera field of view at the current time t is indicated in grey. (**b**) Visualization of masks $\boldsymbol{m}$ for all timesteps, where $\boldsymbol{m}_t^i = 1$ where dark, and $\boldsymbol{m}_t^i = 0$ where light. The mask at time t, which corresponds to the frame shown in (**a**), is highlighted in grey.

states of off-screen players, which can subsequently be used for downstream football analytics. Imputation of multivariate time series data involving interacting entities has various practical applications besides football. In financial markets, certain foreign exchange quotations are available more frequently than others, yet correlations between these financial instruments can be used to impute the missing data[4,5,20]. In clinical trials, multi-sensory data may be made with irregular measurements or unavailable for some sensors at certain times[21]. In natural language processing, in-filling of text conditioned on surrounding sentence context is an area of active research[22], and can naturally extend to multiagent conversational dialogue in-filling.

The partially-observable multiagent trajectories imputation problem is a new regime, which stands in contrast to previous works that only consider forecasting/future-predictions. The key contributions of this paper are as follows. First, we introduce a technique for multiagent imputation, which is applicable even under dynamic occlusion of random subsets of agents in a given trajectory sequence. Our model uses a combination of bidirectional variational LSTMs[23] and graph networks[24], with elements in place to handle arbitrarily-complex occlusions of sensory observations in multiagent settings. Second, we illustrate how our approach can be used to extend existing football analytics frameworks (namely, pitch control[14]) to partially-observable settings. Our experiments are conducted on a large suite of 105 full-length real-world football matches, wherein we compare our method against a number of existing approaches including Social LSTMs[7] and graph variational RNNs (GVRNNs)[3,12]. To our knowledge, this is the first study of trajectory imputation models in the football regime, and bears the potential to unlock the applicability of a vast number of prior analysis techniques (similar to pitch control) to football games that have only intermittent or partially-observable player tracking information.

Results

This section provides a high-level overview of the proposed approach and empirical results. Readers are referred to the "Methods" section for full technical details of the approach.

Problem formulation. We first define the multiagent time-series imputation problem, with football as the motivating example. As shown in Fig. 1a, observations of individual players may be unavailable when they are out of the camera frame, and players may disappear and reappear in view multiple times throughout a trajectory sequence. Moreover, the role of any individual player may change multiple times throughout a given trajectory sequence (e.g., a defender can behave in the manner of a midfielder or forward). This characteristic has been well-investigated in prior works[10,12] and ultimately implies that learned models should be invariant to permutations of player orders within each team. Such models should learn to predict the behavior of players conditioned on the game context, rather than purely on the players' prescribed roles in the team's formation.

Regarding notation, we henceforth refer to any scalars associated with an agent i at time t using unbolded variables, e.g., s_t^i. We use bold notation for vectors (e.g., $\boldsymbol{v}_t^i$). The concatenation of scalars or vectors across time and/or agent indices is denoted by, respectively, dropping the corresponding subscripts and superscripts (e.g., $\boldsymbol{s} = s_{0:T}^{1:N}$ and $\boldsymbol{s}_t = \boldsymbol{s}_t^{1:N}$).

We consider a set of N agents $\mathbb{I} = \{1, \ldots, N\}$. Let $\boldsymbol{x}_t^i \in \mathbb{R}^d$ denote the d-dimensional observation of the agent $i \in \mathbb{I}$ at time $t \in \mathbb{T} = \{0, \ldots, T\}$. In the football scenario considered in our evaluations, $d = 2$, with $\boldsymbol{x}_t^i$ corresponding to the (x, y) position of a player or the ball on the pitch at time t. For simplicity, we henceforth refer to $\boldsymbol{x}_t^i$ as the state (rather than observation) of agent i, as it comprises the variable of interest we seek to estimate in this work. In the time-series imputation regime, at each time step $t \in \mathbb{T}$, observations may be missing for any subset of players. Let $\boldsymbol{x} = \boldsymbol{x}_{0:T}^{1:N}$ be observed at the timesteps indicated by an agent-wise masking matrix $\boldsymbol{m}$ valued in $\{0, 1\}^d$, such that a given dimension of $\boldsymbol{m}_t^i$ is equal to 1 whenever an observation of agent i is available at timestep t, and 0 otherwise (see Fig. 1b). In the football context, each player's on-pitch (x, y) position is either fully observed at a given time, or fully unobserved (i.e., $\boldsymbol{m}_t^i \in \{(0, 0), (1, 1)\}$, such that there are no situations where a player's x-position is observed while their y-position is not, or vice versa). The objective is then to compute estimates $\hat{\boldsymbol{x}} \in \mathbb{R}^d$ of all the unobserved agent states at all timesteps. More precisely, the multiagent

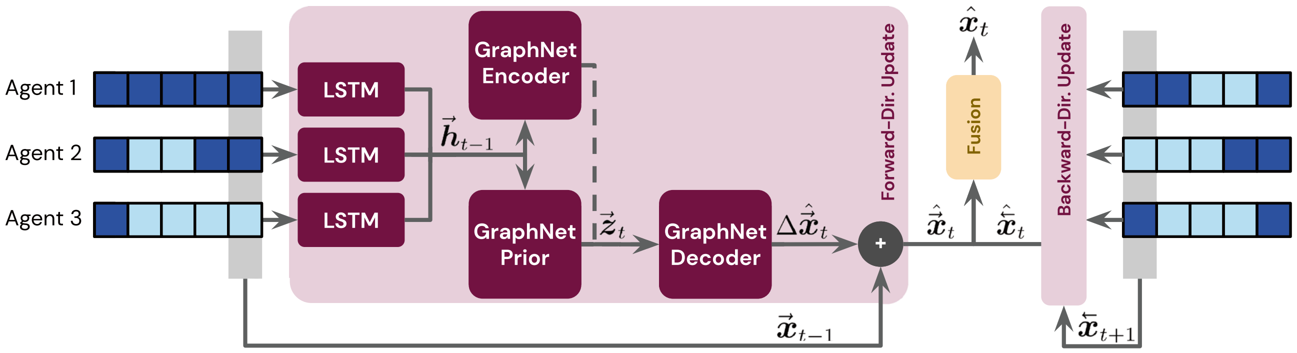

Figure 2. Graph Imputer model. Our model imputes missing information at each timestep using a combination of bidirectional LSTMs and graph networks. An exposition of a forward-direction update (corresponding to DIRECTIONALUPDATE in Algorithm 1 in the "Methods" section) is provided in the left portion of the figure. Dark blue boxes indicate trajectory segments that are observed for each agent (with light blue indicating otherwise). In each direction, agent-specific temporal context is updated via LSTMs with shared parameters. All agents' LSTM hidden states, $\vec{\boldsymbol{h}}_{t-1}$, are subsequently used as node features in variational graph networks to ensure information-sharing across agents. This enables learning of a distribution over agent state deviations, $\Delta\vec{\boldsymbol{x}}_t$. The process is likewise repeated in the backward-direction (right portion of the figure), with the directional updates fused to produce an imputed estimate $\hat{\boldsymbol{x}}_t$ at each time t. The dotted line indicates that the Graphnet encoder is used only at training time, with the GraphNet prior being used for the final evaluations conducted at test time.

time-series imputation problem takes the observed states $\boldsymbol{x} \odot \boldsymbol{m}$ as input, where $\odot$ refers to the Hadamard (or element-wise) product, and aims to output a full prediction $\hat{\boldsymbol{x}}_{0:T}^{1:N}$. We quantify this in our experiments via the evaluation loss $\mathcal{L}_2(\hat{\boldsymbol{x}} \odot (1-\boldsymbol{m}), \boldsymbol{x} \odot (1-\boldsymbol{m}))$.

Workflow. This section provides a high-level overview of the proposed model, called the *Graph Imputer*. For full technical details on the model architecture, hyperparameters, and training, readers are referred to the "Methods" section.

Figure 2 provides an overview of our proposed approach. The agents modeled in our domain of interest are human football players, who can exhibit stochastic behaviors on-pitch. To enable learning of stochastic predictions given an observation stream, our model learns along two axes: (i) across time via bidirectional LSTMs, which autoregressively generate unobserved agent states; and (ii) across agents via a combination of graph networks (GraphNets)[24] and variational RNNs (VRNNs)[23], which model the multiagent interactions involved and enable sampling of distributions of imputed trajectories. The forward- and backward-direction imputed states are fused at each timestep, thus ensuring that all available temporal and agent-interaction information is used throughout the entire generated sequence.

In Fig. 2, we expand the forward-directional update for the agents to provide clearer exposition of the updates conducted; the backward-direction update is analogous, as detailed in "Methods". Given a stream of observations in each direction, LSTM hidden states are collected across all agents, thus summarizing temporal information; in Fig. 2, we refer to the collection of agents' hidden states as $\vec{\boldsymbol{h}}_{t-1}$ for forward-directional updates. We subsequently use GraphNets to conduct inter-agent information sharing, with each graph being composed of N nodes corresponding to the number of agents in the system (e.g., $N = 23$ in football, corresponding to players from both teams and the ball itself). To estimate the distribution of player behaviors, we use a variational approach, relying on a GraphNet encoder, prior, and decoder. The GraphNet encoder and prior take as input the agent hidden states, which are used to initialize their node features; as in typical variational training schemes, the encoder is also provided access to privileged information available only at training-time (in this case, the full ground truth state $\boldsymbol{x}_t$). The prior, by contrast, does not have access to this information, and is ultimately the model used at evaluation-time. Similar to typical variational approaches, we impose a Kullback-Leibler (KL) divergence term in our training loss to encourage consistency between the encoder and prior distributions (see (15) in our "Methods" section for details).

Following initialization of node features, message passing is conducted within both the GraphNet encoder and prior, which output variables parameterizing the latent distribution. In the message passing step, each graph node (agent) shares information with all other nodes via the graph edges; all node features are then updated given the shared information. Latent distributions are assumed to be Gaussian in our case, as this permits closed-form solution of the KL-divergence term in the training loss. Thus, the parameters output from the GraphNet encoder and prior correspond to the means and covariances of the latent Gaussian distributions. Latent variables are then sampled for each GraphNet node (i.e., agent) from these distributions, which summarize the state of play at the particular timestep t; these latent variables are denoted $\vec{\boldsymbol{z}}_t$ for the forward-directional update in Fig. 2. The GraphNet decoder subsequently maps these latent variables to estimates of agents' relative state changes (e.g., $\Delta\vec{\boldsymbol{x}}_t$ for forward-direction updates), which are added to the absolute agent states from the previous timestep, $\vec{\boldsymbol{x}}_{t-1}$, thus resulting in a directional state estimate $\vec{\boldsymbol{x}}_t$. An analogous approach is used for computing a backward-directional state estimate, $\overleftarrow{\boldsymbol{x}}_t$. These direction-specific predictions are then fused to compute the final bidirectional updates for the agents, $\hat{\boldsymbol{x}}_t$. In our ablative experiments (in the Supplementary Information), we test two forms of forward–backward fusion: mean-fusion (simply taking the mean of directional estimates to compute the final prediction) and nearest-weighted fusion (weighing each direction according to the nearest ground truth observation available in that direction). Both of these fusion modes are detailed in the "Methods" section.

Our overall approach autoregressively estimates agent states, using available information in both directions. The model is inherently designed to handle noisy data through two means. First, the bidirectional nature of the model helps ensure it uses information available in future timesteps to correct for such noise. Second, the model is designed to handle noisy data due to its variational nature; namely, the model itself generates noisy autoregressive predictions during its imputation phase, which capture the distribution over input noise, and can thus lead to generation of diverse samples of trajectory outputs. The trajectory sequences generated can subsequently be used for downstream football analytics, as illustrated in our experiments.

Evaluation.

In this section, we empirically evaluate the Graph Imputer against a range of existing models for trajectory prediction.

Dataset. We use a dataset of 105 English Premier League matches, where all on-pitch players and the ball are tracked at 25 frames-per-second for each match. We partition the data into trajectory sequences of 240 frames (each capturing 9.6s of gameplay), then downsample the data to 6.25 frames-per-second. For training purposes, we retain only trajectories with 22 players available in the raw data (such that we can compute losses against all players' ground truth). The data is spatially realigned such that the team in possession always moves towards the right of the pitch (as done in prior works[25]). Finally, for training and evaluation, we split the resulting data into two partitions of 30838 and 4024 trajectories, respectively.

Simulated camera model. We use a simulated camera model to generate an observation mask for the task of off-screen player trajectory imputation. The camera model is parameterized by its position and horizontal and vertical field of view angles, with the parameters chosen to produce a vantage point similar to a stadium broadcast camera. For simplicity, the camera-normal is set to track the ball position at each timestep. By intersecting the camera view cone with the pitch plane, we obtain the projected in-frame polygon and mask out-of-frame players accordingly (as in Fig. 1). Further details on the camera model and levels of partial observability imposed due to it are provided in the "Methods" section.

Baselines. We compare our approach against the following baselines. *Spline:* Linear, quadratic, and cubic spline interpolation of players' positions from the moment they leave the the camera field of view to the moment they return; these approaches are simple, though can exhibit reasonable performance as they ensure the predicted trajectories adhere to the boundary value constraints imposed by the last observation of each player prior to going off-screen, and their first observation upon re-emergence on-screen. *Autoregressive LSTMs:* A simple baseline using autoregressive LSTMs, run independently per player for state estimation. *Role-invariant VRNNs:* A strong variational baseline that we hand-craft for the football scenario (i.e., assuming two teams of an equal number of players), using VRNNs and a combination of post-processing steps to ensure information-sharing between players on each team, and invariance of model outputs to re-ordering of players in inputs. Refer to the Additional Experiment Details section of the Supplementary Information for further information. *Social LSTM*[7]*:* A model that uses 'social pooling', which is a technique that pools hidden states of neighboring agents to ensure spatially-nearby context is appropriately shared between individual agents. *Bidirectional Social LSTMs:* We also implement a bidirectional Social LSTM variant using a combination of the vanilla Social LSTM updates and the fusion Eqs. (1), (2), (13) and (14) detailed in our "Methods" section, which we have not observed being used in the literature for our problem regime. *GVRNNs*[12]*:* A model that uses a combination of unidirectional VRNNs and Graph Neural Networks (similar in nature to Graph-VRNNs[3]), but assumes full observability of players for a fixed number of timesteps and targets the forward-prediction setting.

Trajectory prediction analysis. Table 1 provides a summary of results for the football off-screen player data imputation regime, including ablations over key model features where applicable. Training and hyperparameter details are provided in the "Methods" section. As noted earlier, the role-invariant models (listed in the first several table rows) are hand-crafted for the football case, and thus are not applicable to general multiagent settings; nonetheless, these models pose a strong evaluation baseline, and outperform several of the more generic approaches. Our proposed model, the Graph Imputer, outperforms the baselines both in terms of the mean and minimum evaluation loss over prediction samples, including the hand-crafted models.

As evident from Table 1, bidirectionality naturally yields a significant improvement in terms of overall performance across the models, as both past and future information is used in estimating player positions when off-screen. This is quantitatively evident even for the linear spline baseline, which is effectively bidirectional as it interpolates the last appearance and first reappearance of each player. While quadratic and cubic interpolation increase performance compared to the linear baseline, our model outperforms them significantly. Despite the additional context provided by past and future observations, such interpolation methods have no understanding of the dynamics of the domain (i.e., football in this case). As such, they can particularly suffer from a decrease in accuracy in situations where off-screen players behave defensively or offensively, exhibit sudden movements, or are off-screen for extended periods of time, which is not well-captured by interpolation. We also anticipate that situations with increased partial observability will further compound these issues with standard interpolation techniques. To further investigate the effects of increased partial observability, we generated a new dataset to test the sensitivity of the models to such changes (see the Sensitivity to Observability Model section in the Supplementary Information for numerical results, which illustrate the robustness of our model to these factors).

	Model	Skip connection	Next-step conditional decoder	$\mathcal{L}_2$(Mean)		$\mathcal{L}_2$(Min.)	
Restricted	Role-invariant VRNN	✗	–	2.020	2.03	1.960	2.063
	Role-invariant VRNN	✓	–	0.958	0.009	0.953	0.009
	Bidir. Role-invariant VRNN	✗	–	0.174	0.002	0.160	0.002
	Bidir. Role-invariant VRNN	✓	–	0.167	0.002	0.166	0.002
General	Spline (linear)	–	–	0.658	0.081	–	
	Spline (Quadratic)	–	–	0.197	0.023	–	
	Spline (Cubic)	–	–	0.193	0.023	–	
	LSTM	–	–	1.579	0.019	–	
	Bidir. LSTM	–	–	0.350	0.006	–	
	Social LSTM[7]	–	–	1.049	0.274	–	
	Bidir. Social LSTM	–	–	0.198	0.052	–	
	GVRNN[12]	✗	✗	2.243	0.136	1.453	0.073
	GVRNN[12]	✗	✓	2.447	1.197	2.400	1.231
	GVRNN[12]	✓	✗	0.882	0.009	0.874	0.009
	GVRNN[12]	✓	✓	0.865	0.018	0.852	0.017
	Graph Imputer (Ours)	✗	✗	0.241	0.05	0.224	0.051
	Graph Imputer (Ours)	✗	✓	0.404	0.102	0.397	0.11
	Graph Imputer (Ours)	✓	✗	0.165	0.005	0.163	0.005
	Graph Imputer (Ours)	✓	✓	**0.153**	**0.003**	**0.151**	**0.003**

Table 1. Football off-screen player state estimation results. We separate models into two categories: restricted models (those that apply only to the football setting, as they process data in a manner explicitly assuming two teams of players, along with a ball), and general models (models that apply to general multiagent prediction settings). The remaining columns refer to the following: *Skip connection* indicates whether a skip-connection from the input to the decoder is enabled for autoencoder based models. *Next-step conditional decoder* indicates whether decoders in graph network-based models condition on available next-timestep observations, as additional context. For each baseline model, we compute the mean evaluation loss, $\mathcal{L}_2$(Mean), compared to the ground truth trajectories (over all seeds). For stochastic models, for each evaluation sequence we also take 6 samples of imputed trajectories, and also report the minimum evaluation loss, $\mathcal{L}_2$(Min.), over all samples, averaged over all seeds. Best results are in bold.

Considering the more complex baselines in Table 1, we note that the unidirectional Social LSTM model is outperformed by the strongest-performing GVRNN model (which uses both a skip connection and next-step conditioned graph decoder), as observed in earlier literature. However, as the Social LSTM is fundamentally deterministic in nature, it cannot be used for sampling multiple viable player trajectories. We additionally observe that use of a skip connection from inputs to the decoder results in a significant improvement in results, for all variational models considered. While use of a next-step conditioned graph decoder slightly improves results for the Graph Imputer, it has a more significant impact on the GVRNN model, which we conjecture is due to the former model's bidirectional nature already providing significant information about future observations.

Figure 3 provides static visualizations of trajectory results for several example sequences, with more examples included in the Additional Experiment Results section of the Supplementary Information. We recommend readers view our animated visualizations on our website (https://sites.google.com/view/imputation-of-football/), which also illustrate the simulated camera model. In Fig. 3, observed trajectory segments for the attacking and defending team are, respectively, illustrated in dark blue and red, with the ball trajectory indicated in black. In the first row of the figure, we illustrate the portion of player trajectories that are unobserved in light blue and pink for each team, respectively. Recall that the observations provided to the models are the raw positions available for in-camera players, with the camera tracking the ball in each timestep. Well-performing models will, ideally, learn the key behavioral characteristics of player interactions and physics (e.g., velocities, constraints on acceleration, player turning radii, etc.) given the available positional information to make realistic predictions. The subsequent rows illustrate the predictions made by both the GVRNN model and the Graph Imputer, under the same observations. Notably, the bidirectional nature of our Graph Imputer approach enables predictions to not only more accurately model the flow of movement of players on the pitch, but also to appropriately adhere to the boundary value constraints imposed by players when they appear back in the camera view. For additional experiments, including numerous visualizations over baseline models and ablations over the bidirectional fusion modes discussed in "Methods", refer to the Additional Experiment Results section of the Supplementary Information.

Pitch control analysis. A key benefit of our model is that it can enable the downstream use of well-established football analytics models that rely on full player tracking information. In situations where this is infeasible (e.g., where player tracking information is obtained from broadcast video footage, as opposed to proprietary sensors installed in stadiums), such downstream models can no longer be applied. However, through the imputation of off-screen player trajectories, our approach enables the applicability of such models. Moreover, due to the bidirectional nature of our model, the predicted trajectories are significantly more useful for post-hoc analytics

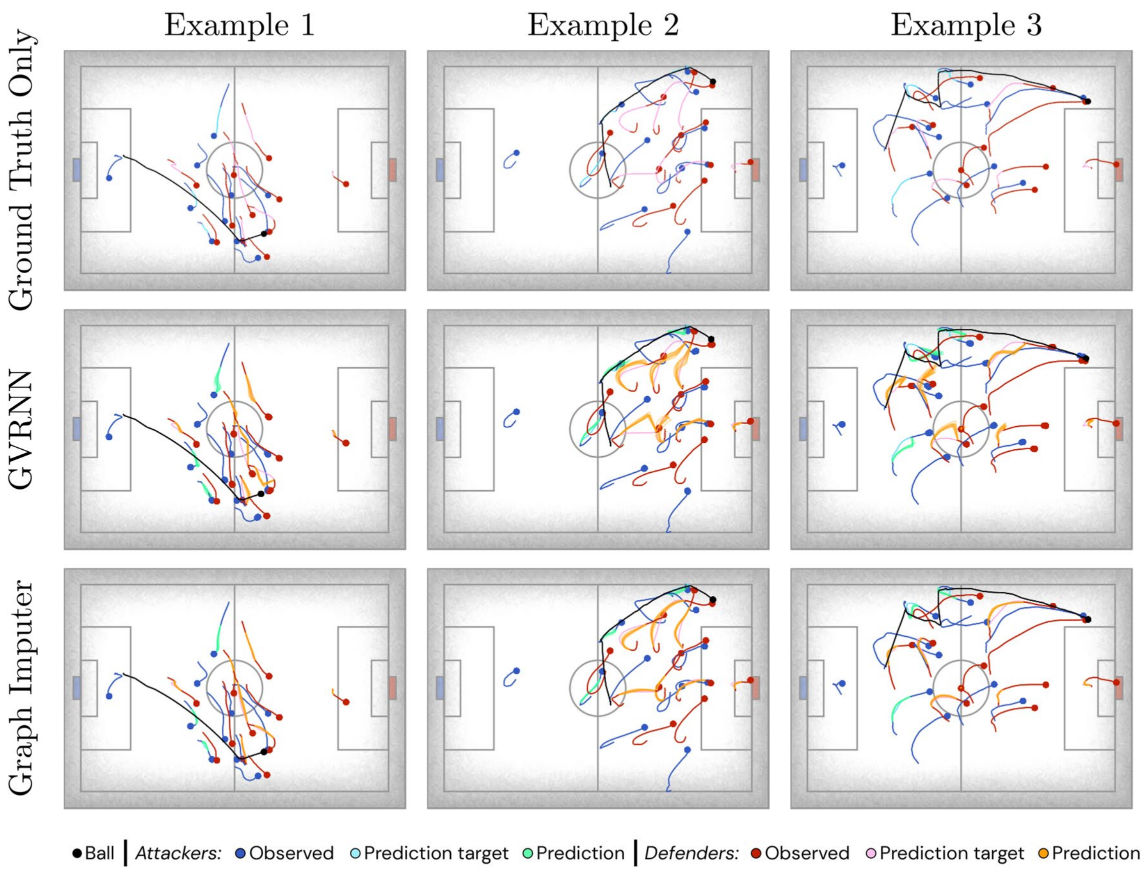

Figure 3. Trajectory visualizations (best viewed when zoomed in). Each column provides an example trajectory sequence, with the first row illustrating the ground truth, and subsequent rows showing results from various models, including the Graph Imputer (ours). For all examples, the Graph Imputer trajectories seamlessly adhere to the boundary value constraints imposed at the moments of disappearance and reappearance of players.

in comparison to standard forward-predictive approaches. To illustrate this, here we run experiments evaluating the performance of our approach when used to compute pitch control[14], which is a well-known model used for trajectory-based football analytics. Pitch Control, at a high-level, is a technique for computing the probability of each player (or team) gaining control of the ball were it to be passed to any location on pitch. For details of the pitch control model itself, we refer readers to the "Methods" section.

To compare models in terms of application to downstream analytics, we first compute the ground truth pitch control *without* any partial observability for all trajectories in our validation dataset. This results in a pitch control field at each timestep of each trajectory, which is a scalar field over the pitch (see examples in Fig. 4—left column). We subsequently consider pitch control under partial observability by occluding player positions as in our previous experiments, generate imputed trajectories using the various trained models, and compute model-specific pitch control fields. We then compute the Mean Absolute Error (MAE) between the ground truth and predicted Pitch Control fields, averaging spatially (across the pitch) and temporally (along the entire trajectory sequence). We report the pitch control MAE across all trajectories in Table 2. Notably, our Graph Imputer model yields the lowest error across all baselines. While the Bidirectional Role-invariant VRNN model comes close in terms of performance, we note that this was also a model carefully handcrafted by us for the football setting. Given its generality, the performance of the Graph Imputer is notable here.

To better understand differences qualitatively, in Fig. 4 we visualize several examples of the predicted and ground truth pitch control fields. Each row of the figure corresponds to a distinct game scenario from our validation dataset. The left column visualizes ground truth player positions for both teams (blue and red points), the camera field-of-view (semi-transparent white overlay), and the ground truth pitch control field. The remaining columns visualize the absolute error between pitch control fields based on predicted model outputs and ground truth. The example in the first row involves a scenario where the majority of players are visible within the camera field-of-view. As such, the pitch control predictions from the GVRNN, Bidirectional Social LSTM, and Graph Imputer are fairly consistent. Nonetheless, the region of the pitch corresponding to the blue team's goalkeeper (towards the right side of the pitch) has notably higher Pitch control error for the GVRNN compared to the latter models. Such a mis-estimation can have quite detrimental side-effects if used for tactical decision-making in games, especially as determination of pitch control in regions near each goalkeepers is crucial for determining goal-scoring opportunities (as illustrated in Spearman[26]). The example in the second row involves a larger number of off-screen players. Here we can see a noticeably high error in the GVRNN model near the bottom-right region of the pitch. Moreover, the Bidirectional Social LSTM model also has higher error than ours near the top

	Model	Pitch control MAE
Restricted	Role-invariant VRNN	0.0447 ± 0.0204
	Bidir. Role-invariant VRNN	0.0209 ± 0.0096
General	LSTM	0.0592 ± 0.0274
	Bidir. LSTM	0.0296 ± 0.0139
	Social LSTM[7]	0.0474 ± 0.0216
	Bidir. Social LSTM	0.0219 ± 0.0098
	GVRNN[12]	0.0418 ± 0.0189
	Graph Imputer (Ours)	0.0207 ±0.0094

Table 2. Predicted vs. ground truth pitch control[14] Mean Average Error (MAE) across models, under partially observable settings. Mean and standard deviations are reported over all trajectories in our validation dataset. The Graph Imputer model yields the lowest pitch control error across all baselines. Note that the Bidir. Role-invariant VRNN model, which comes closest to our Graph Imputer model in terms of performance, was handcrafted by us specifically for the football domain.

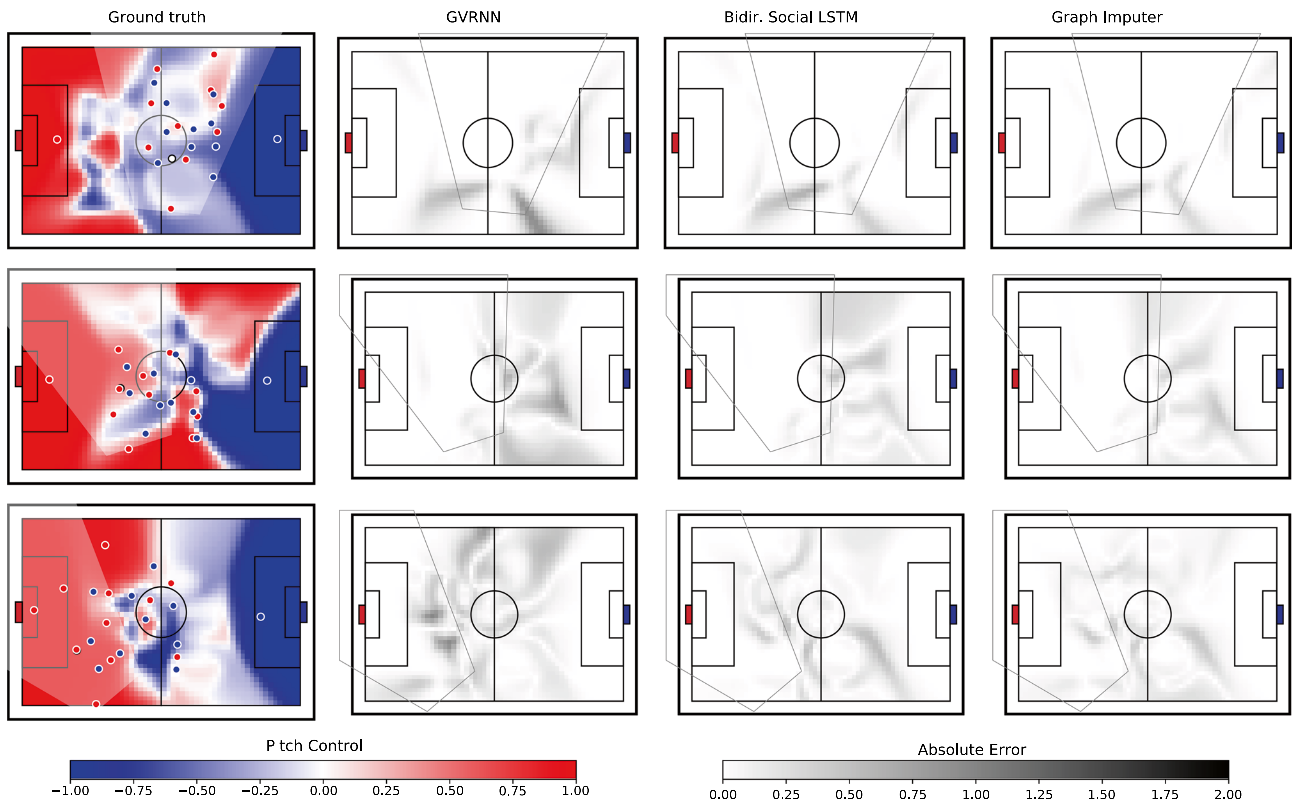

Figure 4. Pitch control error visualizations. The first column shows the ground truth pitch control field, player positions, and the camera field of view. Each remaining column provides a visualization of the absolute error between pitch control fields based on predicted model outputs and ground truth.

region of the pitch. The third example further exacerbates these errors, involving a situation where the camera is pointed towards the far left side of the pitch, with half of the players not visible on-screen. Here we see not only a high pitch control error for the GVRNN, but even regions with particularly high error *within the camera field-of-view* for the Bidirectional Social LSTM (see regions directly in front of the red goalkeeper). Such levels of error in on-screen regions are not prominent in the Graph Imputer model, which instead exhibits slightly larger error than the Bidirectional Social LSTM in a small region towards the bottom-right of the pitch. For interested readers, we additionally include animated visualizations of Pitch Control fields on our submission website.

Overall, our evaluations are indicative of the Graph Imputer model's performance not only in terms of raw trajectory prediction, but also for downstream use-cases such as pitch control-based analysis.

Related work

There exist a number of works from various fields of research that are related to our approach. Specifically, a number of works from robotics and computer vision[7–9,27], sports analytics[11–13,16,18], economics[4–6] and machine learning[1–3] focus on various combinations of missing data imputation and multiagent trajectory predictions. Given the broad scope of time series prediction as a research field[28], we focus particularly on models that predict human trajectories[9], as they are the most relevant for our problem regime. We also provide a table summarizing key similarities and differences of the most-closely related models in the Additional Details on Related Works section of the Supplementary Information.

One of the most common applications within human trajectory prediction (albeit not directly related to sports), is pedestrian modeling[7,29]. More closely related to our work are models that predict the trajectories of athletes in a team, such as basketball[16,17,19,30] or football[10–12]. Efforts that focus on the latter vary in the way they treat the interactions between players. While some models directly use the information about the players as conditioning for imitation learning[10,11], others use more complex interaction models such as graph networks for forward-prediction[3,12]. Our approach builds on the related works of Yeh et al.[12] and Sun et al.[3], which operate in the regime of predicting forward-rollouts of trajectories. As noted in Yang et al.[31] and Shang et al.[32], the models in Sun et al.[3] and Yeh et al.[12] are quite similar to one another, combining Graph Networks with VRNNs. In our comparisons and baselines table, we cite Yeh et al.[12] as we followed their implementation details most closely. Nonetheless, the problem setting considered by Sun et al.[3] and Yeh et al.[12] is quite different from ours. They consider settings involving forward-rollouts of agent trajectories (i.e., observations of agents are made for some number of timesteps, and forward forecasts are conducted thereafter). Our setting, by contrast, involves interacting agents that pop in and out of view sporadically, and bidirectional use of temporal information at test-time, which has not been considered in past works to our knowledge. The main limitation of prior works in application to this regime is that their forward-rollout predictions deviate over time, and thus do not adhere to the constraints imposed by future ground truth observations. An analogous comparison would be Kalman Filtering (using only past observations) versus Kalman Smoothing (using both past and future observations); these share core similarities, but are both independently useful in entirely different settings.

Mohamed et al.[33], Salzmann et al.[34] focus on forward-predictions of pedestrian trajectories. Mangalam et al.[35] also considers future-trajectory prediction, by first learning a distribution of endpoints and subsequently sampling from it. Kipf et al.[36] focus on forward-predictions of interacting physical systems, while Graber and Schwing[37] uses backward-information to update the approximate posterior during training, it still targets the forward-prediction regime at test-time (e.g., as noted in Fig. 8 of Graber and Schwing[37]). Thus, as mentioned earlier, these prior works consider only future predictions and not the imputation regime considered in our paper, which we believe is worthy of independent investigation. Finally, despite being framed as a supervised learning problem rather than sequence prediction, Hoshen[15] also take into account the interactions between the different variables in their multivariate trajectory prediction problem by using interaction networks.

Rather than targeting the forward-prediction regime, the goal of our model is to carry out imputation of incomplete time series involving multiple interacting agents. Imputation of sequential data itself can be treated as a means to an end for a separate task such as classification[38]. Also related to our line of work is prior work on a bidirectional model that carries out trajectory imputation[39]. Unlike ours, however, their approach does not target specifically the multiagent setting, though applies to the regime by essentially treating it as a large single-agent scenario. Finally, approaches that focus more directly on the imputation task itself include GAN-based models[40,41] and bidirectional inference models[42,43].

Conclusion

In this paper, we considered the problem regime of multiagent time-series imputation, which has not been formally analyzed in the literature, in contrast to the forward-predictive regime that has been significantly studied. Our introduced approach, called the Graph Imputer, uses a combination of bidirectional recurrent models to ensure use of all available temporal information, and graph networks to model inter-agent relations. Our experiments focused on the football analytics regime to illustrate a practical and intuitive real-world use case of such models. We illustrated that our approach outperforms several state-of-the-art methods on a large dataset of football tracking data, and qualitatively yields trajectory samples that capture player interactions and adhere to the constraints imposed by available observations. We subsequently illustrated how our approach can be used for unlocking downstream application of more complex analytical tools, such as Pitch control[14], which have traditionally relied on availability of fully-observed data. Empirical results illustrated that our approach outperforms strong baselines both in terms of raw trajectory prediction performance, and also in terms of these latter football-specific metrics, even outperforming models specifically handcrafted for the football regime.

One of the limitations of our model is that the forward and backward-direction latent vectors, $\overrightarrow{z}$ and $\overleftarrow{z}$, are sampled independently in our model; sampling these from a joint underlying distribution could significantly improve correlations in the directional predictions. Moreover, our model requires observations of each agent for at least a single timestep throughout each trajectory. While this is not a major limitation given long enough trajectory sequences in practice, investigating a means of enabling the model to seamlessly handle *completely missing* agents would increase its generality. An additional improvement to the model could be incorporation of distributional information (namely, the predicted covariance matrices) in the bidirectional fusion methods used, leveraging ideas from existing information filtering techniques. Finally, note that the training dataset constructed in our approach replicates the setup of related approaches (e.g.[12]). The diversity of behaviors expressed even within a single game makes the prediction problem challenging even with such a dataset split. Nonetheless, it may be interesting in future work to consider transfer learning situations (involving training on one set of games, and evaluating on a hold-out set), which are likely to be of interest in situations involving vision-in-the-loop (e.g., where camera systems are used to track players with varying uniforms, across different leagues).

Overall, the key benefit of our approach is its generality, in the sense that it permits any subset of agents to be unobserved at any timestep, works with temporal occlusions of arbitrary time horizons, and can apply directly to general multiagent domains beyond football. Predictive modeling of human trajectories is a complex problem with numerous applications, such as sports analytics, pedestrian modeling on roads, and crowd modeling in stadia. Given the fairly general nature of our approach, a key avenue of future work will be to apply it to these related regimes of predictive modeling.

Methods

This section provides technical details of our approach, called the Graph Imputer. All methods and research were performed in accordance with relevant guidelines/regulations of Nature Research journals. Algorithm 1 provides the associated pseudocode, and Fig. 2 illustrates the approach at a high level. We next define the specific components of our model in detail.

Algorithm 1 Graph Imputer Pseudocode

```
1:  function DirectionalUpdate($\boldsymbol{x}_t, \boldsymbol{m}_t, \hat{\boldsymbol{x}}_t$)
2:      Update autoregressively-filled state at current timestep $t$ via (1)
3:      Update LSTM states via (2)
4:      Sample prior and posterior latent states via (6) and (7)
5:      $\Delta\hat{\boldsymbol{x}}' \leftarrow$ Sample relative state update for next timestep via (8)
6:      $\hat{\boldsymbol{x}}' \leftarrow$ accumulate $\Delta\hat{\boldsymbol{x}}'$ via (12)
7:      return $\hat{\boldsymbol{x}}'$

8:  function GraphImputer($\boldsymbol{x}, \boldsymbol{m}$)
9:      Initialize network parameters, initial directional estimates $\hat{\vec{\boldsymbol{x}}}_0$ and $\hat{\overleftarrow{\boldsymbol{x}}}_T$ to ground truth
10:     for iteration $\leftarrow 1$ to $K$ do
11:         for $t \leftarrow 0$ to $T-1$ do
12:             $\hat{\vec{\boldsymbol{x}}}_{t+1} \leftarrow$ DirectionalUpdate($\boldsymbol{x}_t, \boldsymbol{m}_t, \hat{\vec{\boldsymbol{x}}}_t$)
13:         for $t \leftarrow T$ to $1$ do
14:             $\hat{\overleftarrow{\boldsymbol{x}}}_{t-1} \leftarrow$ DirectionalUpdate($\boldsymbol{x}_t, \boldsymbol{m}_t, \hat{\overleftarrow{\boldsymbol{x}}}_t$)
15:         $\hat{\boldsymbol{x}} \leftarrow$ Fuse the forward-backward estimates $\hat{\vec{\boldsymbol{x}}}$ and $\hat{\overleftarrow{\boldsymbol{x}}}$ via (13) or (14)
16:         Update model parameters via gradient step on ELBO (15)
17:     return $\hat{\boldsymbol{x}}$
```

Bidirectional autoregression. The key distinction between our problem regime and that of many prior multiagent predictive modeling approaches is that we target the more general imputation setting, involving both future and past contextual information about subsets of various agents. The temporal backbone of our model is, thus, a bidirectional autoregressive LSTM, which leverages all available information at the time of prediction.

Specifically, at each time t, let $\vec{\boldsymbol{x}}_t$ and $\overleftarrow{\boldsymbol{x}}_t$ denote the forward- and backward-direction inputs to the model. These inputs correspond to the combination of ground truth states, $\boldsymbol{x}_t$, for observed agents, and autoregressively-predicted states, $\hat{\vec{\boldsymbol{x}}}_t$ and $\hat{\overleftarrow{\boldsymbol{x}}}_t$ (defined below), for unobserved agents, as follows:

$$\vec{\boldsymbol{x}}_t = \boldsymbol{x}_t \odot \boldsymbol{m}_t + \hat{\vec{\boldsymbol{x}}}_t \odot (1 - \boldsymbol{m}_t) \qquad \overleftarrow{\boldsymbol{x}}_t = \boldsymbol{x}_t \odot \boldsymbol{m}_t + \hat{\overleftarrow{\boldsymbol{x}}}_t \odot (1 - \boldsymbol{m}_t). \tag{1}$$

We use bidirectional LSTMs to temporally-integrate observation sequences and learn the forward- and backward-dynamics involved. Agent-wise hidden states, $\vec{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$, are updated as follows:

$$\vec{\boldsymbol{h}}^i_t = \mathrm{LSTM}_{\vec{\boldsymbol{\omega}}}(\vec{\boldsymbol{x}}^i_t, \vec{\boldsymbol{h}}^i_{t-1}) \qquad \overleftarrow{\boldsymbol{h}}^i_t = \mathrm{LSTM}_{\overleftarrow{\boldsymbol{\omega}}}(\overleftarrow{\boldsymbol{x}}^i_t, \overleftarrow{\boldsymbol{h}}^i_{t+1}) \quad \forall i \in \mathbb{I}, \tag{2}$$

where $\vec{\boldsymbol{\omega}}$ and $\overleftarrow{\boldsymbol{\omega}}$ refer to direction-specific LSTM parameters, which are shared across agents.

We next detail the computation of the autoregressively-predicted states $\hat{\vec{\boldsymbol{x}}}$ and $\hat{\overleftarrow{\boldsymbol{x}}}$ appearing in (1), which are sampled from a variational graph network capturing multiagent interactions in the system.

Graph networks. We define a graph network consisting of N nodes, each corresponding to an agent or entity in the system (e.g., $N = 23$ in the football domain, capturing the state of the 22 players and the ball). Let $\boldsymbol{v}^i_t$ denote the node feature vector associated with an agent $i \in \mathbb{I}$, which encodes its spatiotemporal context at time t. Likewise, let $\boldsymbol{e}^{(i,j)}$ denote the directed edge feature connecting agent $i \in \mathbb{I}$ to agent $j \in \mathbb{I}$. Graph networks operate via rounds of message passing, which update edge and node features to propagate information across the various nodes involved. In our instance, the message-passing update is expressed as follows,

$$\boldsymbol{e}'^{(i,j)} = f^e_{\boldsymbol{\theta}}(\boldsymbol{v}^i, \boldsymbol{v}^j) \qquad \text{(Update edges from sender nodes } i \in N^-(j) \text{ to recipients } j \in \mathbb{I}\text{)}, \tag{3}$$

$$\boldsymbol{e}'^{j} = \sum_{i \in N^{-}(j)} \boldsymbol{e}'^{(i,j)} \qquad \text{(Aggregate incoming edges for all receiver nodes } j \in \mathbb{I}\text{)}, \tag{4}$$

$$\boldsymbol{v}'^{j} = f^{v}_{\boldsymbol{\theta}}(\boldsymbol{e}'^{j}) \qquad \text{(Update all receiver nodes } j \in \mathbb{I}\text{)}, \tag{5}$$

where $N^{-}(j)$ are in-neighbors of node j, and $f^{e}_{\boldsymbol{\theta}}$ and $f^{v}_{\boldsymbol{\theta}}$ are, respectively, edge and node update functions with learned parameters $\boldsymbol{\theta}$. In shorthand, given an initial set of node features $\boldsymbol{v}$, we refer to the updated features following the message-passing steps in (3) to (5) as $\boldsymbol{v}' = \mathrm{GN}_{\boldsymbol{\theta}}(\boldsymbol{v})$.

Variational updates. At any time t, the history of autoregressively-filled directional inputs, $\overrightarrow{\boldsymbol{x}}_{<t}$ and $\overleftarrow{\boldsymbol{x}}_{>t}$, is encoded by the LSTM states $\boldsymbol{h}_{t-1}$ and $\boldsymbol{h}_{t+1}$. Conditioned on this context, our model uses variational graph networks to enable information-sharing across agents, and learn a distribution over latent random variables $\boldsymbol{z}$ and predicted state updates $\Delta\hat{\boldsymbol{x}}_t$. Specifically, the graph imputer learns to approximate the directional prior distributions $p_{\boldsymbol{\theta}}(\overrightarrow{\boldsymbol{z}}_t|\cdot)$ and $p_{\boldsymbol{\theta}}(\overleftarrow{\boldsymbol{z}}_t|\cdot)$, posterior distributions $q_{\boldsymbol{\phi}}(\overrightarrow{\boldsymbol{z}}_t|\cdot)$ and $q_{\boldsymbol{\phi}}(\overleftarrow{\boldsymbol{z}}_t|\cdot)$, and decoded output distribution $p_{\boldsymbol{\theta}}(\Delta\overrightarrow{\boldsymbol{x}}_t|\cdot)$ and $p_{\boldsymbol{\theta}}(\Delta\overleftarrow{\boldsymbol{x}}_t|\cdot)$, as follows,

$$p_{\boldsymbol{\theta}}(\overrightarrow{\boldsymbol{z}}^{i}_{t}|\overrightarrow{\boldsymbol{x}}_{<t}, \overrightarrow{\boldsymbol{z}}_{<t}) = \mathcal{N}\left(\overrightarrow{\boldsymbol{\mu}}^{i}_{\mathrm{pri},t}, (\overrightarrow{\boldsymbol{\sigma}}^{i}_{\mathrm{pri},t})^2\right) \quad p_{\boldsymbol{\theta}}(\overleftarrow{\boldsymbol{z}}^{i}_{t}|\overleftarrow{\boldsymbol{x}}_{>t}, \overleftarrow{\boldsymbol{z}}_{>t}) = \mathcal{N}\left(\overleftarrow{\boldsymbol{\mu}}^{i}_{\mathrm{pri},t}, (\overleftarrow{\boldsymbol{\sigma}}^{i}_{\mathrm{pri},t})^2\right), \tag{6}$$

$$q_{\boldsymbol{\phi}}(\overrightarrow{\boldsymbol{z}}^{i}_{t}|\boldsymbol{x}_t, \overrightarrow{\boldsymbol{x}}_{<t}, \overrightarrow{\boldsymbol{z}}_{<t}) = \mathcal{N}\left(\overrightarrow{\boldsymbol{\mu}}^{i}_{\mathrm{enc},t}, (\overrightarrow{\boldsymbol{\sigma}}^{i}_{\mathrm{enc},t})^2\right) \quad q_{\boldsymbol{\phi}}(\overleftarrow{\boldsymbol{z}}^{i}_{t}|\boldsymbol{x}_t, \overleftarrow{\boldsymbol{x}}_{>t}, \overleftarrow{\boldsymbol{z}}_{>t}) = \mathcal{N}\left(\overleftarrow{\boldsymbol{\mu}}^{i}_{\mathrm{enc},t}, (\overleftarrow{\boldsymbol{\sigma}}^{i}_{\mathrm{enc},t})^2\right), \tag{7}$$

$$p_{\boldsymbol{\theta}}(\Delta\hat{\overrightarrow{\boldsymbol{x}}}^{i}_{t}|\overrightarrow{\boldsymbol{x}}_{<t}, \overrightarrow{\boldsymbol{z}}_{\leq t}) = \mathcal{N}\left(\overrightarrow{\boldsymbol{\mu}}^{i}_{\mathrm{dec},t}, (\overrightarrow{\boldsymbol{\sigma}}^{i}_{\mathrm{dec},t})^2\right) \quad p_{\boldsymbol{\theta}}(\Delta\hat{\overleftarrow{\boldsymbol{x}}}^{i}_{t}|\overleftarrow{\boldsymbol{x}}_{>t}, \overleftarrow{\boldsymbol{z}}_{\geq t}) = \mathcal{N}\left(\overleftarrow{\boldsymbol{\mu}}^{i}_{\mathrm{dec},t}, (\overleftarrow{\boldsymbol{\sigma}}^{i}_{\mathrm{dec},t})^2\right). \tag{8}$$

In the above, (6) enables sampling of latent variables, $\overrightarrow{\boldsymbol{z}}_t$ and $\overleftarrow{\boldsymbol{z}}_t$, conditioned on the prior information available up to, though not including, the prediction timestep t. Likewise, (7) captures the posterior latent state distribution, conditioned on the same information as the prior *in addition to* the ground truth state $\boldsymbol{x}_t$. Finally, (8) enables sampling of a next-state prediction for each direction. As in typical VRNN-based approaches, the encoder is used only during training to sample latent states $\boldsymbol{z}_t$, which are used as inputs for the decoder; during evaluation, samples $\boldsymbol{z}_t$ from the prior are used instead, as the encoder can, naturally, no longer be used due to the ground truth state $\boldsymbol{x}_t$ being unavailable.

The collection of mean and variance parameters above, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$, parameterize underlying Gaussian distributions. These parameters simply correspond to node features output by underlying graph networks, which exchange information between agents following a message-passing step:

$$\left[\overrightarrow{\boldsymbol{\mu}}^{i}_{\mathrm{pri},t}, (\overrightarrow{\boldsymbol{\sigma}}^{i}_{\mathrm{pri},t})^2\right]_{i\in\mathbb{I}} = \mathrm{GN}_{\mathrm{pri},\boldsymbol{\theta}}\left(\overrightarrow{\boldsymbol{h}}_{t-1}\right) \quad \left[\overleftarrow{\boldsymbol{\mu}}^{i}_{\mathrm{pri},t}, (\overleftarrow{\boldsymbol{\sigma}}^{i}_{\mathrm{pri},t})^2\right]_{i\in\mathbb{I}} = \mathrm{GN}_{\mathrm{pri},\boldsymbol{\theta}}\left(\overleftarrow{\boldsymbol{h}}_{t+1}\right), \tag{9}$$

$$\left[\overrightarrow{\boldsymbol{\mu}}^{i}_{\mathrm{enc},t}, (\overrightarrow{\boldsymbol{\sigma}}^{i}_{\mathrm{enc},t})^2\right]_{i\in\mathbb{I}} = \mathrm{GN}_{\mathrm{enc},\boldsymbol{\phi}}\left(\left[\boldsymbol{x}_t, \overrightarrow{\boldsymbol{h}}_{t-1}\right]\right) \quad \left[\overleftarrow{\boldsymbol{\mu}}^{i}_{\mathrm{enc},t}, (\overleftarrow{\boldsymbol{\sigma}}^{i}_{\mathrm{enc},t})^2\right]_{i\in\mathbb{I}} = \mathrm{GN}_{\mathrm{enc},\boldsymbol{\phi}}\left(\left[\boldsymbol{x}_t, \overleftarrow{\boldsymbol{h}}_{t+1}\right]\right), \tag{10}$$

$$\left[\overrightarrow{\boldsymbol{\mu}}^{i}_{\mathrm{dec},t}, (\overrightarrow{\boldsymbol{\sigma}}^{i}_{\mathrm{dec},t})^2\right]_{i\in\mathbb{I}} = \mathrm{GN}_{\mathrm{dec},\boldsymbol{\theta}}\left(\left[\overrightarrow{\boldsymbol{z}}_t, \overrightarrow{\boldsymbol{h}}_{t-1}\right]\right) \quad \left[\overleftarrow{\boldsymbol{\mu}}^{i}_{\mathrm{dec},t}, (\overleftarrow{\boldsymbol{\sigma}}^{i}_{\mathrm{dec},t})^2\right]_{i\in\mathbb{I}} = \mathrm{GN}_{\mathrm{dec},\boldsymbol{\theta}}\left(\left[\overleftarrow{\boldsymbol{z}}_t, \overleftarrow{\boldsymbol{h}}_{t+1}\right]\right). \tag{11}$$

Subsequent to their sampling in (8), the relative (delta) state updates, $\Delta\hat{\boldsymbol{x}}_t$, are accumulated to produce predictions in absolute-space,

$$\hat{\overrightarrow{\boldsymbol{x}}}_t = \overrightarrow{\boldsymbol{x}}_{t-1} + \Delta\hat{\overrightarrow{\boldsymbol{x}}}_t \quad \hat{\overleftarrow{\boldsymbol{x}}}_t = \overleftarrow{\boldsymbol{x}}_{t+1} + \Delta\hat{\overleftarrow{\boldsymbol{x}}}_t. \tag{12}$$

These predicted states $\hat{\overrightarrow{\boldsymbol{x}}}_t$ and $\hat{\overleftarrow{\boldsymbol{x}}}_t$ are then used to autoregressively update the next-timestep inputs using (1). The procedure then continues to autoregressively update the states for all timesteps t in each respective direction.

Forward–backward fusion. The final directional outputs from the model are subsequently fused to produce the bidirectional estimates $\hat{\boldsymbol{x}}^{i}_{t}$ for all agents. As in recent works on bidirectional LSTM-based imputation[42], one method of fusion is to simply take the mean,

$$\hat{\boldsymbol{x}}^{i}_{t} = 0.5\left(\hat{\overrightarrow{\boldsymbol{x}}}^{i}_{t} + \hat{\overleftarrow{\boldsymbol{x}}}^{i}_{t}\right). \tag{13}$$

Alternatively, at time t, let $\overrightarrow{\tau}^{i}_{t}$ and $\overleftarrow{\tau}^{i}_{t}$ denote the number of timesteps until the next ground truth observation in each direction, respectively. One can then weigh the contribution of each direction as,

$$\hat{\boldsymbol{x}}^{i}_{t} = \left(\overrightarrow{\tau}^{i}_{t}\hat{\overrightarrow{\boldsymbol{x}}}^{i}_{t} + \overleftarrow{\tau}^{i}_{t}\hat{\overleftarrow{\boldsymbol{x}}}^{i}_{t}\right)\left(\overrightarrow{\tau}^{i}_{t} + \overleftarrow{\tau}^{i}_{t}\right)^{-1}. \tag{14}$$

This ensures predictions corresponding to the direction with the most recent observation are weighted higher. For example, if at prediction timestep t, the nearest ground truth observations in the future and past for agent i occur at $t+8$ and $t-2$, then $\overrightarrow{\tau}_t = 8$ and $\overleftarrow{\tau}_t = 2$, such that $\hat{\boldsymbol{x}}^i_t = 0.8\overrightarrow{\boldsymbol{x}}_t + 0.2\overleftarrow{\boldsymbol{x}}_t$.

Training. As in prototypical VAE pipelines, we update model parameters in each iteration of the algorithm by maximizing the evidence lower bound (ELBO) over all the agents in each trajectory,

$$\sum_{t\in\mathbb{T}}\Bigg[\mathbb{E}_{q_{\boldsymbol{\phi}}(\overrightarrow{\boldsymbol{z}}_t|\boldsymbol{x}_t,\overrightarrow{\boldsymbol{x}}_{<t},\overrightarrow{\boldsymbol{z}}_{<t})}\Big[\log p_{\boldsymbol{\theta}}\Big(\Delta\hat{\overrightarrow{\boldsymbol{x}}}_t|\overrightarrow{\boldsymbol{x}}_{<t},\overrightarrow{\boldsymbol{z}}_{\le t}\Big)\Big]-\beta D_{KL}\Big(q_{\boldsymbol{\phi}}\Big(\overrightarrow{\boldsymbol{z}}_t|\boldsymbol{x}_t,\overrightarrow{\boldsymbol{x}}_{<t},\overrightarrow{\boldsymbol{z}}_{<t}\Big)||p_{\boldsymbol{\theta}}\Big(\overrightarrow{\boldsymbol{z}}_t|\overrightarrow{\boldsymbol{x}}_{<t},\overrightarrow{\boldsymbol{z}}_{<t}\Big)\Big)$$
$$+\mathbb{E}_{q_{\boldsymbol{\phi}}(\overleftarrow{\boldsymbol{z}}_t|\boldsymbol{x}_t,\overleftarrow{\boldsymbol{x}}_{>t},\overleftarrow{\boldsymbol{z}}_{>t})}\Big[\log p_{\boldsymbol{\theta}}\Big(\Delta\hat{\overleftarrow{\boldsymbol{x}}}_t|\overleftarrow{\boldsymbol{x}}_{>t},\overleftarrow{\boldsymbol{z}}_{\ge t}\Big)\Big]-\beta D_{KL}\Big(q_{\boldsymbol{\phi}}\Big(\overleftarrow{\boldsymbol{z}}_t|\boldsymbol{x}_t,\overleftarrow{\boldsymbol{x}}_{>t},\overleftarrow{\boldsymbol{z}}_{>t}\Big)||p_{\boldsymbol{\theta}}\Big(\overleftarrow{\boldsymbol{z}}_t|\overleftarrow{\boldsymbol{x}}_{>t},\overleftarrow{\boldsymbol{z}}_{>t}\Big)\Big)\Bigg], \tag{15}$$

where β is a weighing term on the VAE KL-regularizer[44]. For training, we maximize (15) over mini-batches of trajectories sampled from our dataset.

In our experiments, we also consider several ablations of the models, including: decoders that take as input only the latent states $\overrightarrow{\boldsymbol{z}}_t$ and $\overleftarrow{\boldsymbol{z}}_t$ (i.e., disabling the *skip-connection* from the LSTM hidden states $\overrightarrow{\boldsymbol{h}}_{t-1}$ and $\overleftarrow{\boldsymbol{h}}_{t+1}$ to the decoder in (11)); and *next-step conditioned* graph-decoders that include nodes with features $\boldsymbol{v}^i$ locked to agent observations available for the timestep being predicted (i.e., observed decoder nodes with features $\boldsymbol{x}^i_t \odot \boldsymbol{m}^i_t$, which only send messages during message-passing, and thus do not update their states at prediction timestep t as they are observable).

Sweeps and hyperparameters. We conduct a wide hyperparameter sweep and report the results corresponding to the best hyperparameters for each model. We train for 10^5 iterations, with a batch size of 64 trajectories, using the Adam optimizer[45] with a learning rate of 0.001 (and default exponential decay parameters, $b_1 = 0.9$, and $b_2 = 0.999$). For LSTM-based models (including the ones used in the Graph Imputer), we use 2-layer LSTMs with 64 hidden units each. For the graph edge and node update networks, $f^e_{\boldsymbol{\theta}}$ and $f^v_{\boldsymbol{\theta}}$, we use 2-layer MLPs with 64 hidden units each, with internal ReLU activations[46]. In the ELBO (15), we anneal β from an initial value of 0.1 to final values of 0.01 and 1 in our sweeps. All variational models use 16-dimensional latent variables $\boldsymbol{z}$. For all bidirectional models, we sweep over the two fusion modes specified in (13) and (14). For each model, training for each hyperparameter set is conducted and reported over a sweep of 5 random seeds. Additional hyperparameters and computational resources used are detailed in the Additional Experiment Details section of the Supplementary Information.

Camera model details. Given the camera model described in the main text, on average, 12.76 ± 3.70 players (out of 22) are in-frame in each sequence, with a consecutive in-frame duration of $4.94s \pm 3.49$ s. Under this camera model, certain players are at times completely out of view for the entire trajectory duration. To provide some warm-up context to the models during training, we include an additional 5 frames of observations at the beginning and end of all trajectories for all players. In practical evaluation settings involving longer trajectory sequences, the camera pans around such that all players are effectively observed at some stage, thus not requiring this.

Pitch control model. Pitch control[14] is a technique for quantifying a football team's level of ball control, throughout the pitch. At a high level, the model takes as input the state of play (i.e., the positions and velocities of the players and the ball), and computes the probability of each player being able to gain control of the ball were it to be passed to every location on pitch; in practice, the pitch is discretized such that these probabilities are computed over a finite number of locations (we use a 60×40 discretization of the pitch in our experiments, as this provides a fairly smooth view of the pitch control field without incurring significant computational expense).

In more detail, the pitch control model relies on a physics-based motion model, incorporating factors such as ball time-of-flight and player time-to-intercept to compute control probabilities. In our experiments, we use a slightly simplified variant, wherein at any given timestep in a trajectory we consider the ball being passed to each discrete pitch location under a constant reference speed (as detailed later). Subsequently, each player on the pitch is assumed to travel in a straight line to the ball's target destination at a reference top speed. Following these simplifications, we use the model detailed in Spearman et al.[14] to compute the player-wise pitch control probabilities. Next, to compute team-wise pitch control field, we simply sum up player-wise pitch control per team and compute the difference between the resulting scalar fields for home and away team. Both the ball and player reference speeds are fit to our data, by first computing all player and ball velocities throughout the dataset, and subsequently choosing the 75th percentile speed for each (in practice, we found the Pitch Control model to be fairly insensitive to the specific percentile chosen, as long as the ratio between ball and player speeds remained similar).

Data availibility

The datasets generated and/or analyzed during the current study are not publicly available due to licensing restrictions. However, contact details of the data providers are available from the corresponding authors on reasonable request.

Received: 24 January 2022; Accepted: 10 May 2022
Published online: 23 May 2022

References

1. Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. Counterfactual multi-agent policy gradients. In *Proc. Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2–7, 2018* (eds. McIlraith, S. A. & Weinberger, K. Q.) 2974–2982 (AAAI Press, 2018).
2. Brown, N., Lerer, A., Gross, S. & Sandholm, T. Deep counterfactual regret minimization. In *Proc. 36th International Conference on Machine Learning, ICML 2019, 9–15 June 2019, Long Beach, California, USA, Volume 97 of Proc. Machine Learning Research* (eds. Chaudhuri, K. & Salakhutdinov, R.) 793–802 (PMLR, 2019).
3. Sun, C., Karlsson, P., Wu, J., Tenenbaum, J. B. & Murphy, K. Predicting the present and future states of multi-agent systems from partially-observed visual data. In *International Conference on Learning Representations*. https://openreview.net/forum?id=r1xdH3CcKX (2019).
4. Taylor, S. J. *Modelling Financial Time Series. Number 6578 in World Scientific Books* (World Scientific Publishing, 2007).
5. Sezer, O. B., Gudelek, M. U. & Özbayoglu, A. M. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *CoRR*. http://arxiv.org/abs/1911.13288 (2019).
6. Tay, F. E. & Cao, L. Application of support vector machines in financial time series forecasting. *Omega* **29**(4), 309–317 (2001).
7. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L. & Savarese, S. Social LSTM: Human trajectory prediction in crowded spaces. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).
8. Sakata, N., Kinoshita, Y. & Kato, Y. Predicting a pedestrian trajectory using seq2seq for mobile robot navigation. In *IECON 2018—44th Annual Conference of the IEEE Industrial Electronics Society* 4300–4305 (2018).
9. Rudenko, A. *et al.* Human motion trajectory prediction: A survey. *Int. J. Robot. Res.* **39**(8), 895–935 (2020).
10. Le, H. M., Yue, Y., Carr, P. & Lucey, P. Coordinated multi-agent imitation learning. In *International Conference on Machine Learning* 1995–2003 (PMLR, 2017).
11. Le, H. M., Carr, P., Yue, Y. & Lucey, P. Data-driven ghosting using deep imitation learning. *MIT Sloan Sports Analytics Conference (SSAC)* (2017).
12. Yeh, R. A., Schwing, A. G., Huang, J. & Murphy, K. Diverse generation for multi-agent sports games. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* 4610–4619 (2019).
13. Hauri, S., Djuric, N., Radosavljevic, V. & Vucetic, S. Multi-modal trajectory prediction of NBA players. *arXiv* (2020).
14. Spearman, W., Basye, A., Dick, G., Hotovy, R. & Pop, P. Physics-based modeling of pass probabilities in soccer. In *Proc. 11th MIT Sloan Sports Analytics Conference* (2017).
15. Hoshen, Y. Vain: Attentional multi-agent predictive modeling. In *Advances in Neural Information Processing Systems* Vol. 30 (eds Guyon, I. *et al.*) (Curran Associates, Inc., 2017).
16. Alcorn, M. A. & Nguyen, A. baller2vec: A multi-entity transformer for multi-agent spatiotemporal modeling. Preprint at http://arXiv.org/2102.03291 (2021).
17. Su, S., Hong, J. P., Shi, J. & Park, H. S. Predicting behaviors of basketball players from first person videos. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017* 1206–1215 (IEEE Computer Society, 2017).
18. Suda, S., Makino, Y. & Shinoda, H. Prediction of volleyball trajectory using skeletal motions of setter player. In *Proc. 10th Augmented Human International Conference 2019, New York, NY, USA* (Association for Computing Machinery, 2019).
19. Li, J., Yang, F., Tomizuka, M. & Choi, C. Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning. In *Proc. Neural Information Processing Systems (NeurIPS)* (2020).
20. Moritz, S. & Bartz-Beielstein, T. Imputets: Time series missing value imputation in r. *R J.* **9**(1), 207 (2017).
21. Silva, I., Moody, G., Scott, D. J., Celi, L. A. & Mark, R. G. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. In *2012 Computing in Cardiology* 245–248 (IEEE, 2012).
22. Fedus, W., Goodfellow, I. & Dai, A. M. MaskGAN: Better text generation via filling in the _. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ByOExmWAb (2018).
23. Chung, J. *et al.* A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems* Vol. 28 (eds Cortes, C. *et al.*) (Curran Associates, Inc., 2015).
24. Battaglia, P. *et al.* Relational inductive biases, deep learning, and graph networks. *arXiv*. http://arxiv.org/abs/1806.01261 (2018).
25. Bialkowski, A., Lucey, P., Carr, P., Yue, Y., Sridharan, S. & Matthews, I. Large-scale analysis of soccer matches using spatiotemporal tracking data. In *2014 IEEE International Conference on Data Mining* 725–730 (IEEE, 2014).
26. Spearman, W. Beyond expected goals. In *Proc. 12th MIT Sloan Sports Analytics Conference* 1–17 (2018).
27. Zhu, H., Claramunt, F. M., Brito, B. & Alonso-Mora, J. Learning interaction-aware trajectory predictions for decentralized multi-robot motion planning in dynamic environments. *arXiv* (2021).
28. Han, Z., Zhao, J., Leung, H., Ma, K. F. & Wang, W. A review of deep learning models for time series prediction. *IEEE Sens. J.* **21**, 7833 (2019).
29. Sun, H., Zhao, Z. & He, Z. Reciprocal learning networks for human trajectory prediction. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).
30. Alcorn, M. A. & Nguyen, A. baller2vec++: A look-ahead multi-entity transformer for modeling coordinated agents. Preprint at http://arXiv.org/2104.11980 (2021).
31. Yang, F., Chen, L., Zhou, F., Gao, Y. & Cao, W. Relational state-space model for stochastic multi-object systems. In *International Conference on Learning Representations*. https://openreview.net/forum?id=B1lGU64tDr (2020).
32. Shang, W., Espeholt, L., Raichuk, A. & Salimans, T. Agent-centric representations for multi-agent reinforcement learning. Preprint at http://arXiv.org/2104.09402 (2021).
33. Mohamed, A., Qian, K., Elhoseiny, M. & Claudel, C. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* 14424–14432 (2020).
34. Salzmann, T., Ivanovic, B., Chakravarty, P. & Pavone, M. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proc., Part XVIII 16* 683–700 (Springer, 2020).
35. Mangalam, K. *et al.* It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *European Conference on Computer Vision* 759–776 (Springer, 2020).
36. Kipf, T., Fetaya, E., Wang, K.-C., Welling, M. & Zemel, R. Neural relational inference for interacting systems. In *International Conference on Machine Learning* 2688–2697 (PMLR, 2018).
37. Graber, C. & Schwing, A. G. Dynamic neural relational inference. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* 8513–8522 (2020).
38. Che, Z., Purushotham, S., Cho, K., Sontag, D. & Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Sci. Rep.* **8**(1), 1–12 (2018).

39. Liu, Y., Yu, R., Zheng, S., Zhan, E. & Yue, Y. Naomi: Non-autoregressive multiresolution sequence imputation. *Advances in Neural Information Processing Systems 32 (NIPS 2019)* (2019).
40. Luo, Y., Cai, X., Zhang, Y., Xu, J. & Yuan, X. Multivariate time series imputation with generative adversarial networks. In *Proc. 32nd International Conference on Neural Information Processing Systems* 1603–1614 (2018).
41. Luo, Y., Zhang, Y., Cai, X. & Yuan, X. *E2gan: End-to-End Generative Adversarial Network for Multivariate Time Series Imputation* 3094–3100 (AAAI Press, 2019).
42. Cao, W. *et al.* Brits: Bidirectional recurrent imputation for time series. In *Advances in Neural Information Processing Systems* Vol. 31 (eds Bengio, S. *et al.*) (Curran Associates, Inc., 2018).
43. Yoon, J., Zame, W. R. & van der Schaar, M. Estimating missing data in temporal data streams using multi-directional recurrent neural networks. *CoRR*. http://arxiv.org/abs/1711.08742 (2017).
44. Higgins, I. *et al.* beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations* (2017).
45. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings* (2015).
46. Nair, V. & Hinton, G. E. Rectified linear units improve restricted Boltzmann machines. In *ICML* 807–814 (2010).

Acknowledgements

The authors thank Mark Rowland and Julien Perolat for their helpful feedback on the paper draft. The authors are also grateful to the DeepMind football analytics team for their help and support on the overall research effort, including particularly Alexandre Galashov, Nathalie Beauguerlange, Jackson Broshear, and Simon Bouton.

Author contributions

S.O. and K.T. coordinated and organized the research effort leading to this article. S.O., D.H., M.G., Z.W., A.R., E.T., Y.Y., R.E., J.C., P.M., N.M., K.C., P.M., P.S., D.H., I.G., W.S., N.H., K.T. contributed to technical discussions, implementations, post-hoc analysis of results, and/or writing and providing feedback on the final manuscript. S.O. made his contributions while at DeepMind, and is now in the People + AI Research (PAIR) team at Google.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-022-12547-0.

Correspondence and requests for materials should be addressed to S.O. or K.T.