EvolveGraph: Multi-Agent Trajectory Prediction with Dynamic Relational Reasoning

Jiachen Li[1,2,*†] **Fan Yang**[2,*] **Masayoshi Tomizuka**[2] **Chiho Choi**[1]

[1]Honda Research Institute, USA [2] University of California, Berkeley
{jiachen_li, fanyang16, tomizuka}@berkeley.edu cchoi@honda-ri.com

Abstract

Multi-agent interacting systems are prevalent in the world, from pure physical systems to complicated social dynamic systems. In many applications, effective understanding of the situation and accurate trajectory prediction of interactive agents play a significant role in downstream tasks, such as decision making and planning. In this paper, we propose a generic trajectory forecasting framework (named EvolveGraph) with explicit relational structure recognition and prediction via latent interaction graphs among multiple heterogeneous, interactive agents. Considering the uncertainty of future behaviors, the model is designed to provide multi-modal prediction hypotheses. Since the underlying interactions may evolve even with abrupt changes, and different modalities of evolution may lead to different outcomes, we address the necessity of dynamic relational reasoning and adaptively evolving the interaction graphs. We also introduce a double-stage training pipeline which not only improves training efficiency and accelerates convergence, but also enhances model performance. The proposed framework is evaluated on both synthetic physics simulations and multiple real-world benchmark datasets in various areas. The experimental results illustrate that our approach achieves state-of-the-art performance in terms of prediction accuracy.

1 Introduction

Multi-agent trajectory prediction is critical in many real-world applications, such as autonomous driving, mobile robot navigation and other areas where a group of entities interact with each other, giving rise to complicated behavior patterns at the level of both individuals and the multi-agent system as a whole. Since usually only the trajectories of individual entities are available without any knowledge of the underlying interaction patterns, and there are usually multiple possible modalities for each agent, it is challenging to model such dynamics and forecast their future behaviors.

There have been a number of existing works trying to provide a systematic solution to multi-agent interaction modeling. Some related techniques include, but not limited to social pooling layers [1], attention mechanisms [41, 18, 11, 39, 20], message passing over graphs [7, 36, 21], etc. These techniques can be summarized as implicit interaction modeling by information aggregation. Another line of research is to explicitly perform inference over the structure of the latent interaction graph, which allows for relational structures with multiple interaction types [17, 2]. Our approach falls into this category but with significant extension and performance enhancement over existing methods.

A closely related work is NRI [17], in which the interaction graph is static with homogeneous nodes during training. This is sufficient for the systems involving homogeneous type of agents with fixed

[*]indicates equal contribution

[†]Work done during Jiachen's internship at Honda Research Institute, USA.

Machine Learning for Autonomous Driving Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

interaction patterns. In many real-world scenarios, however, the underlying interactions are inherently varying even with abrupt changes (e.g. basketball players). And there may be heterogeneous types of agents (e.g. cars, pedestrians, cyclists, etc.) involved in the system, while NRI cannot distinguish them explicitly. Moreover, NRI does not deal with the multi-modality explicitly in future system behaviors. In this work, we address the problem of 1) extracting the underlying interaction patterns with a latent graph structure, which is able to handle different types of agents in a unified way, 2) capturing the dynamics of interaction graph evolution for dynamic relational reasoning, 3) predicting future trajectories (state sequences) based on the historical observations and the latent interaction graph, and 4) capturing the multi-modality of future system behaviors.

The main contributions of this paper are summarized as:
• We propose a generic trajectory forecasting framework with explicit interaction modeling via a latent graph among multiple heterogeneous, interactive agents. Both trajectory information and context information (e.g. scene images, semantic maps, point cloud density maps) can be incorporated into the system.
• We propose a dynamic mechanism to evolve the underlying interaction graph adaptively along time, which captures the dynamics of interaction patterns among multiple agents. We also introduce a double-stage training pipeline which not only improves training efficiency and accelerates convergence, but also enhances model performance in terms of prediction accuracy.
• The proposed framework is designed to capture the uncertainty and multi-modality of future trajectories in nature from multiple aspects.
• We validate the proposed framework on both synthetic simulations and trajectory forecasting benchmarks in different areas. Our EvolveGraph achieves the state-of-the-art performance consistently.

2 Related work

The problem of multi-agent trajectory prediction has been considered as modeling behaviors among a group of interactive agents. Social forces was introduced by [9] to model the attractive and repulsive motion of humans with respect to the neighbors. Some other learning-based approaches were proposed, such as hidden Markov models [22, 45], dynamic Bayesian networks [15], inverse reinforcement learning [38]. In recent years, the conceptual extension has been made to better model social behavior with supplemental cues such as motion patterns [47, 44] and group attributes [43]. Such social models have motivated the recent data-driven methods in [1, 19, 6, 42, 8, 46, 10, 23, 3, 49, 25, 33, 37, 31, 20, 5, 12, 27, 24]. They encode the motion history of individual entities using the recurrent operation of neural networks. However, it is nontrivial for these methods to find acceptable future motions in heterogeneous and interactively changing environments, partly due to their heuristic feature pooling or aggregation, which may not be sufficient for dynamic interaction modeling.

Interaction modeling and relational reasoning have been widely studied in various fields. Recently, deep neural networks applied to graph structures have been employed to formulate a connection between interactive agents or variables [41, 25, 18, 21, 35, 48]. These methods introduce nodes to represent interactive agents and edges to express their interactions with each other. They directly learn the evolving dynamics of node attributes (agents' states) and/or edge attributes (relations between agents) by constructing spatio-temporal graphs. However, their models have no explicit knowledge about the underlying interaction patterns. Some existing works (e.g. NRI [17]) have taken a step forward towards explicit relational reasoning by inferring a latent interaction graph. However, it is nontrivial for NRI to deal with heterogeneous agents, context information and the systems with varying interactions. In this work, we present an effective solution to handle aforementioned issues. Our work is also related to learning on dynamic graphs. Most existing works studied representation learning on dynamically evolving graphs [29, 16], while we attempt to predict evolution of the graph.

3 Problem formulation

We assume that, without loss of generality, there are N homogeneous or heterogeneous agents in the scene, which belongs to M (≥ 1) categories (e.g. cars, cyclists, pedestrians). The number of agents may vary in different cases. We denote a set of state sequences covering the historical and forecasting horizons (T_h and T_f) as $\mathbf{X}_{1:T} = \{\mathbf{x}^i_{1:T}, T = T_h + T_f, i = 1, ..., N\}$. We also denote a sequence of historical context information as $\mathbf{C}_{1:T_h} = \{\mathbf{c}_{1:T_h}\}$ for dynamic scenes or fixed context information $\mathbf{C}$ for static scenes. In the scope of this paper, we define $\mathbf{x}^i_t = (x^i_t, y^i_t)$, where (x, y) is the 2D coordinate in the world space or image pixel space. The context information includes

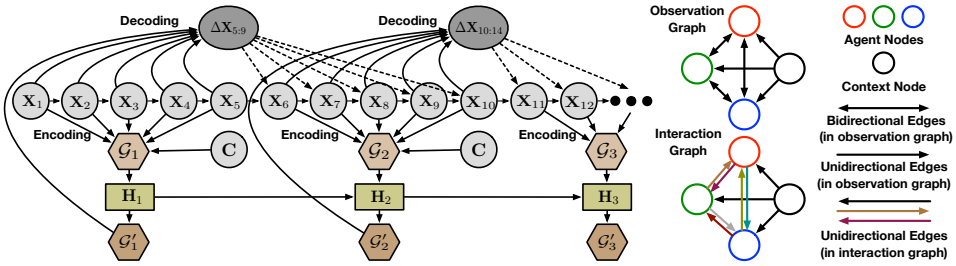

Figure 1: (a) The left part is a high-level graphical illustration of the proposed approach, where the encoding horizon and decoding horizons (re-encoding gap) are both set to 5. $\mathbf{X}_t$ denotes the state of all the agents at time t, $\Delta\mathbf{X}_t$ denotes the change in state, and $\mathbf{C}$ denotes context information. $\mathcal{G}_\beta$ denotes the latent interaction graph obtained from the static encoding process, and $\mathcal{G}'_\beta$ denotes the adjusted interaction graph with time dependence. At each encoding-decoding iteration, $\mathcal{G}_\beta$ is obtained through the encoding of previous trajectories and context information, which goes through a recurrent unit ($\mathbf{H}_\beta$) to get the adjusted interaction graph $\mathcal{G}'_\beta$. The previous trajectories and $\mathcal{G}'_\beta$ are combined as the input of the decoding process, which generates distributions of state changes to get future trajectories. (b) The right part is an illustration of observation graph and interaction graph. In the observation graph, the edges between agent nodes are homogeneous and bidirectional, while in the interaction graph, colored edges are unidirectional with a certain type. (Best viewed in color.)

images or tensors which represent attributes of the scene. We denote the latent interaction graph as $\mathcal{G}_\beta$, where β is the graph index. We aim to estimate $p(\mathbf{X}_{T_h+1:T_h+T_f}|\mathbf{X}_{1:T_h}, \mathbf{C}_{1:T_h})$ for dynamic scenes or $p(\mathbf{X}_{T_h+1:T_h+T_f}|\mathbf{X}_{1:T_h}, \mathbf{C})$ for static scenes. For simplicity, we use $\mathbf{C}$ when referring to the context information in the equations. More formally, if the latent interaction graph is inferred at each time step, then we have the factorization of $p(\mathbf{X}_{T_h+1:T_h+T_f}|\mathbf{X}_{1:T_h}, \mathbf{C})$ below:

$$\int_{\mathcal{G}} p(\mathcal{G}_0|\mathbf{X}_{1:T_h}, \mathbf{C}) p(\mathbf{X}_{T_h+1}|\mathcal{G}_0, \mathbf{X}_{1:T_h}, \mathbf{C}) \prod_{\beta=1}^{T_f-1} p(\mathcal{G}_\beta|\mathcal{G}_{0:\beta-1}, \mathbf{X}_{1:T_h+\beta}, \mathbf{C}) p(\mathbf{X}_{T_h+\beta+1}|\mathcal{G}_{0:\beta}, \mathbf{X}_{1:T_h+\beta}, \mathbf{C}).$$

4 EvolveGraph

An illustrative graphical model is shown in Figure 1 (left part) to demonstrate the essential procedures of the prediction framework with explicit dynamic relational reasoning. Instead of end-to-end training in a single pipeline, our training process contains two consecutive stages:

• ***Static interaction graph learning***: A series of encoding functions are trained to extract interaction patterns from the observed trajectories and context information, and generate a distribution of static latent interaction graphs. A series of decoding functions are trained to recurrently generate multi-modal distributions of future states. At this stage, the prediction is only based on the static interaction graph inferred from the history information, which means the encoding process is only applied once and the interaction graph does not evolve with the decoding process.

• ***Dynamic interaction graph learning***: The pre-trained encoding and decoding functions at the first stage are utilized as an initialization, which are finetuned together with the training of a recurrent network which captures the dynamics of interaction graph evolution. The graph recurrent network serves as a high-level integration which considers the dependency of the current interaction graph on previous ones. At this stage, the prediction is based on the latest updated interaction graph.

4.1 Static interaction graph learning

Observation Graph A fully-connected graph without self-loops is constructed to represent the observed information with node/edge attributes, which is called observation graph. Assume that there are N heterogeneous agents in the scene, which belongs to M categories. Then the observation graph consists of N agent nodes and one context node. Agent nodes are bidirectionally connected to each other, and the context node only have outgoing edges to each agent node. We denote an observation graph as $\mathcal{G}_{obs} = \{\mathcal{V}_{obs}, \mathcal{E}_{obs}\}$, where $\mathcal{V}_{obs} = \{\mathbf{v}_i, i \in \{1, ..., N\}\} \cup \{\mathbf{v}_c\}$ and $\mathcal{E}_{obs} = \{\mathbf{e}_{ij}, i, j \in \{1, ..., N\}\} \cup \{\mathbf{e}_{ic}, i \in \{1, ..., N\}\}$. $\mathbf{v}_i$, $\mathbf{v}_c$ and $\mathbf{e}_{ij}$, $\mathbf{e}_{ic}$ denote agent node attribute, context node attribute and agent-agent, context-agent edge attribute, respectively. More specifically, the $\mathbf{e}_{ij}$ denotes the attribute of the edge from node j to node i. Each agent node has two types of attributes: *self-attribute* and *social-attribute*. The former only contains the node's own

state information, while the latter only contains other nodes' state information. The calculations of node/edge attributes are given by

$$\mathbf{v}_i^{\text{self}} = f_a^m(\mathbf{x}_{1:T_h}^i),\ i \in \{1,...,N\},\ m \in \{1,...,M\}, \quad \mathbf{v}_c = f_c(\mathbf{c}_{1:T_h}) \quad \text{or} \quad \mathbf{v}_c = f_c(\mathbf{c}), \tag{1}$$

$$\mathbf{e}_{ij}^1 = f_e^1([\mathbf{v}_i^{\text{self}}, \mathbf{v}_j^{\text{self}}]),\ \mathbf{e}_{ic}^1 = f_{ec}^1([\mathbf{v}_i^{\text{self}}, \mathbf{v}_c]),\ \mathbf{v}_i^{\text{social-1}} = f_v^1([\textstyle\sum_{i\neq j} \alpha_{ij}\mathbf{e}_{ij}^1, \mathbf{e}_{ic}^1]),\ \textstyle\sum_{i\neq j}\alpha_{ij} = 1, \tag{2}$$

$$\mathbf{v}_i^1 = [\mathbf{v}_i^{\text{self}}, \mathbf{v}_i^{\text{social-1}}], \quad \mathbf{e}_{ij}^2 = f_e^2([\mathbf{v}_i^1, \mathbf{v}_j^1]), \quad \alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}(\mathbf{a}^\top[\mathbf{W}\mathbf{v}_i||\mathbf{W}\mathbf{v}_j])\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}(\mathbf{a}^\top[\mathbf{W}\mathbf{v}_i||\mathbf{W}\mathbf{v}_k])\right)}, \tag{3}$$

where α_{ij} are learnable attention coefficients computed similar as [40], $f_a^m(\cdot)$, $f_c(\cdot)$ are agent, context node embedding functions, and $f_e(\cdot)$, $f_{ec}(\cdot)$ and $f_v(\cdot)$ are agent-agent edge, agent-context edge, and agent node update functions, respectively. Different types of nodes (agents) use different embedding functions. Note that the attributes of the context node are never updated and the edge attributes only serve as intermediates for the update of agent node attributes. These $f(\cdot)$ functions are implemented by deep networks with proper architectures. At this time, we obtain a complete set of node/edge attributes which include the information of direct (first-order) interaction. The higher-order interactions can be modeled by multiple loops of equations (2)-(3), in which the social node attributes and edge attributes are updated by turns. Note that the *self-attribute* is fixed in the whole process.

Interaction Graph The interaction graph represents interaction patterns with a distribution of edge types for each edge, which is built on top of the observation graph. We set a hyperparameter L to denote the number of possible edge types (interaction types) between pairwise agent nodes to model *agent-agent* interactions. Also, there is another edge type that is shared between the context node and all agent nodes to model *agent-context* interactions. Note that "no edge" can also be treated as a special edge type, which implies that there is no message passing along such edges. More formally, the interaction graph is a discrete probability distribution $q(\mathcal{G}|\mathbf{X}_{1:T_h}, \mathbf{C}_{1:T_h})$ or $q(\mathcal{G}|\mathbf{X}_{1:T_h}, \mathbf{C})$, where $\mathcal{G} = \{\mathbf{z}_{ij}, i, j \in \{1,...,N\}\} \cup \{\mathbf{z}_{ic}, i \in \{1,...,N\}\}$ is a set of interaction types for all the edges, and $\mathbf{z}_{ij}$ and $\mathbf{z}_{ic}$ are random variables to indicate pairwise interaction types for a specific edge.

Encoding The goal of the encoding process is to infer a latent interaction graph from the observation graph, which is essentially a multi-class edge classification task. We employ a softmax function with a continuous approximation of the discrete distribution [26] on the last updated edge attributes to obtain the probability of each edge type, which is given by

$$q(\mathbf{z}_{ij}|\mathbf{X}_{1:T_h}, \mathbf{C}) = \text{Softmax}((\mathbf{e}_{ij}^2 + \mathbf{g})/\tau),\ i, j \in \{1,...,N\}, \tag{4}$$

where $\mathbf{g}$ is a vector of independent and identically distributed samples drawn from $\text{Gumbel}(0, 1)$ distribution and τ is the Softmax temperature, which controls the sample smoothness. We also use the repramatrization trick to obtain gradients for backpropagation. The edge type between context node and agent nodes $\mathbf{z}_{ic}$, without loss of generality, is hard-coded with probability one. For simplicity, we summarize all the operations in the observation graph and the encoding process as $q(\mathbf{z}|\mathbf{X}_{1:T_h}, \mathbf{C}) = f_{\text{enc}}(\mathbf{X}_{1:T_h}, \mathbf{C})$, which gives a factorized distribution of $\mathbf{z}_{ij}$.

Decoding Since in many real-world applications the state of agents has long-term dependence, a recurrent decoding process is applied to the interaction graph and observation graph to approximate the distribution of future trajectories $p(\mathbf{X}_{T_h+1:T_h+T_f}|\mathcal{G}, \mathbf{X}_{1:T_h}, \mathbf{C})$. The output at each time step is a Gaussian mixture distribution with K components, where the covariance of each Gaussian component is manually set equal. The detailed operations in the decoding process consists of two stages: burn-in stage ($1 \le t \le T_h$) and prediction stage ($T_h + 1 \le t \le T_h + T_f$), which are given by

$$\tilde{\mathbf{e}}_t^{ij} = \textstyle\sum_{l=1}^L z_{ij,l} f_e^l([\tilde{\mathbf{h}}_t^i, \tilde{\mathbf{h}}_t^j]), \quad \text{MSG}_t^i = \textstyle\sum_{j\neq i} \tilde{\mathbf{e}}_t^{ij}, \tag{5}$$

- $1 \le t \le T_h$ (Burn-in stage):

$$\tilde{\mathbf{h}}_{t+1}^i = \text{GRU}^i([\text{MSG}_t^i, \mathbf{x}_t^i, \mathbf{v}_c], \tilde{\mathbf{h}}_t^i), \quad w_{t+1}^{i,k} = f_{weight}^k(\tilde{\mathbf{h}}_{t+1}^i), \tag{6}$$

$$\boldsymbol{\mu}_{t+1}^{i,k} = \mathbf{x}_t^i + f_{\text{out}}^k(\tilde{\mathbf{h}}_{t+1}^i), \quad p(\hat{\mathbf{x}}_{t+1}^i|\mathbf{z}, \mathbf{x}_{1:t}^i, \mathbf{c}) = \textstyle\sum_{k=1}^K w_{t+1}^{i,k}\mathcal{N}(\boldsymbol{\mu}_{t+1}^{i,k}, \sigma^2\mathbf{I}), \tag{7}$$

- $T_h + 1 \le t \le T_h + T_f$ (Prediction stage):

$$\tilde{\mathbf{h}}_{t+1}^i = \text{GRU}^i([\text{MSG}_t^i, \hat{\mathbf{x}}_t^i, \mathbf{v}_c], \tilde{\mathbf{h}}_t^i),\ w_{t+1}^{i,k} = f_{weight}^k(\tilde{\mathbf{h}}_{t+1}^i),\ \boldsymbol{\mu}_{t+1}^{i,k} = \hat{\mathbf{x}}_t^i + f_{\text{out}}^k(\tilde{\mathbf{h}}_{t+1}^i), \tag{8}$$

$$p(\hat{\mathbf{x}}_{t+1}^i|\mathbf{z}, \hat{\mathbf{x}}_{T_h+1:t}^i, \mathbf{x}_{1:T_h}^i, \mathbf{c}) = \textstyle\sum_{k=1}^K w_{t+1}^{i,k}\mathcal{N}(\boldsymbol{\mu}_{t+1}^{i,k}, \sigma^2\mathbf{I}), \tag{9}$$

$$\text{Sample a Gaussian component from the mixture based on } \mathbf{w}_{t+1}^i, \quad \text{Set } \hat{\mathbf{x}}_{t+1}^i = \boldsymbol{\mu}_{t+1}^{i,k}, \tag{10}$$

where MSG is a symbolic acronym for "message" here without specific meanings, $\tilde{\mathbf{h}}_t^i$ is the hidden state of GRU^i at time t, $w_{t+1}^{i,k}$ is the weight of the kth Gaussian distribution at time step $t+1$ for agent i. $\tilde{f}_e^l(\cdot)$ is the edge update function of edge type l, $f_{weight}^k(\cdot)$ is a mapping function to get the weight of the kth Gaussian distribution, and $f_{\text{out}}^k(\cdot)$ is a mapping function to get the mean of the kth Gaussian component. Note that the predicted $\hat{\mathbf{x}}_t^i$ is needed in equation (8). In the previous decoding step, we only have its corresponding distribution $p(\hat{\mathbf{x}}_t^i|\mathbf{z}, \hat{\mathbf{x}}_{T_h+1:t-1}^i, \mathbf{x}_{1:T_h}^i, \mathbf{c})$ from the previous step. We first sample a Gaussian component from the mixture based on the component weights w_{t+1}^i. Say we get the kth component, then we set $\hat{\mathbf{x}}_t^j$ as $\boldsymbol{\mu}_t^{j,k}$, which is the trajectory with maximum likelihood within this component. We fix the covariance σ as a constant. The nodes (agents) of the same type share the same GRU decoder. During the burn-in stage, the ground-truth states are used; while during the prediction stage, the state prediction hypotheses are used as the input at the next time step iteratively. For simplicity, the whole decoding process is summarized as $p(\mathbf{X}_{T_h+1:T_h+T_f}|\mathcal{G}, \mathbf{X}_{1:T_h}, \mathbf{C}) = f_{\text{dec}}(\mathcal{G}, \mathbf{X}_{1:T_h}, \mathbf{C})$.

4.2 Dynamic interaction graph

In many applications, the interaction patterns recognized from the past time steps are likely not static in the future. Instead, they are rather dynamically evolving throughout the future time steps. Moreover, many interaction systems have multi-modal properties in its nature. Different modalities afterwards are likely to result in different interaction patterns. A single static interaction graph is neither sufficiently flexible to model dynamically changing situations (especially those with abrupt changes), nor to capture all the modalities. Therefore, we introduce an effective dynamic mechanism to evolve the interaction graph.

The encoding process is repeated every τ (re-encoding gap) time steps to obtain the latent interaction graph based on the latest observation graph. Since the new interaction graph also has dependence on previous ones, we also need to consider their effects. Therefore, a recurrent unit (GRU) is utilized to maintain and propagate the history information, as well as adjust the prior interaction graphs. More formally, the calculations are given by

$$q(\mathbf{z}_\beta|\mathbf{X}_{1+\beta\tau:T_h+\beta\tau}, \mathbf{C}) = f_{\text{enc}}(\mathbf{X}_{1+\beta\tau:T_h+\beta\tau}, \mathbf{C}), \tag{11}$$

$$q(\mathbf{z}'_\beta|\mathbf{X}_{1+\beta\tau:T_h+\beta\tau}, \mathbf{C}) = \text{GRU}(q(\mathbf{z}_\beta|\mathbf{X}_{1+\beta\tau:T_h+\beta\tau}, \mathbf{C}), \mathbf{H}_\beta) \tag{12}$$

where β is the re-encoding index starting from 0, $\mathbf{z}_\beta$ is the interaction graph obtained from the static encoding process, $\mathbf{z}'_\beta$ is the adjusted interaction graph with time dependence, and $\mathbf{H}_\beta$ is the hidden state of the graph evolution GRU. After obtaining $\mathcal{G}'_\beta = \{\mathbf{z}'_\beta\}$, the decoding process is applied to get the states of the next τ time steps,

$$p(\mathbf{X}_{T_h+\beta\tau+1:T_h+(\beta+1)\tau}|\mathcal{G}'_\beta, \mathbf{X}_{1:T_h}, \hat{\mathbf{X}}_{T_h+1:T_h+\beta\tau}, \mathbf{C}) = f_{\text{dec}}(\mathcal{G}'_\beta, \mathbf{X}_{1:T_h}, \hat{\mathbf{X}}_{T_h+1:T_h+\beta\tau}, \mathbf{C}). \tag{13}$$

The decoding and re-encoding processes are iterated to obtain the distribution of future trajectories.

4.3 Uncertainty and multi-modality

Here we emphasize the efforts to encourage diverse and multi-modal trajectory prediction and generation. In our framework, the uncertainty and multi-modality mainly come from three aspects. First, in the decoding process, we output Gaussian mixture distributions indicating that there are several possible modalities at the next step. We only sample a single Gaussian component at each step based on the component weights which indicate the probability of each modality. Second, different sampled trajectories will lead to different interaction graph evolution. Evolution of interaction graphs contributes to the multi-modality of future behaviors, since different underlying relational structures enforce different regulations on the system behavior and lead to various outcomes. Third, directly training such a model, however, tends to collapse to a single mode. Therefore, we employ an effective mechanism to mitigate the mode collapse issue and encourage multi-modality. During training, we run the decoding process d times, which generates d trajectories for each agent under specific scenarios. We only choose the prediction hypothesis with the minimal loss for backpropagation, which is the most likely to be in the same mode as the ground truth. The other prediction hypotheses may have much higher loss, but it doesn't necessarily imply that they are implausible. They may represent other potential reasonable modalities.

Table 1: Comparison of Accuracy (Mean ± Std in %) of Interaction (Edge Type) Recognition.

	Corr. (LSTM)	NRI (dynamic)	**EvolveGraph** (static)	**EvolveGraph** (RNN re-encoding)	**EvolveGraph** (dynamic)	Supervised
No Change	63.2±0.9	91.3±0.3	**95.6**±0.2	91.4±0.3	93.8±1.1	98.1±0.4
Change	—	71.5±3.1	64.1±0.8	75.2±1.4	**82.3**±3.2	94.3±1.5

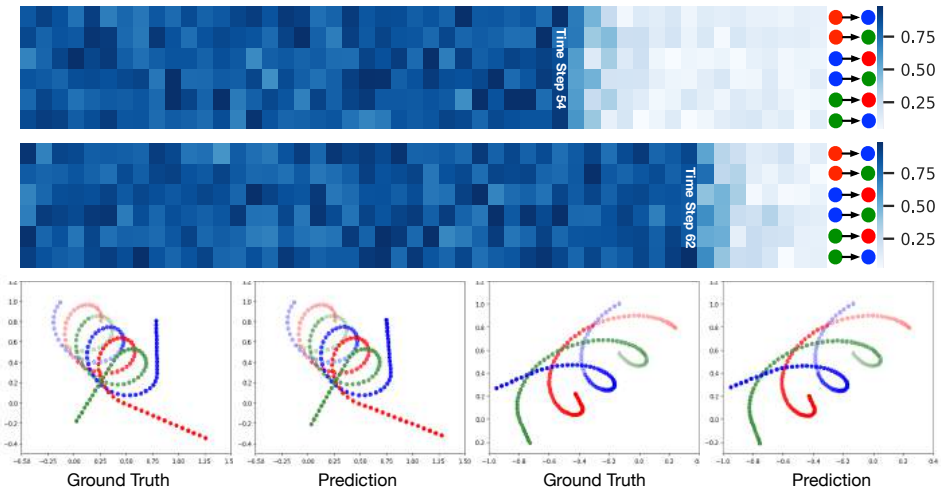

Figure 2: Visualization of latent interaction graph evolution and particle trajectories. (a) The top two figures show the probability of the first edge type ("with link") at each time step. Each row corresponds to a certain edge (shown in the right). The actual times of graph evolution are 54 and 62, respectively. The model is able to capture the underlying criterion of relation change and further predict the change of edge types with nearly no delay. (b) The figures in the last row show trajectory prediction results, where semi-transparent dots are historical observations.

5 Experiments

In this paper, we validated the proposed framework EvolveGraph on one synthetic dataset and three benchmark datasets for real-world applications: Honda 3D Dataset (H3D) [30], NBA SportVU Dataset (NBA), and Stanford Drone Dataset (SDD) [32]. The dataset details, baseline approaches, as well as implementation details are introduced in supplementary materials.

For the synthetic dataset, since we have access to the ground truth of the underlying interaction graph, we quantitatively and qualitatively evaluate the model performance in terms of both interaction (edge type) recognition and average state prediction error. For the benchmark datasets, we evaluate the model performance in terms of two widely used standard metrics: minimum average displacement error (minADE_{20}) and minimum final displacement error (minFDE_{20}) [3]. The minADE_{20} is defined as the minimum average distance between the 20 predicted trajectories and the ground truth over all the involved entities within the prediction horizon. The minFDE_{20} is defined as the minimum deviated distance of 20 predicted trajectories at the last predicted time step. We also provide ablative analysis (right part of Table 2-4), analysis on double-stage training, analysis on the selection of edge types and re-encoding gap, and additional qualitative results in supplementary materials.

5.1 Synthetic simulations: particle physics system

We experimented with a simulated particle system with change of relations. Multiple particles are initially linked and move together. The links disappear as long as a certain criterion on particle state is satisfied and the particles move independently thereafter. The model is expected to learn the criterion by itself, and perform edge type prediction and trajectory prediction. Since the system is deterministic in nature, we do not consider multi-modality in this task. Further details on the dataset generation are introduced in Section 8.1.1 in the supplementary materials.

We predicted the particle states at the future 50 time steps based on the observations of 20 time steps. We set two edge types in this task, which correspond to "with link" and "without link". The results of edge type prediction are summarized in Table 1, which are averaged over 3 independent runs. *No Change* means the underlying interaction structure keeps the same in the whole horizon,

Table 2: minADE_{20} / minFDE_{20} (Meters) of Trajectory Prediction (H3D dataset).

Time	Baseline Methods							**EvolveGraph (Ours)**				
	STGAT	Social-Attention	Social-STGCNN	Social-GAN	Gated-RN	Trajectron++	NRI (dynamic)	SG (same node type)	SG	RNN re-encoding	DG (single stage)	DG (double stage)
1.0s	0.24 / 0.33	0.29 / 0.45	0.23 / 0.32	0.27 / 0.37	**0.18** / 0.32	0.21 / 0.34	0.24 / 0.30	0.28 / 0.37	0.27 / 0.35	0.25 / 0.32	0.24 / 0.31	0.19 / **0.25**
2.0s	0.34 / 0.48	0.53 / 0.96	0.36 / 0.52	0.45 / 0.77	0.32 / 0.64	0.33 / 0.62	0.32 / 0.60	0.40 / 0.58	0.38 / 0.55	0.35 / 0.51	0.33 / 0.46	**0.31 / 0.44**
3.0s	0.46 / 0.77	0.87 / 1.62	0.49 / 0.89	0.68 / 1.29	0.49 / 1.03	0.46 / 0.93	0.48 / 0.94	0.51 / 0.80	0.48 / 0.76	0.44 / 0.70	0.40 / 0.60	**0.39 / 0.58**
4.0s	0.60 / 1.18	1.21 / 2.56	0.73 / 1.49	0.94 / 1.91	0.69 / 1.56	0.71 / 1.63	0.73 / 1.56	0.64 / 1.21	0.61 / 1.14	0.57 / 1.07	0.50 / 0.90	**0.48 / 0.86**

Table 3: minADE_{20} / minFDE_{20} (Meters) of Trajectory Prediction (NBA dataset).

Time	Baseline Methods							**EvolveGraph (Ours)**				
	STGAT	Social-STGCNN	Social-Attention	Social-LSTM	Social-GAN	Trajectron++	NRI (dynamic)	SG (same node type)	SG	RNN re-encoding	DG (single stage)	DG (double stage)
1.0s	0.42 / 0.71	0.46 / 0.76	0.87 / 1.36	0.92 / 1.34	0.82 / 1.25	0.55 / 0.90	0.60 / 0.87	0.70 / 1.09	0.59 / 0.92	0.58 / 0.89	0.48 / 0.76	**0.31 / 0.52**
2.0s	0.91 / 1.39	0.90 / 1.43	1.58 / 2.51	1.64 / 2.74	1.52 / 2.45	0.99 / 1.58	1.02 / 1.71	1.51 / 2.38	1.38 / 2.12	1.09 / 1.88	0.84 / 1.43	**0.74 / 1.10**
3.0s	1.62 / 2.87	1.59 / 2.67	2.78 / 4.66	2.93 / 5.03	2.63 / 4.51	1.89 / 3.32	1.83 / 3.15	2.10 / 3.53	1.88 / 3.23	1.77 / 2.87	1.43 / 2.55	**1.28 / 2.07**
4.0s	2.47 / 3.86	2.35 / 3.71	3.76 / 6.64	4.00 / 7.12	3.60 / 6.24	2.62 / 4.70	2.48 / 4.30	2.83 / 4.85	2.52 / 4.57	2.39 / 3.89	2.08 / 3.74	**1.83 / 3.16**

Table 4: minADE_{20} / minFDE_{20} (Pixels) of Trajectory Prediction (SDD dataset).

Time	Baseline Methods							**EvolveGraph (Ours)**				
	STGAT	Social-STGCNN	Social-Attention	Social-LSTM	Social-GAN	Trajectron++	NRI (dynamic)	SG (same node type)	SG	RNN re-encoding	DG (single stage)	DG (double stage)
4.8s	18.8 / 31.3	20.6 / 33.1	33.3 / 55.9	31.4 / 55.6	27.0 / 43.9	19.3 / 32.7	25.6 / 43.7	22.5 / 40.3	20.6 / 36.4	18.4 / 32.1	16.1 / 26.6	**13.9 / 22.9**

while *Change* means the change of interaction patterns happens at some time. It shows that the supervised learning baseline, which directly trains the encoding functions with ground truth labels, performs the best in both setups and serves as a "gold standard". Under the *No Change* setup, NRI (dynamic) is comparable to EvolveGraph (RNN re-encoding), while EvolveGraph (static) achieves the best performance. The reason is that dynamic evolution of interaction graph leads to higher flexibility but may result in larger uncertainty, which affects edge prediction in the systems with static relational structures. Under the *Change* setup, NRI (dynamic) re-evaluates the latent graph at every time step during the testing phase, but it is hard to capture the dependency between consecutive graphs, and the encoding functions may not be flexible enough to capture the evolution. EvolveGraph (RNN re-encoding) performs better since it considers dependency of consecutive steps during the training phase, but it still captures the evolution only at the feature level instead of the graph level. EvolveGraph (dynamic) achieves a significantly higher accuracy than the other baselines (except Supervised), due to the explicit evolution of interaction graphs.

We also provide visualization of interaction graphs and particle trajectories of random testing cases in Figure 2. In the heatmaps, despite that the predicted probabilities fluctuate within a small range at each step, they are very close to the ground truth (1 for "with link" and 0 for "without link"). The change of relation can be quickly captured within two time steps. The results of particle state prediction are shown in Figure 3. The standard deviation was calculated over 3 runs. Within the whole horizon, EvolveGraph (dynamic) consistently outperforms the other baselines with stable performance (small standard deviation).

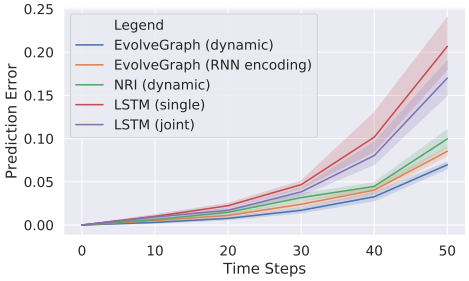

Figure 3: Average error of particle state.

5.2 H3D dataset: traffic scenarios

We predicted the future 10 time steps (4.0s) based on the historical 5 time steps (2.0s). The comparison of quantitative results is shown in Table 2, where the unit of reported minADE_{20} and minFDE_{20} is meters in the world coordinates. Note that we included cars, trucks, cyclists and pedestrians in the experiments. All the baseline methods consider the relations and interactions among agents. The Social-Attention employs spatial attention mechanisms, while the Social-GAN demonstrates a deep generative model which learns the data distribution to generate human-like trajectories. The Gated-RN and Trajectron++ both leverage spatio-temporal information to involve relational reasoning, which leads to smaller prediction error. The NRI infers a latent interaction graph and learns the dynamics of agents, which achieves similar performance to Trajectron++. The STGAT and Social-STGCNN further take advantage of the graph neural network to extract relational features in the multi-agent setting. Our proposed method achieves the best performance, which implies the advantages of

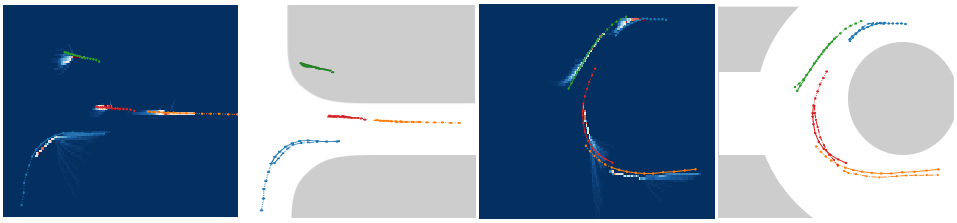

Figure 4: Qualitative results of testing cases of H3D dataset. Dashed lines are historical trajectories, solid lines are ground truth, and dash-dotted lines are prediction hypothesis. White areas represent drivable areas and gray areas represent sidewalks. We plotted the prediction hypothesis with the minimal ADE, and the heatmap to represent the distributions. (a) Intersection; (b) Roundabout.

explicit interaction modeling via evolving interaction graphs. The 4.0s minADE_{20} / minFDE_{20} are *significantly* reduced by 20.0% / 27.1% compared to the best baseline approach (STGAT).

We also provide visualization of results. Figure 4(a) and Figure 4(b) show two random testing samples from H3D results. We can tell that our framework can generate accurate and plausible trajectories. More specifically, in Figure 4(a), for the blue prediction hypothesis at the left bottom, there is an abrupt change at the fifth prediction step. This is because the interaction graph evolved at this step (Our re-encoding gap τ was set to be 5 in this case). Moreover, in the heatmap, there are multiple possible trajectories starting from this point, which represent multiple potential modalities. These results show that the evolving interaction graph can reinforce the multi-modal property of our model, since different samples of trajectories at the previous steps lead to different directions of graph evolution, which significantly influences the prediction afterwards. In Figure 4(b), each car may leave the roundabout at any exit. Our model can successfully show the modalities of exiting the roundabout and staying in it. Moreover, if exiting the roundabout, the cars are predicted to exit on their right, which implies that the modalities predicted by our model are plausible and reasonable.

5.3 NBA dataset: sports games

We also predicted the future 10 time steps (4.0s) based on the historical 5 time steps (2.0s). The comparison of quantitative results is shown in Table 3, where the unit of reported minADE_{20} and minFDE_{20} is meters in the world coordinates. Note that we included both players and the basketball in the experiments. The players are divided into two different types according to their teams. The basketball players are highly interactive and behaviors often change suddenly due to the reaction to other players. The baselines all consider the relations and interactions among agents with different strategies, such as soft attention mechanisms, social pooling layers, and graph-based representation. Owing to the dynamic interaction modeling by evolving interaction graph, our method achieves *significantly* better performance than state-of-the-art, which reduces the 4.0s minADE_{20} / minFDE_{20} by 22.1% / 18.1% with respect to the best baseline (Social-STGCNN). Qualitative results and analysis can be found in Section 7.3 in the supplementary materials.

5.4 SDD dataset: university campus

We predicted the future 12 time steps (4.8s) based on the historical 8 time steps (3.2s). The comparison of quantitative results is shown in Table 4, where the unit of reported minADE_{20} and minFDE_{20} is pixels in the image coordinates. Note that we included all the types of agents (e.g. pedestrians, cyclists, vehicles)in the experiments, although most of them are pedestrians. Our proposed method achieves the best performance. The 4.8s minADE_{20} / minFDE_{20} are reduced by 26.1% / 26.8% compared to the best baseline approach (STGAT).

6 Conclusions

In this paper, we present a generic trajectory forecasting framework with explicit relational reasoning among multiple heterogeneous, interactive agents with a graph representation. Multiple types of context information (e.g. static / dynamic, scene images / point cloud density maps) can be incorporated in the framework together with the trajectory information. In order to capture the underlying dynamics of the evolution of relational structures, we propose a dynamic mechanism to evolve the interaction graph, which is trained in two consecutive stages. The double-stage training mechanism can both speed up convergence and enhance prediction performance. The method is able

to capture the multi-modality of future behaviors. The framework is validated by synthetic physics simulations and multiple trajectory forecasting benchmarks for different applications, which achieves state-of-the-art performance in terms of prediction accuracy. For the future work, we will handle the prediction task involving a time-varying number of agents with an extended adaptive framework. EvolveGraph can also be applied to find the underlying patterns of large-scale interacting systems which involve a large number of entities, such as very complex physics systems.

Broader Impact

In this work, the authors introduce EvolveGraph, a generic trajectory prediction framework with dynamic relational reasoning, which can handle evolving interacting systems involving multiple heterogeneous, interactive agents. The proposed framework could be applied to a wide range of applications, from pure physics systems to complex social dynamics systems. In this paper, we demonstrate some illustrative applications to physics objects, traffic participants, and sports players. The framework could also be applied to analyze and predict the evolution of larger interacting systems, such as social networks and traffic flows. Although there are existing works using graph neural networks to handle trajectory prediction tasks, here we emphasize the impact of using our framework to recognize and predict the evolution of the underlying relations. With accurate and reasonable relational structures, we can forecast or generate plausible system behaviors, which help much with optimal decision making. However, if the predicted relational structures are wrong or misleading, the prediction performance may be degraded since the forecast highly depends on relational structures. There is no guarantee that such frameworks are able to work well on all kinds of applications. Therefore, users are expected to assess the applicability and risk for specific purpose.

References

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[2] Ferran Alet, Erica Weng, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Neural relational inference with fast modular meta-learning. In *Advances in Neural Information Processing Systems*, pages 11804–11815, 2019.

[3] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Conference on Robot Learning*, pages 86–99, 2020.

[4] Chiho Choi and Behzad Dariush. Looking to relations for future trajectory forecast. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 921–930, 2019.

[5] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020.

[6] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018.

[7] Nicholas Guttenberg, Nathaniel Virgo, Olaf Witkowski, Hidetoshi Aoki, and Ryota Kanai. Permutation-equivariant neural networks applied to dynamics prediction. *arXiv preprint arXiv:1612.04530*, 2016.

[8] Irtiza Hasan, Francesco Setti, Theodore Tsesmelis, Alessio Del Bue, Fabio Galasso, and Marco Cristani. Mx-lstm: mixing tracklets and vislets to jointly forecast trajectories and head poses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6067–6076, 2018.

[9] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[10] Joey Hong, Benjamin Sapp, and James Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8454–8462, 2019.

[11] Yedid Hoshen. Vain: Attentional multi-agent predictive modeling. In *Advances in Neural Information Processing Systems*, pages 2701–2711, 2017.

[12] Xin Huang, Stephen G McGill, Brian C Williams, Luke Fletcher, and Guy Rosman. Uncertainty-aware driver trajectory prediction at urban intersections. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9718–9724. IEEE, 2019.

[13] Yingfan Huang, Huikun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6272–6281, 2019.

[14] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5308–5317, 2016.

[15] Dietmar Kasper, Galia Weidl, Thao Dang, Gabi Breuel, Andreas Tamke, Andreas Wedel, and Wolfgang Rosenstiel. Object-oriented bayesian networks for detection of lane change maneuvers. *IEEE Intelligent Transportation Systems Magazine*, 4(3):19–31, 2012.

[16] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Relational representation learning for dynamic (knowledge) graphs: A survey. *arXiv preprint arXiv:1905.11485*, 2019.

[17] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697, 2018.

[18] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, Hamid Rezatofighi, and Silvio Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. In *Advances in Neural Information Processing Systems*, pages 137–146, 2019.

[19] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017.

[20] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. Conditional generative neural system for probabilistic trajectory prediction. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6150–6156. IEEE, 2019.

[21] Jiachen Li, Hengbo Ma, Zhihao Zhang, and Masayoshi Tomizuka. Social-wagdat: Interaction-aware trajectory prediction via wasserstein graph double-attention network. *arXiv preprint arXiv:2002.06241*, 2020.

[22] Jiachen Li, Wei Zhan, Yeping Hu, and Masayoshi Tomizuka. Generic tracking and probabilistic prediction framework and its application in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3634–3649, 2020.

[23] Chao Lu, Fengqing Hu, Dongpu Cao, Jianwei Gong, Yang Xing, and Zirui Li. Transfer learning for driver model adaptation in lane-changing scenarios using manifold alignment. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[24] Hengbo Ma, Jiachen Li, Wei Zhan, and Masayoshi Tomizuka. Wasserstein generative learning with kinematic constraints for probabilistic interactive driving behavior prediction. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2477–2483. IEEE, 2019.

[25] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6120–6127, 2019.

[26] C Maddison, A Mnih, and Y Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

[27] Srikanth Malla, Behzad Dariush, and Chiho Choi. Titan: Future forecast using action priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11186–11196, 2020.

[28] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14424–14432, 2020.

[29] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B Schardl, and Charles E Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *AAAI*, pages 5363–5370, 2020.

[30] Abhishek Patil, Srikanth Malla, Haiming Gang, and Yi-Ting Chen. The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes. In *International Conference on Robotics and Automation*, 2019.

[31] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2821–2830, 2019.

[32] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision*, pages 549–565. Springer, 2016.

[33] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1349–1358, 2019.

[34] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. In *Proceedings of Europe Conference on Computer Vision (ECCV)*, 2020.

[35] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the International Conference on Machine Learning*, pages 11648–11657, 2020.

[36] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

[37] Shan Su, Cheng Peng, Jianbo Shi, and Chiho Choi. Potential field: Interpretable and unified representation for trajectory prediction. *arXiv preprint arXiv:1911.07414*, 2019.

[38] Liting Sun, Wei Zhan, and Masayoshi Tomizuka. Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2111–2117. IEEE, 2018.

[39] Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *International Conference on Learning Representations*, 2018.

[40] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[41] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018.

[42] Yanyu Xu, Zhixin Piao, and Shenghua Gao. Encoding crowd interaction with deep neural network for pedestrian trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5275–5284, 2018.

[43] Kota Yamaguchi, Alexander C Berg, Luis E Ortiz, and Tamara L Berg. Who are you with and where are you going? In *CVPR 2011*, pages 1345–1352. IEEE, 2011.

[44] Shuai Yi, Hongsheng Li, and Xiaogang Wang. Understanding pedestrian behaviors from stationary crowd groups. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3488–3496, 2015.

[45] Wei Zhan, Liting Sun, Yeping Hu, Jiachen Li, and Masayoshi Tomizuka. Towards a fatality-aware benchmark of probabilistic reaction prediction in highly interactive driving scenarios. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3274–3280. IEEE, 2018.

[46] Pu Zhang, Wanli Ouyang, Pengfei Zhang, Jianru Xue, and Nanning Zheng. Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12085–12094, 2019.

[47] Yanhao Zhang, Lei Qin, Hongxun Yao, and Qingming Huang. Abnormal crowd behavior detection based on social attribute-aware force model. In *2012 19th IEEE International Conference on Image Processing*, pages 2689–2692. IEEE, 2012.

[48] Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Multivariate time-series anomaly detection via graph attention network. *arXiv preprint arXiv:2009.02040*, 2020.

[49] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12126–12134, 2019.

7 Additional Experimental Results and Further Analysis

In this section, we provide further experimental results and analysis, including ablative analysis, analysis on the selection of edge types and re-encoding gap, as well as additional qualitative results.

7.1 Ablative Analysis

We conducted ablative analysis on the benchmark datasets to demonstrate the effectiveness of heterogeneous node types, dynamically evolving interaction graph and two-stage graph learning. The best minADE_{20} / minFDE_{20} of each model setting are shown in the right parts of Table 2, Table 3 and Table 4. The descriptions of each model setup are provided in **Section 8.2**.

• **SG (same node type) v.s. SG**: We show the effectiveness of the distinction of agent node types. According to the prediction results in Table 2, Table 3 and Table 4, utilizing distinct agent-node embedding functions for different agent types achieves consistently smaller minADE_{20} / minFDE_{20} than a universal embedding function. The reason is that different types of agents have distinct behavior patterns or feasibility constraints. For example, the trajectories of on-road vehicles are restricted by roadways, traffic rules and physical constraints, while the restrictions on pedestrian behaviors are much fewer. Moreover, since vehicles usually have to yield pedestrians at intersections, it is helpful to indicate agent types explicitly in the model. With differentiation of agent types, the 4.0s minADE_{20} / minFDE_{20} are reduced by 4.7% / 5.8% on the H3D dataset, 8.6% / 5.8% on the NBA dataset. The 4.8s minADE_{20} / minFDE_{20} are reduced by 8.9% / 9.9% on the SDD dataset.

• **SG v.s. RNN re-encoding v.s. DG (double stage)**: We compare the performance of our method and SG / RNN encoding baselines. It is shown that the improvement of RNN re-encoding is limited and the prediction errors are slightly smaller than SG. Although both RNN re-encoding baseline and our method attempt to capture the potential changes of underlying interaction graph at each time step, our method achieves a significantly smaller prediction error consistently. A potential reason is despite that the RNN re-encoding process is iteratively extracting the patterns from node attributes (agent states), its capability of inferring graph evolution is limited.

• **DG (single stage) v.s. DG (double stage)**: We show the effectiveness and necessity of double-stage dynamic graph learning. It is shown that the double-stage training scheme leads to remarkable improvement in terms of minADE_{20} / minFDE_{20} on all three datasets. During the first training stage, the encoding / decoding functions are well trained to a local optimum, which is able to extract a proper static interaction graph. According to empirical findings, the encoding / decoding functions are sufficiently good as an initialization for the second stage training after several epochs' training. During the second training stage, the encoding / decoding functions are initialized from the first stage and finetuned, along with the training of graph evolution GRU. This leads to faster convergence and better performance, since it may help avoid some bad local optima at which the loss function may be stuck if all the components are randomly initialized. With the same hyperparameters, the single-stage / double-stage training took about 25 / 14 epochs to reach their smallest validation loss on the NBA dataset and 41 / 26 epochs on the H3D dataset. Compared to single-stage training, the 4.0s minADE_{20} / minFDE_{20} of double-stage training are reduced by 12.0% / 15.5% on the NBA dataset and 9.4% / 12.2% on the H3D dataset. The 4.8s minADE_{20} / minFDE_{20} are reduced by 13.7% / 13.9% on the SDD dataset.

7.2 Analysis on Edge Types and Re-encoding Gap

We also provide a comparison of minADE_{20} / minFDE_{20} (in meters) and testing running time on the NBA dataset to demonstrate the effect of different numbers of edge types and re-encoding gaps. In Figure 5(a), it is shown that as the number of edge type increases, the prediction error first decreases to a minimum and then increases, which implies too many edge types may lead to overfitting issues, since some edge types may capture subtle patterns from data which reduces generalization ability. The cross-validation is needed to determine the number of edge types. In Figure 5(b), it is illustrated that the prediction error increases consistently as the re-encoding gap raises, which implies more frequent re-identification of underlying interaction pattern indeed helps when it evolves along time. However, we need to trade off between the prediction error and testing running time if online prediction is required. The variance of minADE_{20} / minFDE_{20} in both figures are small, which implies the model performance is stable with random initialization and various settings in multiple experiments.

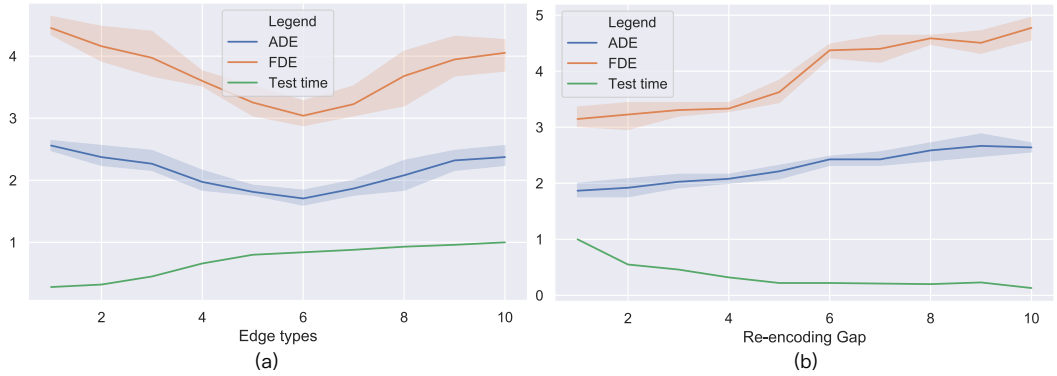

Figure 5: The comparison of minADE_{20} / minFDE_{20} (in meters) and testing running time of different model settings on the NBA dataset. We trained three models for each setting to illustrate the robustness of the method. (a) Different numbers of edge types; (b) Different re-encoding gaps. The testing running time is re-scaled to [0,1] for better illustration.

7.3 Additional Qualitative Results and Analysis

Figure 6 and Figure 7 show more visualizations of testing results on the H3D and NBA datasets. First, we tell that in such cases the ball follows a player at most times, which implies that the predicted results represent plausible situations. Second, most prediction hypotheses are very close to the ground truth, even if some predictions are not similar to the ground truth, they represent a plausible behavior. Third, the heatmaps show that our model can successfully predict most reasonable future trajectories and their multi-modal distributions. More specifically, in the first case of Figure 7, for the player of the green team in the middle, the historical steps move forward quickly, while our model can successfully predict that the player will suddenly stop, since he is surrounded by many opponents and he is not carrying the ball. In the second case of Figure 7, our model shows that three pairs of players from different teams competing against each other for chances. the defensing team is closer to the basket. and the player carrying the ball is running quickly towards the basket. Two opponents are trying to defend him. Such case is a very common situation in basketball games. In general, not only does our model achieve high accuracy, it can also understand and predict most moving, stopping, offending and defensing behaviors in basketball games.

8 Further Experimental Details

In this section, we provide important further details of experiments, which includes dataset generation, baseline approaches, as well as implementation details.

8.1 Datasets

8.1.1 Synthetic Particle Simulations

We designed a synthetic particle simulation to validate the performance of our model. In this simulation, we have n particles in an x-y plane and all the locations of particles are randomly initialized on the $y > 0$ half plane. The movement of these particles contains two phases, corresponding to two interaction graphs. Initially, particles are rigidly connected to each other and form a "star" shape. More specifically, there is a virtual centroid and each particle is rigidly connected to the centroid. it is equivalent to using a stick to connect the particle and the virtual centroid. And the distance between a certain particle and the centroid keeps the same. The particles are uniformly distributed around the centroid, which means the angle between two adjacent "sticks" is $\frac{2\pi}{n}$. In the first phase, particles move as a whole, with both translational and rotational motions. The velocity and angular velocity of the whole system are randomly initialized in a certain range. Once any one of the particles reaches $y = 0$ (the switching criterion), the movement of all the particles will transfer to the second phase. In the second phase, particles are no longer connected to each other. The motion of one particle will by

no means affect the motion of the others. In other words, each particle keeps uniform linear motions once the second phase begins. We generated 50k sample in total for training, validation and testing.

8.1.2 Benchmark Datasets

- H3D [30]: A large scale full-surround 3D multi-object detection and tracking dataset, which provides point cloud information and trajectory annotations for heterogeneous traffic participants (e.g. cars, trucks, cyclists and pedestrians). We selected 90k samples in total for training, validation and testing.
- NBA: A trajectory dataset collected by NBA with the SportVU tracking system, which contains the trajectory information of all the ten players and the ball in real games. We randomly selected 50k samples in total for training, validation and testing.
- SDD [32]: A trajectory dataset containing a set of top-down-view images and the corresponding trajectories of involved entities, which was collected in multiple scenarios in a university campus full of interactive pedestrians, cyclists and vehicles. We randomly selected 50k samples in total for training, validation and testing.

8.2 Baseline Methods

We compared the performance of our proposed approach with the following baseline methods. Please refer to the reference papers for more details.

8.2.1 For Synthetic Particle Simulations

- Corr. (LSTM): The baseline method for edge prediction in [17].
- LSTM (single) / LSTM (joint): The baseline methods for state sequence prediction in [17].
- NRI (static): The NRI model with static latent graph [17].
- NRI (dynamic): The NRI model with latent graph re-evaluation at each time step [17].

8.2.2 For Benchmark Datasets

- Social-LSTM [1]: The model encodes the trajectories with an LSTM layer whose hidden states serve as the input of a social pooling layer.
- Social-GAN [6]: The model introduces generative adversarial learning scheme into S-LSTM to improve performance.
- Social-Attention [41]: The model deals with spatio-temporal graphs with recurrent neural networks, which is based on the architecture of Structural-RNN [14].
- Gated-RN [4]: The model infers relational behavior between road users and the surrounding environment by extracting spatio-temporal features.
- Trajectron++ [34]: The approach represents a scene as a directed spatio-temporal graph and extract features related to the interaction. The whole framework is based on conditional variational auto-encoder.
- NRI [17]: The model is formulated as a variational inference task with an encoder-decoder structure. This is the most related work.
- STGAT [13]: The model is a variant of graph attention network, which is applied to spatio-temporal graphs.
- Social-STGCNN [28]: The model is a variant of graph convolutional neural network, which is applied to spatio-temporal graphs.

8.2.3 Ablative Baselines

- SG (same node type): This is the simplest model setting, where only a static interaction graph is extracted based on the history information. The same node embedding function is shared among all the agent nodes.
- SG: This setting is similar to the previous one, except that different node embedding functions are applied to different types of agent nodes.

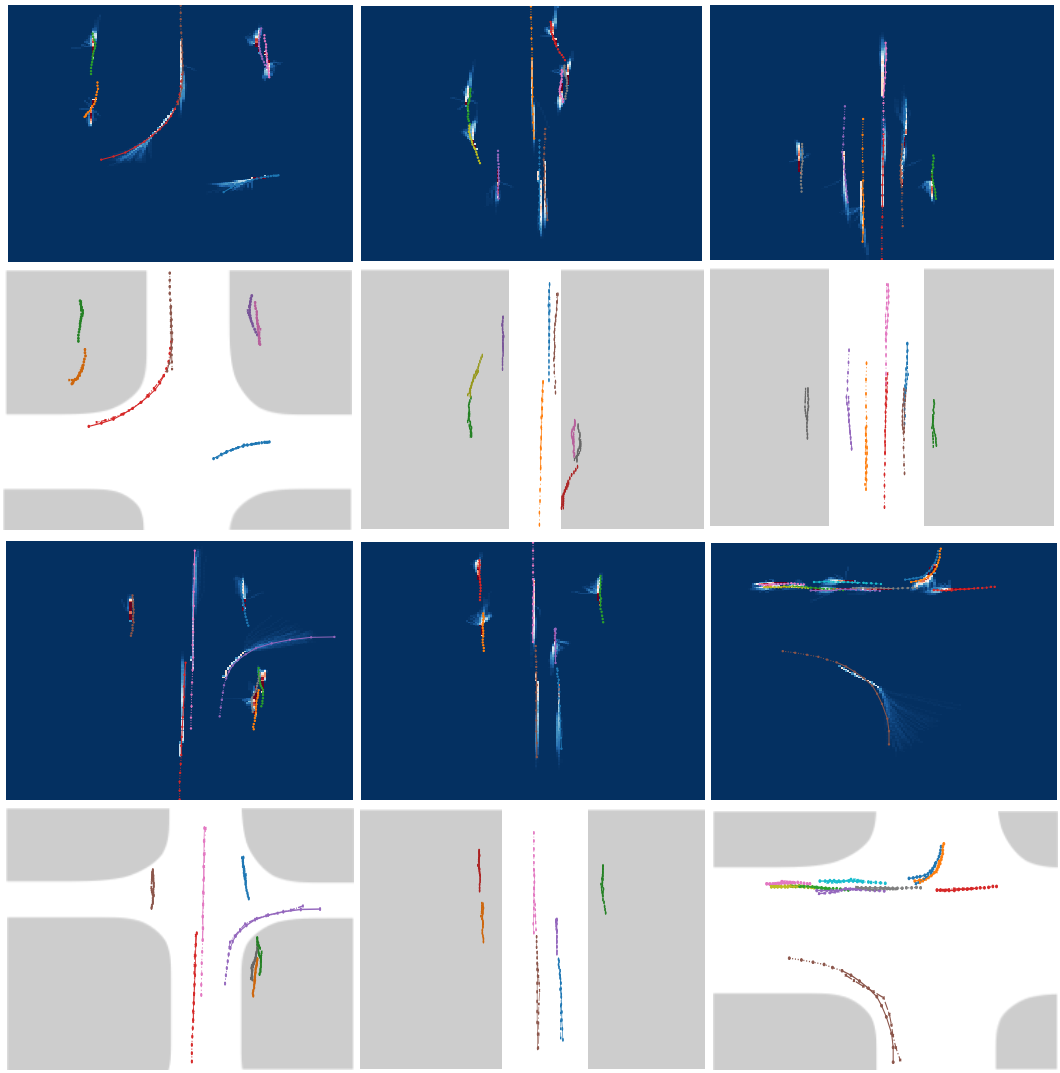

Figure 6: Additional qualitative results of the H3D dataset. The upper figures are the visualization of predicted distributions, and the lower figures are the best prediction hypotheses. The white areas are drivable areas and gray areas are sidewalks. Note that there are agents moving on the sidewalks since we also include pedestrians and cyclists.

- RNN re-encoding: The interaction graph is re-encoded every τ time steps using an RNN encoding process. Note that this is different from our model, since the RNN encoding process only captures evolution of node attributes without explicitly modeling the dependency of consecutive underlying interaction graphs.
- DG (single stage): This is our whole model, where the encoding, decoding functions and the graph evolving GRU are all trained together from scratch.
- DG (double stage): This is our whole model with double stage interaction graph learning, where the encoding, decoding functions trained at the first stage are employed as an initialization in the second stage.

8.3 Implementation Details

For all the experiments, a batch size of 32 was used and the models were trained for up to 20 epochs during the static graph learning stage and up to 100 epochs during the dynamic graph learning stage

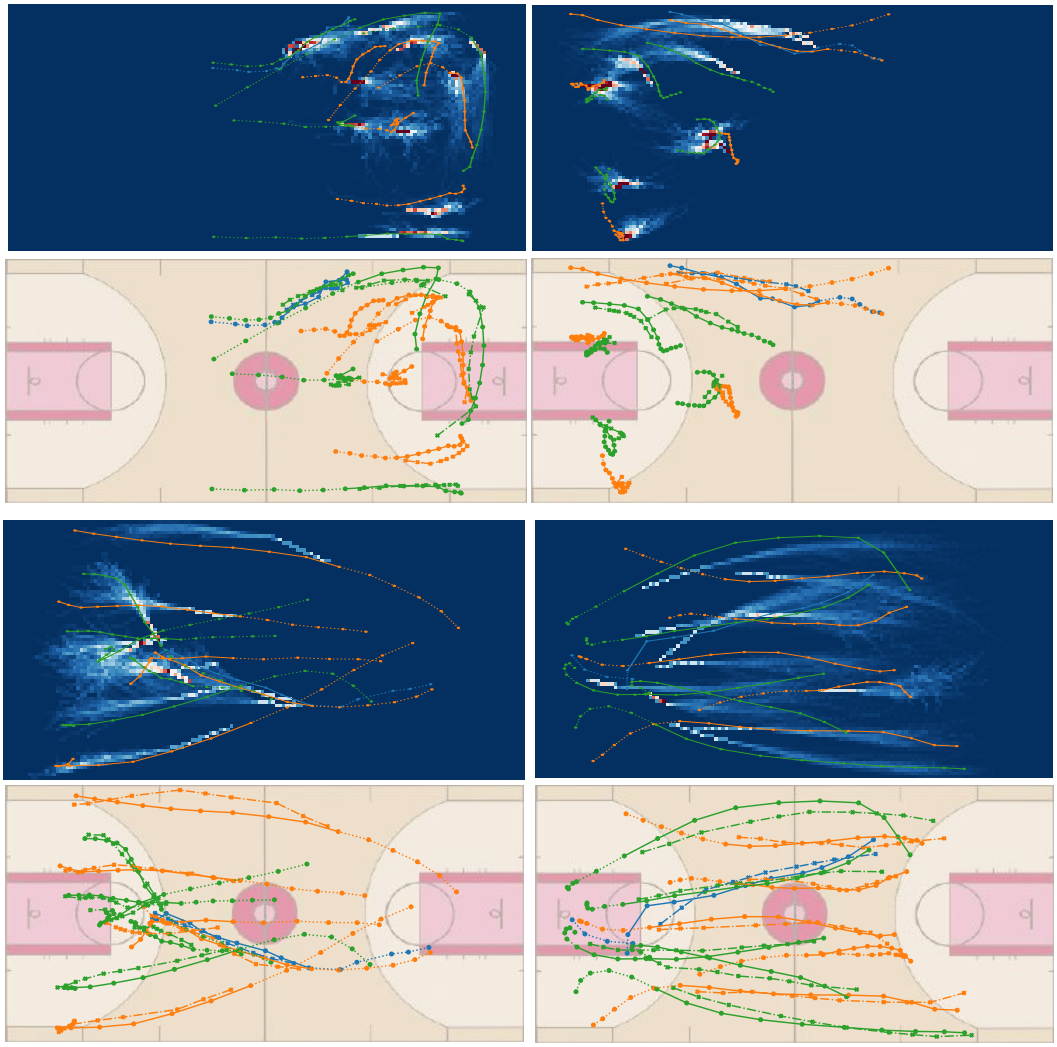

Figure 7: Additional qualitative results of the NBA dataset. The upper figures are the visualization of predicted distributions, and the lower figures are the best prediction hypotheses. The line colors indicate teams and blue lines are the trajectories of basketball.

with early stopping. We used Adam optimizer with an initial learning rate of 0.001. The models were trained on a single TITAN Xp GPU. We used a split of 65%, 10%, 25% as training, validation and testing data.

Specific details of model components are introduced below:

- *Agent node embedding function*: for each different node type, a distinct two-layer gated recurrent unit (GRU) with hidden size = 128.
- *Context node embedding function*: four-layer convolutional blocks with kernel size = 5 and padding = 3. The structure is [[Conv, ReLU, Conv, ReLU, Pool], [Conv, ReLU, Conv, ReLU, Pool]].
- *Agent node update function*: a three-layer MLP with hidden size = 128.
- *Edge update function*: for both agent-agent edges and agent-context edges, a distinct three-layer MLP with hidden size = 128.
- *Encoding function*: a three-layer MLP with hidden size = 128.
- *Decoding function*: a two-layer gated recurrent unit (GRU) with hidden size = 128.

- *Recurrent graph evolution module*: a two-layer GRU with hidden size = 256.

Specific experimental details of different datasets are introduced below:

- *Synthetic simulations*: 2 edge types, re-encoding gap = 1, encoding horizon = 20.
- *H3D dataset*: 5 edge types, re-encoding gap = 5, encoding horizon = 5.
- *NBA dataset*: 6 edge types, re-encoding gap = 4, encoding horizon = 5.
- *SDD dataset*: 4 edge types, re-encoding gap = 5, encoding horizon = 5.

8.4 Loss Function and Training

In our experiments, we first train the encoding / decoding functions using a static interaction graph. Then in the process of training dynamic interaction graph, we use the pre-trained encoding / decoding functions at the first stage to initialize the parameters of the modules used in the dynamic training. This step is reasonable since the encoding / decoding functions used in these two training process play similar roles and their optima are supposed to be close. And if we train dynamic graphs directly, it will lead to longer convergence time and is likely to be trapped into some bad local optima due to large number of learnable parameters. It is possible that this method may accelerate the whole training process and avoid some bad local optima.

In the training process, our loss function is defined as follows:

$$\mathcal{L}_S = -\mathbb{E}_{q(\mathbf{z}|\mathbf{X}_{1:T_h},\mathbf{C})}\left[\sum_{i=1}^{N}\sum_{t=T_h+1}^{T_h+T_f}\sum_{k=1}^{K} w_t^{i,k}\log p_t^{i,k}(\mathbf{x}_t|\mathbf{z},\mathbf{X}_{1:T_h},\hat{\mathbf{X}}_{T_h+1:t-1},\mathbf{C})\right] \text{ (Static)}, \tag{14}$$

$$\mathcal{L}_D = -\mathbb{E}_{q(\mathbf{z}'_{\beta(t)}|\mathbf{X}_{1+\beta\tau:T_h+\beta\tau},\mathbf{C})}\left[\sum_{i=1}^{N}\sum_{t=T_h+1}^{T_h+T_f}\sum_{k=1}^{K} w_t^{i,k}\log p_t^{i,k}(\mathbf{x}_t|\mathbf{z}'_{\beta(t)},\mathbf{X}_{1:T_h},\hat{\mathbf{X}}_{T_h+1:t-1},\mathbf{C})\right] \text{ (Dynamic)}, \tag{15}$$

where $q(\cdot)$ denotes the encoding and re-encoding operations, which return a factorized distribution of $\mathbf{z}_{ij}$ or $\mathbf{z}'_{ij}$. The $p_t^{i,k}(\mathbf{x}_t|\mathbf{z},\mathbf{X}_{1:T_h},\hat{\mathbf{X}}_{T_h+1:t-1},\mathbf{C})$ and $p_t^{i,k}(\mathbf{x}_t|\mathbf{z}'_{\beta(t)},\mathbf{X}_{1:T_h},\hat{\mathbf{X}}_{T_h+1:t-1},\mathbf{C})$ denote a certain Gaussian distribution.

8.5 Illustrative Diagram of the Decoding Process

We provide an illustrative diagram of the decoding process, which is shown in Figure 8. In this figure, without loss of generality we demonstrate the decoding process for only one node in a five-node observation graph to illustrate how the decoding process works. Figure 8(a) shows the observation graph, we choose the node on the right as an example. Figure 8(b) shows the process of using MLPs to process a specific edge, where $z_{ij,l}, l = 1, 2, ..., L$ denotes the probability of the edge belonging to a certain edge type l. The processed edges are shown in red. Figure 8(c) shows the sum over every incoming edge attribute of this node. Then we input the result into the decoding GRU. The decoding GRU outputs several Gaussian components and their corresponding weights. We sample one specific Gaussian component based on the weights. Then we use the $\boldsymbol{\mu}$ of the sampled Gaussian distribution as the output state at this step. $\boldsymbol{\mu}$ is used as the input into the next decoding step (if it's not the burn-in step). We iterate the decoding process several times until the desired prediction horizon is reached.

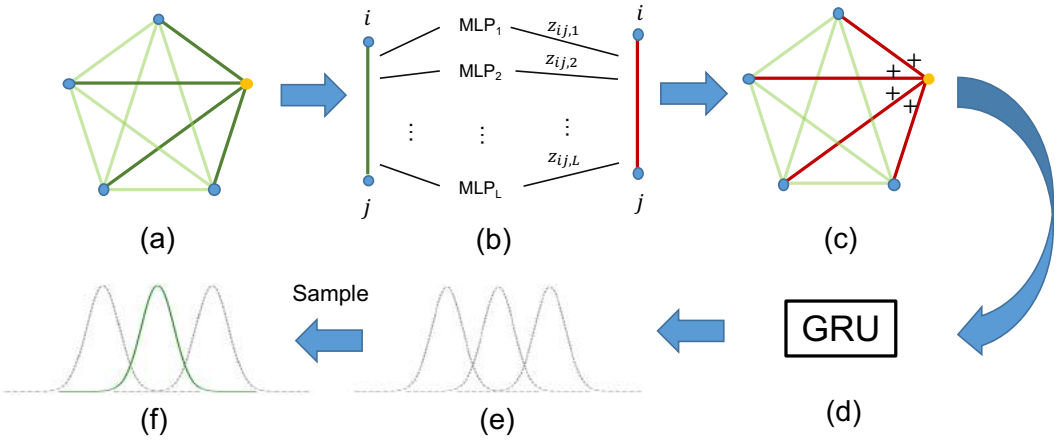

Figure 8: An illustrative diagram of the decoding process.