Supplementary

1 Experimental Setup **1**
- 1.1 Dataset . . . 1
- 1.2 Baselines . . . 2
- 1.3 Additional Model Details and Hyperparameters . . . 2
- 1.4 Computational complexity . . . 5

2 Analytical Proofs **5**
- 2.1 Definitions . . . 5
- 2.2 Theorems for Analysis of Fisher Information in Multiset Embeddings for Temporal Graph . . . 6

3 Additional Empirical Results **10**
- 3.1 Analyzing the Impact of Hidden Agent Interaction Strength on Model Prediction . 10
- 3.2 Deciphering Temporal Context Feature Attention Maps: The Interplay between Hidden Agents, Information Density, and Prediction Accuracy . . . 11
- 3.3 Performance of STEMFold in Varied Topological Conditions with Fixed Number of visible Agents . . . 13
- 3.4 Evaluation of STEMFold with Sensor Failures for Visible Agents . . . 13
- 3.5 Robustness of STEMFold against Noisy Data . . . 14
- 3.6 Exploring Systems with Heterogeneous Agent Characteristics . . . 17
- 3.7 Model ablation: Impact of ODE Latent Dimension on Model Predictive Accuracy . 17

4 Broader Impact **18**

1. Experimental Setup

1.1. Dataset

Simulated Datasets: In our particle simulation experiments, we consider N particles, with N taking values from the set $\{5, 40\}$, placed within a 2D box. In the springs model, we randomly establish connections between pairs of particles with a 50% probability and these particles interact via Hooke's law, where the force F_{ij} acting on particle v_i due to particle v_j follows Hooke's law: $F_{ij} = -k(r_i - r_j)$, with k as the spring constant and r_i representing the 2D position vector of particle v_i. We sample initial positions from a Gaussian distribution ($N(0, 0.5)$), and initial velocities are assigned as random vectors with a norm of 0.5. Trajectories are simulated by numerically solving Newton's equations of motion using a leapfrog integration method similar to Kipf et al. (2018) with a fixed step size of 0.001, and we subsample the trajectories by selecting every 100th step for training and testing.

In contrast, for the charged particle model, we equip each particle with positive or negative charges, q_i, sampled uniformly from $\pm q$. The interaction between these charged particles is governed by Coulomb forces, defined as $F_{ij} = C \cdot \text{sign}(q_i \cdot q_j) \cdot \frac{(\mathbf{r}_i - \mathbf{r}_j)}{|\mathbf{r}_i - \mathbf{r}_j|^3}$, where C is a constant. Unlike the springs model, all pairs of charged particles interact, potentially resulting in attraction or repulsion, depending on their relative distances. For each of the simulated datasets, 10,000 training samples and 2,000 testing samples are generated. To incorporate hidden agents within the simulation, we randomly select M agents from the system to hide after the completion of all simulations while only preserving the edges with visible agents.

CMU Motion Capture Dataset: The Carnegie Mellon University (CMU) Motion Capture dataset (cmu), a comprehensive and widely recognized collection of motion capture data, was utilized in this study. This dataset embodies a diverse array of human movements, encompassing activities from walking and running to more intricate motions such as dancing, recorded from various subjects. Our empirical focus was on Subject 35 and their walking trajectories. The dataset extracted for our study consists of 8,063 frames, each documenting 31 specific points. All attributes, including position and velocity, were normalized to have a maximum absolute value of 1. We trained our models on 30-timestep sequences and subsequently assessed their performance on sequences of equivalent length.

Basket Ball Dataset: In the basketball dataset, each trajectory provides detailed information about the 2D positions and velocities of the offensive team, consisting of 5 players. Initially, these trajectories are divided into 49 frames, which collectively capture approximately 8 seconds of gameplay. During the training phase, all models undergo training using the initial 30 frames extracted from the training trajectories. When it comes to evaluation, the models are presented with input data comprised of sampled trajectories from the first 30 frames, and this sampling strategy is adjusted based on temporal sparsity. Specifically, for a temporal sparsity of 10%, we select 27 observations from the initial 30 observations for each individual player, and subsequently, the models are tasked with predicting the subsequent 19 frames.

1.2. Baselines

Recurrent Neural Networks We implement two recurrent baselines: Single RNN and Joint RNN. The first RNN baseline utilizes separate LSTMs (with shared weights) for each object. The second baseline, labeled as ”joint,” combines all state vectors by concatenation and feeds them into a single LSTM, which is trained to predict all future states simultaneously.

Fully Convolutional Graph Messaging(Watters et al. (2017)) We implement a message-passing network decoder similar to Kipf et al. (2018) operating over a fully connected graph of visible agents with only one edge type.

DNRI(Graber and Schwing (2020)) DNRI combines the power of graph neural networks and variational inference to model the interactions and dependencies between entities over time. It introduces a latent variable model that captures the temporal evolution of the system by incorporating a recurrent neural network (RNN) component. It allows for inferring the latent variables that represent the hidden states and interactions between the entities at different time steps. By using variational inference, DNRI provides a probabilistic framework that can capture uncertainty and make predictions about future interactions.

Table 1 presents the hyperparameters used for the evaluation of the baselines across all three datasets.

1.3. Additional Model Details and Hyperparameters

All components of the STEMFold are illustrated in Figure 1. The hyperparameters utilized to assess STEMFold on all the datasets are listed in Table 2.

Table 1: List and description of hyperparameters for baselines

Hyperparameter	Value	Description
Encoder latent	128	Latent size of encoder.
Decoder latent	128	Latent size of Decoder decoder.
Batch_size	128	The number of samples processed in a single pass.
lr	5×10^{-4}	The learning rate for training the model.
Optimizer	Adam	Model optimization algorithm.
Teacher forcing steps	30	Number of steps for which teacher forcing is applied.
Val teacher forcing steps	30	Whether to apply teacher forcing during validation.
Edge types	2	Number of types of edges in the graph.
Encoder layers	2	Number of layers in the encoder's MLP.

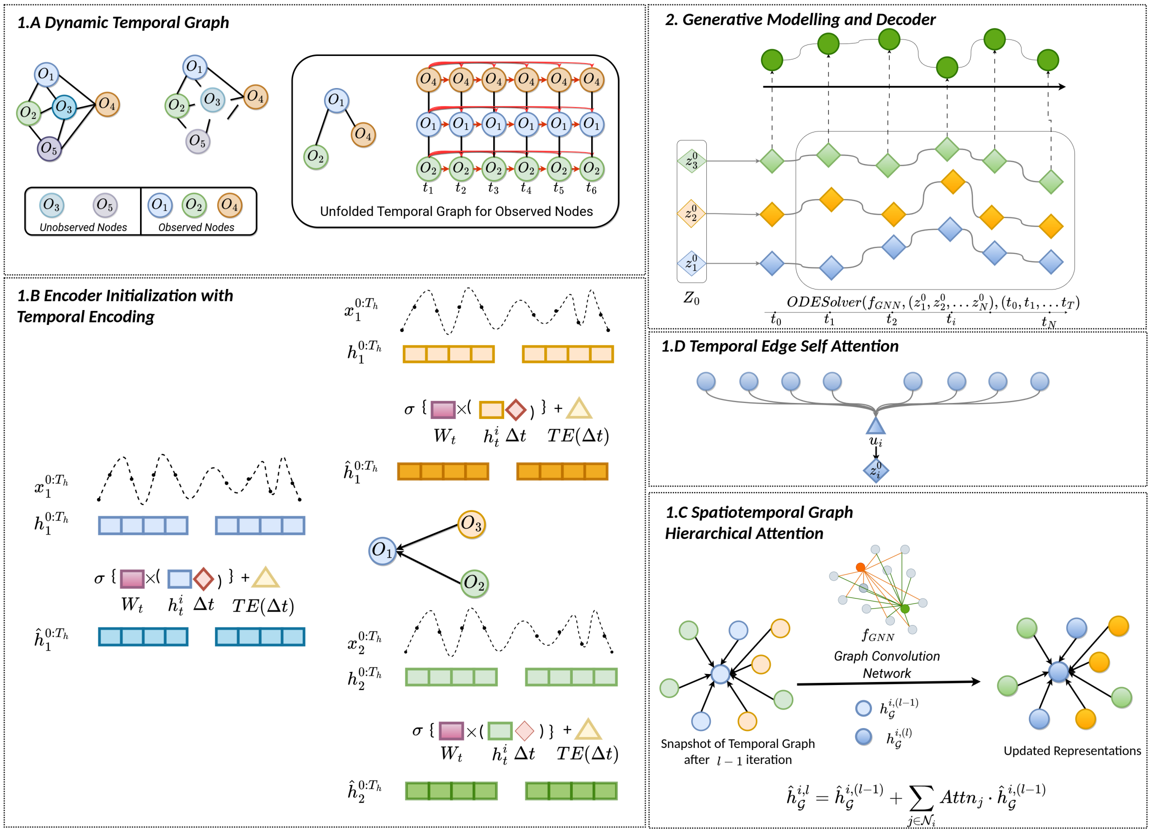

1.A Dynamic Temporal Graph
O_1
O_2
O_3
O_4
O_5
Unobserved Nodes
Observed Nodes
t_1 t_2 t_3 t_4 t_5 t_6
Unfolded Temporal Graph for Observed Nodes
2. Generative Modelling and Decoder
z_3^0
z_2^0
z_1^0
Z_0
$ODESolver(f_{GNN}, (z_1^0, z_2^0, \dots z_N^0), (t_0, t_1, \dots t_T)$
t_0 t_1 t_2 t_i t_N
1.B Encoder Initialization with Temporal Encoding
$x_1^{0:T_h}$
$h_1^{0:T_h}$
$\sigma \{ \times (\;) \} +$
W_t h_t^i Δt $TE(\Delta t)$
$\hat{h}_1^{0:T_h}$
$x_2^{0:T_h}$
$h_2^{0:T_h}$
$\hat{h}_2^{0:T_h}$
1.D Temporal Edge Self Attention
u_i
z_i^0
1.C Spatiotemporal Graph Hierarchical Attention
f_{GNN}
Graph Convolution Network
$h_{\mathcal{G}}^{i,(l-1)}$
$h_{\mathcal{G}}^{i,(l)}$
Snapshot of Temporal Graph after $l-1$ iteration
Updated Representations
$\hat{h}_{\mathcal{G}}^{i,l} = \hat{h}_{\mathcal{G}}^{i,(l-1)} + \sum_{j \in \mathcal{N}_i} Attn_j \cdot \hat{h}_{\mathcal{G}}^{i,(l-1)}$

Figure 1: Design framework for encoder and decoder in STEMFold (Best viewed in color.)

Neural ODE for Generative Modelling In systems involving continuous multi-variable dynamics, the state's dynamic nature is depicted through continuous values of t over a collection of dependent variables, and it progresses according to a sequence of first-order ordinary differential equations (ODEs):

$$\dot{z}_t^i := \frac{dz_t^i}{dt} = g_i(z_t^1, z_t^2, \dots, z_t^N)$$

These equations advance the states of the system in tiny steps over time. With the latent initial states $z_0^0, z_0^1, \dots, z_0^N \in \mathbb{R}^d$ for every object, z_t^i is the resolution to an ODE initial-value problem (IVP) and can be computed at any required times using numerical ODE solvers like Runge-Kutta:

$$z_T^i = z_0^i + \int_0^T g_i(z_t^1, z_t^2, \dots, z_t^N)dt$$

The function g_i outlines the dynamics of the latent state, and it has been proposed to be parameterized with a neural network in recent research, allowing for data-driven learning.

By generalizing to continuous scenarios, where N_i denotes the set of immediate neighbors of object o_i, we reformulate it as:

$$\dot{z}_t^i := \frac{dz_t^i}{dt} = g_i(z_t^1, z_t^2, \dots, z_t^N) = f_O\left(\sum_{j \in N_i} f_R([z_t^i, z_t^j])\right)$$

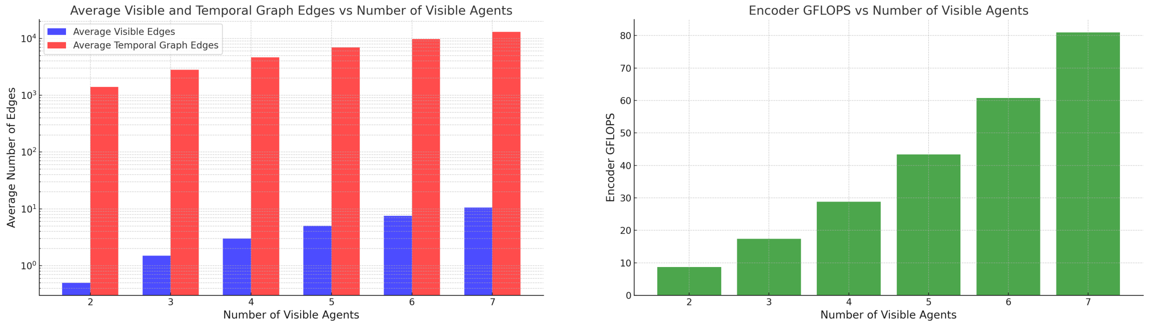
Here, the $||$ is the concatenation operations, and f_O, f_R are two neural networks to capture the interaction of the latent system. The ODE function and the latent initial state z_0^i will define the complete trajectories for each object. For the ode solver, we use the fourth-order Runge-Kutta method based on Chen et al. (2018) using the torchdiffeq python package (Chen (2018)).

Hyperparameter	**Value**	**Description**
Scheduler	Cosine	Schedulerused to adjust the learning rate during training.
Test Data Size	2000	The number of samples in the test dataset.
Observation Std. Dev.	0.01	The standard deviation of the observation noise.
Number of Epochs	100	The number of times the learning algorithm will work through the entire training dataset.
Learning Rate	5×10^{-4}	The step size at each iteration while moving toward a minimum of the loss function.
Batch Size (Simulated)	128	The number of training examples utilized in one iteration.
Random Seed	1991	The seed used by the random number generator.
Dropout Rate	0.2	The probability of setting a neuron to zero during training.
Latent Size	16	The dimensionality of the latent space.
GNN Dimension	64	The dimensionality of the Graph Neural Network.
ODE Func Dimension	128	The dimensionality of the ODE Function.
GNN Layers	2	The number of layers in the Graph Neural Network.
Number of Heads in z_0 Encoder	1	The number of attention heads in the initial encoder.
ODE Func Layers	1	The number of layers in the ODE Function.
ODE Solver	RK4	The method used to solve the Ordinary Differential Equation, Runge-Kutta of order 4 in this case.
Gradient Norm Clipping	10	The maximum allowed value for the gradient norm, used to prevent exploding gradients.
Number of Edge Types	2	The number of different types of edges in the graph.
L2 Regularization	1×10^{-3}	The weight decay parameter to prevent overfitting.
Optimizer	AdamW	The optimization algorithm used to minimize the loss function.

Table 2: List and description of hyperparameters used in STEMFold

1.4. Computational complexity

In Figure 2, we present the computational complexity of our encoder's temporal graph. For evaluation, a spring system comprising 10 agents was simulated, generating simulations with varying distributions of visible and hidden agents. We observe the number of edges in the visible graph and temporal graph. For example, a model trained on data with 7 visible and 3 hidden agents yields an average of 10.5 edges for the visible agents. In contrast, our encoder's temporal graph, constructed over 30 timesteps, encompasses 13,048 edges. As the count of visible agents escalates, there's a corresponding increase in the temporal graph's edges, scaling at $\mathcal{O}((E + N)T^2)$, where E and N denote the edges and nodes of the initial interaction graph, excluding hidden agents. This relationship is illustrated in Figure 2a, which plots the average temporal edges against the average visible edges in the interaction graph. Additionally, Figure 2b showcases the GFLOPs of the STEMFold's encoder in relation to the increment in visible agents.



(a)

(b)

Figure 2: a.) A comparative representation of the average visible edges and average temporal graph edges against the number of visible agents. b.) Representation of Encoder GFLOPS against the Number of Visible Agents. Each bar signifies the computational complexity in GFLOPS of the encoder for the corresponding number of visible agents, highlighting the proportional increase in computational demand with the increase in visible agents

2. Analytical Proofs

2.1. Definitions

Multisets and kernels for multisets A *multiset* is a generalized notion of a set of a set, which accommodates multiple instances of its elements. We deliberate on multisets of features in $\mathbb{R}^d$, represented as:

$$\mathcal{X}^d = \left\{ \mathbf{x} \mid \mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \text{with each } \mathbf{x}_i \in \mathbb{R}^d \text{ for some } n \geq 1 \right\}$$

The cardinality of a multiset symbolized as $|\cdot|$, is determined by summing the multiplicities of its elements.

In this context, we assume the existence of a kernel on the space of multisets, represented as $K_{\mathrm{ms}} : \mathcal{X}^d \times \mathcal{X}^d \to \mathbb{R}$and its either an exact or an approximate embedding, $\psi_{\mathrm{ms}} : \mathcal{X}^d \to \mathbb{R}^p$, such that

$$K_{\mathrm{ms}}(\mathbf{x}, \mathbf{x}') \approx \langle \psi_{\mathrm{ms}}(\mathbf{x}), \psi_{\mathrm{ms}}(\mathbf{x}') \rangle$$

Temporal Graph Let $G(V(t), E(t))$ be the graph with nodes $V(t)$ and edges $E(t)$ at time t. Let G' be a subgraph of G with observed nodes $x_1(t), x_2(t), \ldots, x_N(t)$. The *Temporal Graph* T' can be defined as a multiset of the states of graph G' at different time points, represented as: $T' = \{G'(t_1), G'(t_2), \ldots, G'(t_r)\}$ where each $G'(t_i)$ is a member of the multiset representing the state of graph G' at time t_i, and additional temporal edges are added between nodes in $G'(t_i)$ and $G'(t_{i+1})$ for all $i = 1, 2, \ldots, r-1$ to represent the temporal connections between the different states of a graph G'.

In the derivation of all our analytical results, we base our arguments on the subsequent assumptions:

Assumptions:

1. We assume the embedding of each individual node, $x_i(t)$, to conform to a multivariate Gaussian distribution, parametrized by $\theta = \{\mu, \Sigma\}$.

2. The embedding of the multiset, X_i, is hypothesized to adhere to a Gaussian Mixture Model (GMM) with K components, described by parameters $\phi = \{\pi, \mu, \Sigma\}$. Here, π signifies the mixture weights, μ represents the means, and Σ defines the covariance matrices of the components.

3. $I(\theta; N)$ represent the Fisher Information Matrix (FIM) as a function of the parameter θ and the number of observed nodes N.

4. The Fisher Information is a differentiable function with respect to the number of observed nodes.

2.2. Theorems for Analysis of Fisher Information in Multiset Embeddings for Temporal Graph

Theorem 1 *Let X_i denote the embedding of the multiset representing the collection of graph states $\{x_i(t_1), x_i(t_2), \ldots, x_i(t_r)\}$ in the temporal graph T'. Let $I(\theta|X_i)$ and $I(\theta|x_i(t))$ denote the Fisher information of θ given the embedding of X_i and the embedding of each individual element $x_i(t)$, respectively. Then,*

$$I(\phi|X_i) \geq \sum_{t=t_1}^{t_r} I(\theta|x_i(t)). \tag{1}$$

Proof: Let the probability density function representing the embedding of node x_i at time t be $f(x_i(t); \theta)$, parameterized by θ. Similarly, let the probability density function representing the embedding of the multiset X_i be $g(X_i; \phi)$, parameterized by ϕ. Each individual node embedding $x_i(t)$ is assumed to follow a Gaussian distribution:

$$f(x_i(t); \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i(t) - \mu)^2}{2\sigma^2}\right) \tag{2}$$

The multiset embedding X_i is assumed to follow a Gaussian Mixture Model with K components:

$$g(X_i;\pi,\mu,\Sigma)=\sum_{k=1}^{K}\pi_k\mathcal{N}(X_i;\mu_k,\Sigma_k) \tag{3}$$

If the Fisher information for an individual node is given by $T(\theta)$ and the Fisher information of the multiset X_i is given as $I(\phi|X_i)$, then:

$$I(\theta)=\mathbb{E}\left[\left(\frac{d}{d\theta}\log f(x_i(t);\theta)\right)^2\right]\Rightarrow I(\phi|X_i)=\mathbb{E}\left[\left(\frac{d}{d\phi}\log g(X_i;\phi)\right)^2\right] \tag{4}$$

Let's assume that the covariate distribution between any two nodes $x_i(t)$ and $x_j(t)$ is Gaussian, with parameters $\theta=\{\mu_{ij},\sigma_{ij}^2\}$, where μ_{ij} is mean and σ_{ij}^2 is the variance of the Gaussian distribution representing the covariate between nodes i and j. Given the Gaussian covariate distribution between the nodes, the Fisher Information for the covariate distribution between nodes i and j is given by:

$$I(\theta)=\begin{bmatrix}\frac{1}{\sigma_{ij}^2} & Cov(\mu,\sigma^2)\\ Cov(\mu,\sigma^2) & \frac{1}{2\sigma_{ij}^4}\end{bmatrix}$$

For a Gaussian Mixture Model, the Fisher information matrix $I(\phi|X_i)$ where $\phi=\{\pi,\mu,\Sigma\}$ depends on the derivatives of the log-likelihood with respect to the parameters. The elements of the Fisher information matrix are given by the expected second derivatives of the log-likelihood, which can be computed using the Expectation-Maximization (EM) algorithm. Assume that the embedding of node $x_i(t)$ follows a Gaussian distribution with mean μ and variance σ^2, both parameterized by θ. The Fisher information, $I(\theta)$, for this node is derived as follows:

$$I(\theta)=\mathbb{E}\left[\left(\frac{d}{d\theta}\log f(x_i(t);\mu,\sigma^2)\right)^2\right] \tag{5}$$

Assume that the embedding of the multiset X_i follows a Gaussian mixture model with K components, each with its own mean μ_k and variance σ_k^2, all parameterized by ϕ. The Fisher information, $I(\phi|X_i)$, for this multiset is derived as follows:

$$I(\phi|X_i)=\mathbb{E}\left[\left(\frac{d}{d\phi}\log g(X_i;\{\mu_k,\sigma_k^2\}_{k=1}^{K})\right)^2\right] \tag{6}$$

To compare $I(\phi|X_i)$ and $I(\theta)$, we need to compare the respective Fisher information matrices.

Since these matrices are of different dimensions, a direct comparison is not straightforward. However, we can compare the determinant of the Fisher information matrices as a scalar representation of the information contained in the embeddings. We aim to compare the determinant of the Fisher Information Matrix for a Gaussian Mixture Model (GMM) with that of a Gaussian distribution. We will symbolically represent the Fisher Information Matrix for a GMM and derive its determinant to compare with the determinant of the Fisher Information Matrix for a Gaussian distribution. Let's consider a GMM with K components, each with parameters $\phi_k=\{\pi_k,\mu_k,\Sigma_k\}$, where π_k is the weight, μ_k is the mean, and Σ_k is the covariance matrix of the k-th component. The log-likelihood for the GMM is given by:

$$\log L(\phi) = \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i; \mu_k, \Sigma_k)\right) \tag{7}$$

The Fisher Information Matrix, $I(\phi|X_i)$, for the GMM is a block-diagonal matrix, where each block corresponds to the Fisher Information Matrix for the parameters of component k, $J(\phi_k)$. Each block, $J(\phi_k)$, can be represented symbolically as:

$$[J(\phi_k)]_{mn} = \mathbb{E}\left[\frac{\partial^2 \log L(\phi)}{\partial \phi_{km} \partial \phi_{kn}}\right] \tag{8}$$

Now, considering Gaussian covariance between the components, we need to consider the interaction between the components of the Gaussian Mixture Model (GMM) and derive the Fisher Information Matrix accordingly. When the components are not independent, the blocks of the Fisher Information Matrix are not necessarily diagonal, and the off-diagonal elements represent the covariance between the components. Let's denote the covariance between component k and component l as Σ_{kl}. The Fisher Information Matrix, $I(\phi|X_i)$, for the GMM with covariance can be represented as:

$$[I(\phi|X_i)]_{mn} = \mathbb{E}\left[\frac{\partial^2 \log L(\phi)}{\partial \phi_{km} \partial \phi_{ln}}\right] + \Sigma_{kl}$$

The Fisher Information for the GMM can be expressed as a weighted sum of the Fisher Information of the individual components:

$$J_{X_i}(\phi) = \sum_{k=1}^{K} \pi_k I_{x_i}(\theta_k; N_k)$$

where π_k are the mixture weights, θ_k are the parameters for each component, and N_k is the number of observations assigned to the k-th component.

Let $I(\phi|X_i)$ denote the Fisher Information Matrix with covariance, represented as a block matrix:

$$I(\phi|X_i) = \begin{bmatrix} J(\phi_k) & \Sigma_{kl} \\ \Sigma_{lk} & J(\phi_l) \end{bmatrix}$$

where $J(\phi_k)$ and $J(\phi_l)$ are the Fisher Information Matrices for individual components, and Σ_{kl} and Σ_{lk} are the covariance matrices between the components.

There can be two cases that arise here.

Case I: $\Sigma_{kl} = \Sigma_{lk} = 0$ Then, the determinant of $I(\phi|X_i)$ is strictly greater than the product of the determinants of the individual Fisher Information Matrices, i.e.,

$$\det(I(\phi|X_i)) > \det(J(\phi_k)) \cdot \det(J(\phi_l))$$

i.e the determinant of the Fisher Information Matrix with covariance for two components is greater than the determinant of the Fisher Information Matrix when they are independent. Given that the determinant of the Fisher Information Matrix for the GMM with covariance is greater than the determinant of the Fisher Information Matrix, it is evident that the multiset embedding X_i will contain more information about the parameters than the individual node embedding $x_i(t)$ when considering Gaussian covariance between the components. Thus, since the determinant of $I(\phi|X_i)$ is greater than the determinant of $I(\theta)$, then it can be concluded that the multiset embedding contains more information about the parameters than the individual node embedding.

Case II: $\Sigma_{kl}, \Sigma_{lk} \neq 0$

When there is covariance between two Gaussian components in a GMM, the elements in $I(\phi|X_i)$ representing the covariance between these components would be non-zero, symbolizing the interaction between the components. To prove the inequality $|I(\phi|X_i)| > |I(\theta)|$, let us elaborate that the determinant of the Fisher Information Matrix, $|I(\phi|X_i)|$, for the GMM with covariance, will typically be greater due to the additional terms representing the interaction between the components along with the individual components' information. Let us assume there are K Gaussian components in the GMM, each with its mean and variance, and let's denote the covariance between the i-th and j-th components as $cov(i,j)$. The determinant of $I(\phi|X_i)$ would be the sum of the determinants of the individual components plus the terms representing the covariance interaction between the components:

$$|I(\phi|X_i)| \approx \sum_{i=1}^{K} |I_i| + \sum_{i \neq j} cov(i,j)$$

Since the covariance terms represent additional information not present in a single Gaussian component, it would generally contribute to a greater determinant of $I(\phi|X_i)$ as compared to $|I(\theta)|$:

$$|I(\phi|X_i)| > |I(\theta)|$$

Hence, for both cases, we proved that the Fisher information of the embedding of the multiset X_i is greater than the Fisher information of the embedding of each individual element $x_i(t)$.

Theorem 2 *Given the reduced temporal graph T' , the corresponding reduced spatial graph G', and the static spatial graph G, if the Fisher information of the embedding of T' exceeds the Fisher information of the embedding of G', i.e.,*

$$I(T') > I(G')$$

then it follows that the covariance of the estimator of the reduced temporal graph, $Cov(\hat{\theta}_{T'})$, is less than the covariance of the estimator of the reduced spatial graph, $Cov(\hat{\theta}_{G'})$, represented as:

$$Cov(\hat{\theta}_{T'}) < Cov(\hat{\theta}_{G'})$$

Proof: Let $I(T')$ and $I(G')$ denote the Fisher Information in the reduced temporal graph T' and the reduced spatial graph G' respectively, both of which are derived from a complete graph G. The Fisher Information Matrix for each graph is computed based on the observed nodes and their relationships within the respective graphs.

From definition, the temporal graph T' is the multiset representation of a sequence of spatial graphs G' at different time points. According to Cramér–Rao Lower Bound (CRLB), i $X = (X_1, X_2, \ldots, X_n)$ be a random vector with probability density function $f(\mathbf{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta} = (\theta_1, .., \theta_k)$

is a vector of parameters of interest. Let $\mathbf{T}(\mathbf{X}) = (T_1(\mathbf{X}), T_2(\mathbf{X}), \ldots, T_k(\mathbf{X}))$ be an unbiased estimator of $\boldsymbol{\theta}$, i.e., $\mathbb{E}[\mathbf{T}(\mathbf{X})] = \boldsymbol{\theta}$. Then, for any unbiased estimator $\mathbf{T}(\mathbf{X})$, the covariance matrix of $\mathbf{T}(\mathbf{X})$ satisfies:

$$\mathrm{Cov}(\mathbf{T}(\mathbf{X}), \boldsymbol{\theta}) \geq \mathbf{I}(\boldsymbol{\theta})^{-1},$$

where $\mathbf{I}(\boldsymbol{\theta})$ is the Fisher Information matrix of the random vector $\mathbf{X}$ with respect to the parameter vector $\boldsymbol{\theta}$ From Theorem 1, we can conclude that the Fisher information of the embedding of the T' is greater than the Fisher information on the embedding of each spatial graph G' at any timestep.

$$I(T') > I(G')$$

The Cramér-Rao Lower Bound provides a fundamental limit on the covariance of an unbiased estimator. Formally, for an unbiased estimator $\hat{\theta}$ of a parameter θ, the variance of $\hat{\theta}$ is bounded by:

$$\mathrm{Cov}(\hat{\theta}) \geq \frac{1}{I(\theta)},$$

where $I(\theta)$ is the Fisher Information of θ. Consider two estimators $\hat{\theta}_{T'}$ and $\hat{\theta}_{G'}$ derived from the temporal graph T' and another graph G', respectively. Thus, the Fisher Information of T' is greater than that of G':

$$I(T') > I(G')$$

Applying the CRLB to each estimator, we obtain:

$$\mathrm{Cov}(\hat{\theta}_{T'}) \leq \frac{1}{I(T')}, \quad \mathrm{Cov}(\hat{\theta}_{G'}) \leq \frac{1}{I(G')}$$

Given that $I(T') > I(G')$, it directly follows that:

$$\mathrm{Cov}(\hat{\theta}_{T'}) < \mathrm{Cov}(\hat{\theta}_{G'})$$

Thus, it is concluded that the covariance of the reduced temporal graph T' serves as a more accurate estimator for the complete graph G compared to the covariance of the reduced spatial graph G'. Based on the above two theorems, we can deduce that if $\mathrm{Cov}(T')$ and $\mathrm{Cov}(G')$ are the estimators of parameters θ of the full spatial graph $\mathrm{Cov}(G)$ then: $\mathrm{Cov}(T') < \mathrm{Cov}(G')$ i.e. the covariate of the temporal graph $\mathrm{Cov}(T')$ is a better estimator of the complete graph $\mathrm{Cov}(G)$ than Cov(G'). Hence, constructing a temporal graph from the spatial graph of visible nodes in a multi-agent system where some nodes are unobservable all the time yields a superior representation of the entire system compared to the reduced spatial graph, subsequently enhancing the performance of visible agent trajectory prediction.

3. Additional Empirical Results

3.1. Analyzing the Impact of Hidden Agent Interaction Strength on Model Prediction

In this study, we study the influence of hidden agents by modifying the interaction strength amongst hidden agents in a spring system, with the interaction (coupling) strength systematically adjusted between 0.5 to 5.0. Concurrently, the interaction strength for visible agents is statically maintained at 1. For the spring dataset, interaction strength, symbolized as $F_{i,j}$, is quantified by the equation $F_{i,j} = -k(x_i - x_j)$, where k represents the interaction strength between the entities i and j.

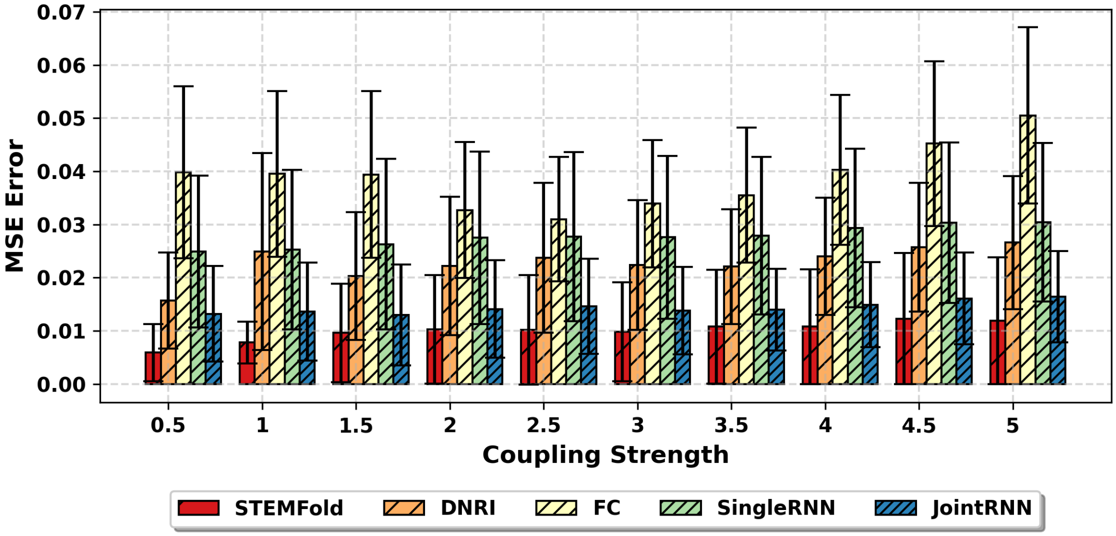


Figure 3: Mean Squared Error (MSE) values for the models as the interaction (coupling) strength is increased for the hidden agents.

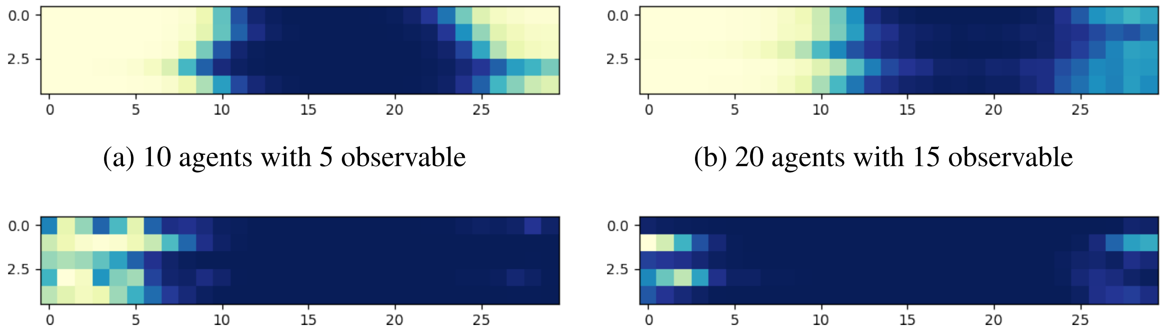
The models were trained on a spring dataset with 50% observability consisting of 10,000 samples for each specified level of coupling and were subsequently evaluated on a separate test dataset, comprising 2,000 samples.

Figure 3 shows the 30^{th}-step prediction error for all the baselines. A prominent observation from our experimental results is the exceptional and consistent performance of the STEMFold model across all degrees of coupling coefficients. STEMFold not only exhibited a lower mean prediction error and low variance compared to the baseline models but also demonstrated remarkable stability, with its error rate not exhibiting a swift increase with the enhancement in interaction strength for hidden agents. This contrasts markedly with the other models, which showed a discernible upward error trend with increasing interaction strength. This empirical evidence underscores the resilience and dependability of STEMFold in scenarios with varied interaction strengths, especially where the influence of hidden agents is pronounced in the system.

3.2. Deciphering Temporal Context Feature Attention Maps: The Interplay between Hidden Agents, Information Density, and Prediction Accuracy

Figure 4 visually illustrates temporal context feature attention maps for spring systems, each with a distinct proportion of hidden agents, ranging from 50% to 87.5%, while maintaining a constant count of five visible agents. The y-axis represents the index of the agent, and the x-axis plots the timesteps, with each row in the attention map representing the temporal attention values at different timesteps.

The attention values are scaled between 0 and 1, with yellow cells indicating a value of 0, and progressively darker shades of blue signifying attention values nearing 1. A critical observation is that as the proportion of hidden agents increases, the attention maps become densely populated with values of 1. This suggests that the network is utilizing every available timestep in the sequence to



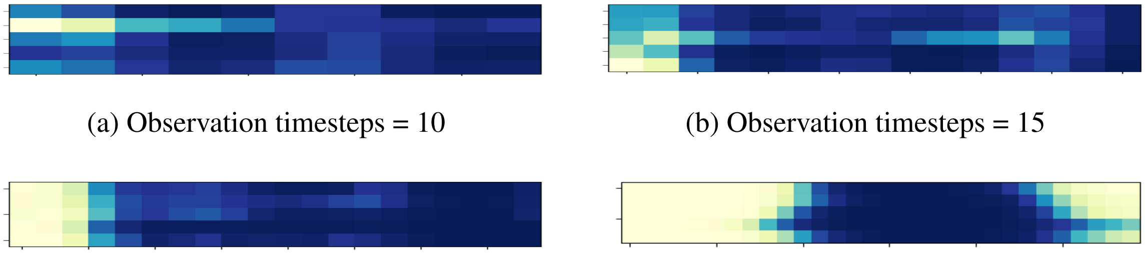
(a) 10 agents with 5 observable

(b) 20 agents with 15 observable

(c) 30 agents with 25 observable

(d) 40 agents with 35 observable

Figure 4: Temporal Context Feature Attention Maps: Visualization of temporal context feature attention across various configurations each with 50% observability



(a) Observation timesteps = 10

(b) Observation timesteps = 15

(c) Observation timesteps = 20

(d) Observation timesteps = 30

Figure 5: Temporal Context Feature Attention Maps: Feature attention maps applied to a system with 10 agents, including 50% unobservable agents, with variations in observation time.

refine the precision of its future predictions. This transition to denser attention maps underscores a pivotal implication: the network, when faced with denser maps, is signaling a potential insufficiency in the available information. It is indicative of the network's increasing demand for more comprehensive data to optimize its predictive accuracy. Therefore, this density in attention values implies a heightened necessity to augment the number of timesteps observed. By extending the observation timesteps, we can cater to the network's increasing information needs, thereby enhancing the model's predictive accuracy and precision.

In essence, the densification of attention values in the maps is a clear indicator of the network's struggle with the available information, emphasizing the potential requirement to increase the observation timesteps to fulfill the network's information needs and, consequently, improve the accuracy of predictions.

Figure 5 provides further insight into this phenomenon by showcasing attention maps of four distinct models of a system, each consisting of 10 agents, 5 of which are hidden, across varied observation time periods, extending from 10 to 30 timesteps. A prominent observation from these maps is the progressive sparsification of the attention maps and a concurrent increase in predictive accuracy as the number of timesteps is increased. This is depicted in figure 6 where we plot the average MSE error for systems as their encoder's observation time is increased. This sparsification and enhanced accuracy suggest that the determination of an optimal observation period can be

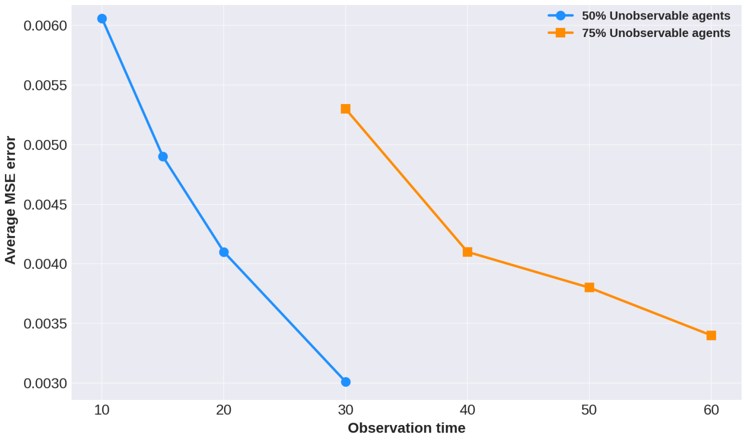


Figure 6: Prediction accuracy for two spring system with 50% and 75% unobservable agents as the observation time for encoder is increased

strategically made, contingent upon the number of hidden agents within the system. This analysis uncovers a crucial correlation: the higher the proportion of hidden agents in a system, the more extensive the observation period required to achieve accurate predictions. This denotes that systems with a greater number of hidden agents demand a more comprehensive observation framework to accurately capture the intricacies of the system dynamics and produce precise predictions.

In conclusion, the decrease in the density of attention maps and the corresponding enhancement in accuracy with extended timesteps emphasize the importance of selecting an optimal observation period, particularly in systems with a significant number of hidden agents. The insights derived from these attention maps serve as a valuable guide in the strategic selection of observation periods, facilitating the development of robust models capable of delivering precise predictions in a variety of scenarios.

3.3. Performance of STEMFold in Varied Topological Conditions with Fixed Number of visible Agents

In this experiment, we fix the quantity of visible agents within the system, while the number of hidden agents is subjected to variation. Figure 7 graphically represents the efficacy of the model, which has been trained on a spring system with 10 total agents out of which 5 are hidden agents. This is evaluated against systems with a diverse range of hidden agents, all the while maintaining the count of visible agents at 5.

The STEMFold models consistently demonstrate superior performance over the baselines, regardless of the variations in the ratio of hidden to visible agents. This superiority of STEMFold models is indicative of their robustness and adaptability across different scenarios, showcasing their ability to yield reliable results with different proportions of hidden and visible agents.

3.4. Evaluation of STEMFold with Sensor Failures for Visible Agents

In this study, we address scenarios where observations for visible agents are intermittently unavailable due to random sensor failures. We consider two types of sensor failures: a) Asynchronous

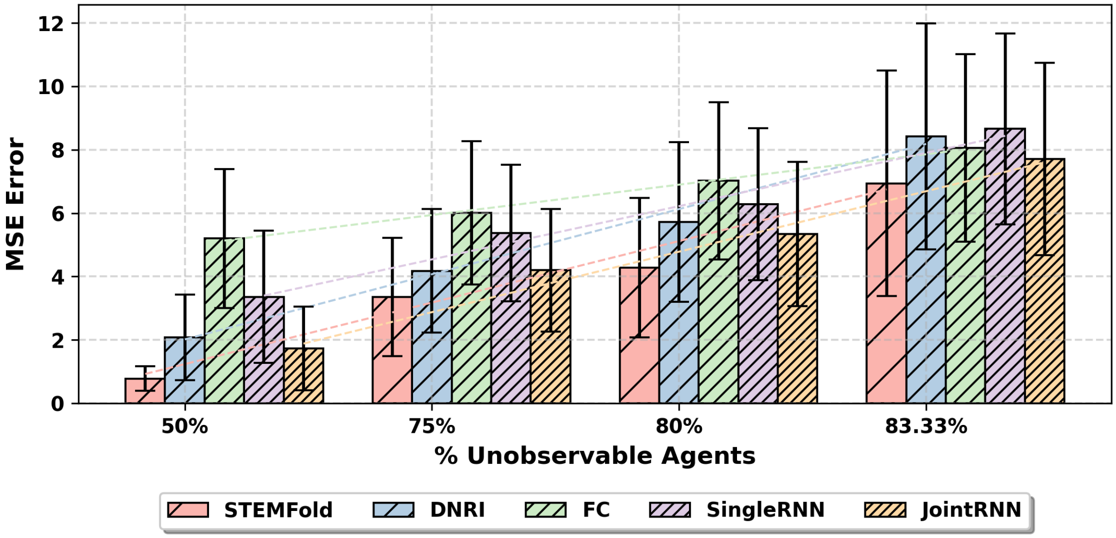


Figure 7: Mean Squared Error (MSE) values ($\times 10^{-2}$) for the model trained on a 10-5 configuration, while altering the total number of agents in the system, while keeping the visible agents fixed at 5.

Sensor Failure, and b) Synchronous Sensor Failure. In the case of Synchronous Sensor Failure, all sensors for the visible agents fail simultaneously, leading to observations being available only at certain timesteps. Specifically, we randomly select 20 out of 30 timesteps, and the model receives observations only for these selected steps. Figure 8 illustrates the MSE error for a spring system with 10 agents, varying the percentage of unobservable agents. In contrast, during Asynchronous Sensor Failure, each agent's sensor fails independently, and we have observations for only 20 timesteps per agent. Figure 9 displays the MSE error for asynchronous sensor failure across the model. Compared to other models, STEMFold demonstrates significantly lower error rates in both asynchronous and synchronous sensor failure scenarios.

3.5. Robustness of STEMFold against Noisy Data

This study further explores STEMFold's resilience to noisy observations by training the model on noise-free data and evaluating it under Gaussian noise conditions (mean = 0) with varying standard deviations (0.001 to 0.1). In our investigation, we normalized the data before introducing noise to simulate real-world scenarios. We observed the model's performance in a spring system with 10 agents, particularly focusing on scenarios with different percentages of unobservable agents. Figure 10 depicts the Mean Squared Error (MSE) under Gaussian noise with a standard deviation of 0.1, highlighting STEMFold's robustness even with high noise levels. Additionally, Figure 11 examines the MSE in a scenario where 50% of the agents are unobservable across different noise intensities, further illustrating the model's substantial resilience to noise. These results underscore STEMFold's superior performance against noise, especially in comparison to baseline models.

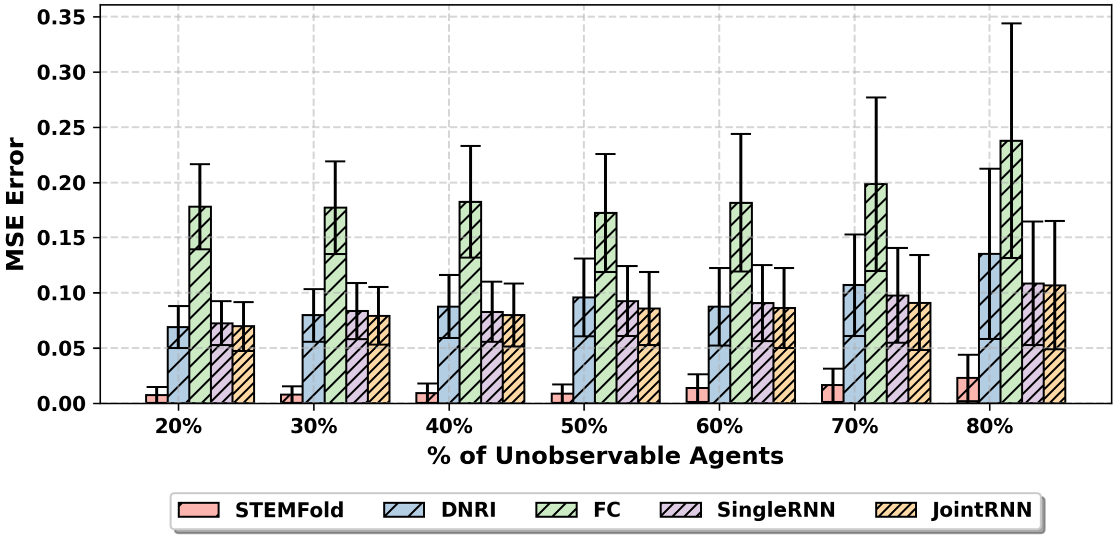


Figure 8: MSE error for synchronous sensor failure. Observations are randomly sampled for 20 steps out of 30 for all agents and provided to the model for evaluation.

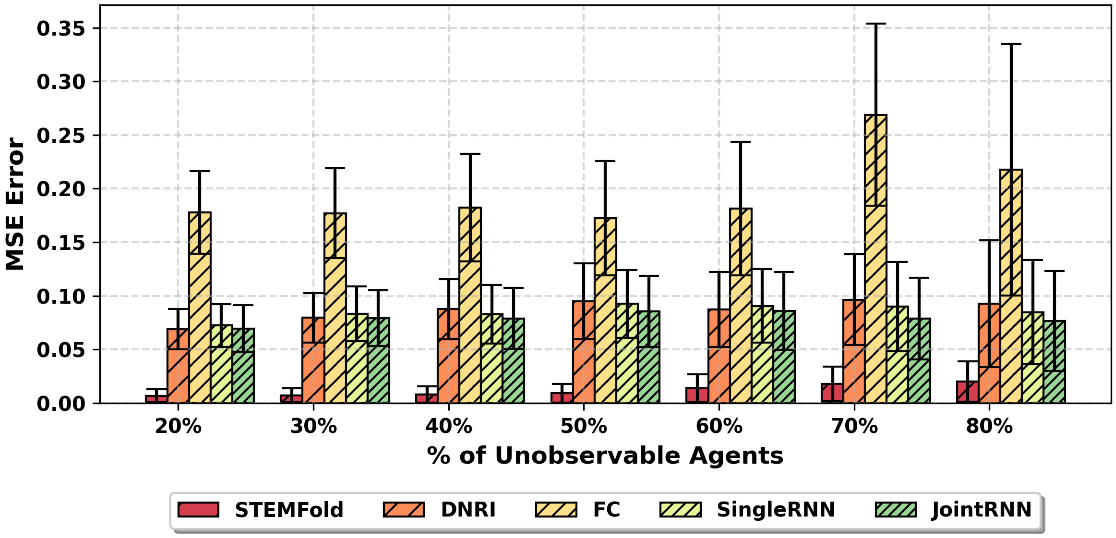


Figure 9: MSE error for asynchronous sensor failure. Observations are randomly sampled for 20 steps out of 30 for each agent and provided to the model for evaluation.

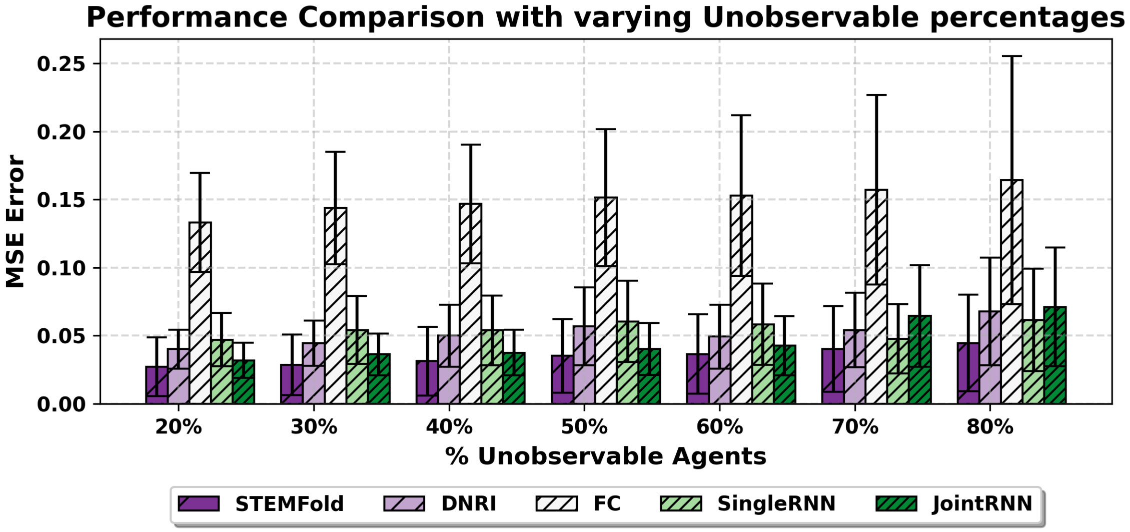


Figure 10: MSE Performance under Gaussian Noise (SD = 0.1) in a Spring System with 10 Agents, demonstrating STEMFold's effective noise handling capabilities.

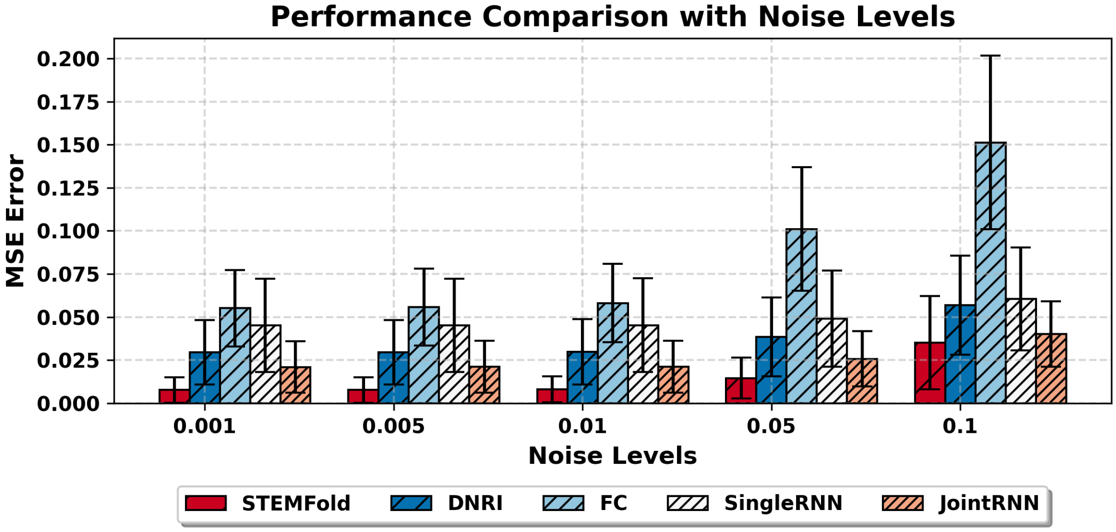
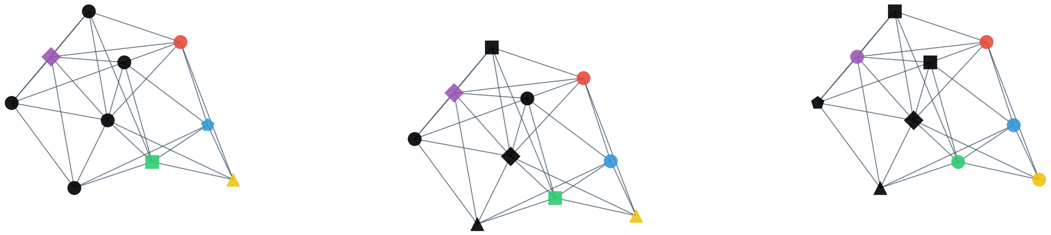


Figure 11: MSE Trends for a System with 50% Unobservable Agents across Various Noise Levels, showcasing the robustness of STEMFold in complex, partially observable environments.



(a) Only visible agents are heterogeneous and hidden agents are homogeneous

(b) All agents are heterogeneous and randomly sampled

(c) Visible agents are homogeneous and hidden agents are heterogeneous

Figure 12: Different configurations of heterogeneous agents in our study

3.6. Exploring Systems with Heterogeneous Agent Characteristics

Our previous analysis primarily addressed systems with homogeneous agents, characterized by uniform dynamics across all entities. This section ventures into the realm of heterogeneous agents, introducing variability in agent dynamics. Specifically, we explore a spring system setup where each agent, as a heterogeneous entity, possesses distinct and unknown coupling parameters. In contrast to our earlier homogeneous agent experiments, which operated under a single coupling parameter setting for all agents, this study delves into varied configurations. We examine three distinct scenarios:

1. *Visible Heterogeneity, Hidden Homogeneity:* Only the visible agents exhibit heterogeneity, while hidden agents maintain homogeneous characteristics.

2. *Universal Heterogeneity:* Every agent in the system, both visible and hidden, is heterogeneous, with their coupling parameters randomly assigned.

3. *Hidden Heterogeneity, Visible Homogeneity:* This scenario reverses the first, with only hidden agents being heterogeneous.

Coupling Parameter Configurations: For the heterogeneous agents, we define three coupling parameter sets: a.) 3 types of agents: $\{0, 0.5, 1\}$, b.) 4 types of agents $\{0, 0.5, 1, 1.5\}$, and c.) 5 types of agents $\{0, 0.5, 1, 1.5, 2\}$.

During simulations, each heterogeneous agent's coupling parameter is randomly selected from these sets with uniform probability. Table 3 presents the error metrics for baseline models across different heterogeneous agent configurations, particularly when all agents are considered heterogeneous. We observe that baseline models struggle to capture the intricate dynamics of this setup, resulting in significantly higher error rates compared to our proposed model. Additional configurations and their outcomes are depicted in Figure 12, where similar trends are noted.

3.7. Model ablation: Impact of ODE Latent Dimension on Model Predictive Accuracy

For this experiment, we chose the spring system system with 50% unobservability and systematically varied the latent dimension of the ODE function. Our findings indicate that the optimal

Table 3: Performance Metrics for Different Models for Heterogeneous Agents.

	STEMFold		**DNRI**		**FC**		**SingleRNN**		**JointRNN**	
Number of Het. Types	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
3	0.0104	0.0096	7.12	0.4076	2.28	0.39	2.92	0.26	3.55	0.31
4	0.0081	0.0077	7.16	0.38	2.26	0.377	2.91	0.2639	3.53	0.288
5	0.0089	0.0079	7.27	0.37	2.28	0.377	2.9	0.2454	3.5	0.27

performance is achieved when the ODE latent size is set to 64, and performance deteriorates as the latent size deviates from this value. This phenomenon can be attributed to the following factors: When the latent size is kept small (e.g., 16 or 32), the model exhibits underfitting, meaning it struggles to capture the crucial characteristics and relationships within the multi-agent observations. Conversely, when the latent size is significantly increased (e.g., 512), it gives rise to the curse of dimensionality. In high-dimensional spaces, generalization becomes challenging as the model requires an extensive amount of data to effectively cover the feature space, leading to potentially poorer performance on the task at hand.

Table 4: Average MSE Error for different ODE latent dimension

Size of ODE Latent	Average MSE Error
16	0.0053
32	0.0041
64	0.0030
128	0.0037
256	0.0034
512	0.0041

4. Broader Impact

Many often we do not operate in complete information settings for these complex co-evolving systems and addressing the practical challenges of measuring the entire system, our work provides valuable insights into the analysis of subgraphs in various domains. This has implications for fields such as protein-protein interactions, metabolic networks, planetary systems, and robotic systems, where complete agent measurements are often unattainable. Additionally, our framework is beneficial for large-scale networks that are either computationally intensive to handle, as it enables deliberate sampling of smaller subnetworks for analysis or have sensor failures thereby having incomplete knowledge of the system's degrees of freedom. This has practical implications for researchers and practitioners working with complex networks, allowing them to focus their analysis on representative subgraphs while maintaining reasonable accuracy.

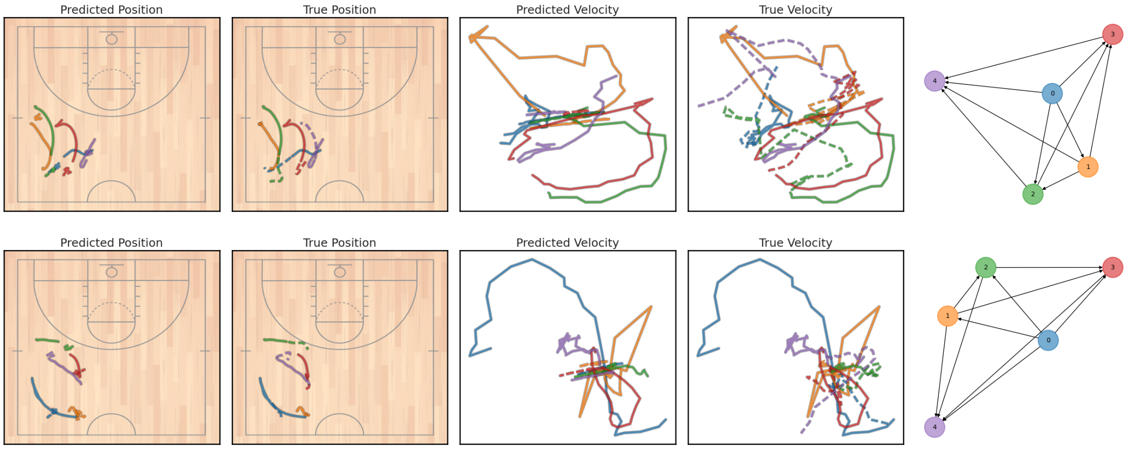


Figure 13: Visualizations depicting predictive trajectories for basketball dataset involving 5 players. Dotted lines represent predicted trajectories, while solid lines represent observed trajectories.

References

Cmu mocap dataset. http://mocap.cs.cmu.edu/. Accessed: 2023-06-05.

Ricky T. Q. Chen. torchdiffeq, 2018. URL https://github.com/rtqichen/torchdiffeq.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *CoRR*, abs/1806.07366, 2018. URL http://arxiv.org/abs/1806.07366.

Colin Graber and Alexander G. Schwing. Dynamic neural relational inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems, 2018.

Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8cbd005a556ccd4211ce43f309bc0eac-Paper.pdf.