Heterogeneous Interaction Modeling With Reduced Accumulated Error for Multiagent Trajectory Prediction

Siyuan Chen and Jiahai Wang, *Senior Member, IEEE*

***Abstract*—Dynamical complex systems composed of interactive heterogeneous agents are prevalent in the world, including urban traffic systems and social networks. Modeling the interactions among agents is the key to understanding and predicting the dynamics of the complex system, e.g., predicting the trajectories of traffic participants in the city. Compared with interaction modeling in homogeneous systems such as pedestrians in a crowded scene, heterogeneous interaction modeling is less explored. Worse still, the error accumulation problem becomes more severe since the interactions are more complex. To tackle the two problems, this article proposes heterogeneous interaction modeling with reduced accumulated error (HIMRAE) for multiagent trajectory prediction. Based on the historical trajectories, our method infers the dynamic interaction graphs among agents, featured by directed interacting relations and interacting effects. A heterogeneous attention mechanism (HAM) is defined on the interaction graphs for aggregating the influence from heterogeneous neighbors to the target agent. To alleviate the error accumulation problem, this article analyzes the error sources from the spatial and temporal perspectives, and proposes to introduce the graph entropy and the mixup training strategy for reducing the two types of errors, respectively. Our method is examined on three real-world datasets containing heterogeneous agents, and the experimental results validate the superiority of our method.**

***Index Terms*—Error accumulation, graph entropy, heterogeneous interaction modeling, mixup training strategy, multiagent trajectory prediction.**

I. Introduction

MANY real-world complex systems, including social networks and urban traffic systems, can be regarded as dynamical systems composed of heterogeneous interacting agents. Different types of agents present various intrinsic behavior patterns, and the dynamic interactions among agents are even more complex, leading to complicated dynamics at both the individual level and the system level. Understanding

Manuscript received 18 January 2022; revised 19 August 2022; accepted 11 November 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0101203, in part by the National Natural Science Foundation of China under Grant 62072483, and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2022A1515011690 and Grant 2021A1515012298. *(Corresponding author: Jiahai Wang.)*

The authors are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China (e-mail: chensy47@mail2.sysu.edu.cn; wangjiah@mail.sysu.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2022.3224007.

Digital Object Identifier 10.1109/TNNLS.2022.3224007

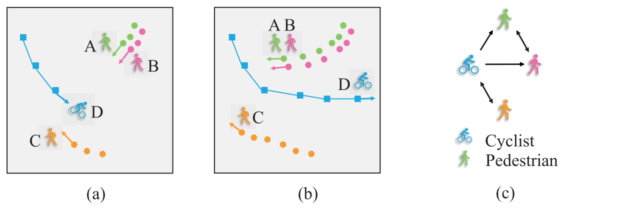


Fig. 1. Heterogeneous dynamical system involving three pedestrians (A, B, and C) and one cyclist (D). Two snapshots of the trajectories are taken at time (a) $t = 3$ and (b) $t = 6$. (c) Possible directed interaction graph among all traffic participants is provided.

the interactions among heterogeneous agents can help us predict and control the behavior of a system [1], [2], [3]. A typical application is trajectory prediction, which is a fundamental task in autonomous driving and mobile robot navigation.

Compared with *homogeneous* systems, the main difficulty of *heterogeneous* multiagent trajectory prediction lies in the heterogeneity of agents' interactions [4], [5]. Inaccurate interaction modeling at each time step, in turn, leads to a more severe error accumulation problem in multistep prediction.

The first problem, heterogeneity of agents' interactions, is that the interacting patterns are diversified among heterogeneous agents. This phenomenon is less explored in pedestrian trajectory prediction [6], where researchers focus on *homogeneous* interaction modeling. Usually, researchers assumed that distance is a consistent indicator of interactions. They built an interaction graph based on the pairwise distances, and then apply social pooling [7], attention mechanisms [8], or graph neural networks (GNNs) [9] to capture the spatial dependence among agents induced by the interaction graph. However, the distance may not be an accurate indicator of interactions [10], [11], especially in *heterogeneous* systems. A motivating example involving three pedestrians and one cyclist is provided in Fig. 1 to illustrate this problem.

This article gains three observations from Fig. 1 as follows. First, at time $t = 3$, Pedestrian A and B walk in a group, and the distance between them suggests a strong interaction. Pedestrian C is distant from A and B, indicating fairly weak interactions. Second, although the distance between Cyclist D and Pedestrian C is larger than that between Pedestrian A and B, their interaction is strong as they tend to avoid a collision. Third, Pedestrian A and B change their walking directions in advance as they try to avoid meeting Cyclist D. By contrast, Cyclist D is less affected by the two pedestrians. To sum up, the distance may not be a consistent indicator of interactions in *heterogeneous* systems. Besides, as a symmetric metric, distance can fail to capture asymmetric interactions.

Therefore, for *heterogeneous* interaction modeling, it is more preferable to learn a latent interaction graph [5] other than using a predefined graph. Furthermore, the information aggregation over the graphs should be able to distinguish interacting effects between heterogeneous agents. Li et al. [5] proposed to learn multirelational graphs, but the number of relation types is hard to decide in practice. Ivanovic and Pavone [4] defined an edge type as the concatenation of node types, but the corresponding category-aware modules suffered from quadratic space complexity. Thus, it remains to explore a more appropriate construction of interaction graphs and a lightweight design for heterogeneous information aggregation over the graphs.

The second problem, error accumulation, is an inevitable issue even in homogeneous systems. A wide class of models makes multistep predictions recursively, i.e., predicting an agent's next position based on its previous predictions. In these models, errors are aggregated from neighboring agents and accumulate over multiple time steps, leading to a possible large deviation in the long-term prediction. Nevertheless, to the best of our knowledge, few efforts have been paid to this issue in the field of trajectory prediction.

Error accumulation can be effectively eliminated in the training stage by feeding the true positions of agents to the model, i.e., teacher forcing [12], but this will give rise to a large discrepancy between testing and training. Such phenomenon is well known as the exposure bias for autoregressive language models in natural language generation [13]. Zhang et al. [14] proposed to select oracle words that simulate the ground-truth words, and randomly switched between the oracle words and the ground truth during the training procedure. Their method proved to be effective in narrowing the gap between training and inference. This inspires us to mix up the true positions and some high-quality candidates as the input of models for reducing the accumulated error.

To deal with the heterogeneity of interactions and the error accumulation issue, this article proposes **H**eterogeneous **I**nteraction **M**odeling with **R**educed **A**ccumulated **E**rror (HIMRAE) for multiagent trajectory prediction. Our method adopts an encoder–decoder framework. The encoder dynamically infers two types of information from the historical trajectories, the directed interacting relations that tell if one agent affects another, and the interacting effects that implicitly describe how the interaction works. The interacting relations and the interacting effects jointly define an edge-featured interaction graph. The decoder defines a heterogeneous attention mechanism (HAM) over the interaction graph to model the interactions among heterogeneous agents, and finally, it predicts future trajectories using category-aware recurrent neural networks.

Regarding the recursive multistep prediction in multiagent systems as the forward pass of a multilayer GNN, this article analyzes the error accumulation problem from the spatial and temporal aspects. To reduce the spatially aggregated errors, the graph entropy is introduced for penalizing the complexity of the interaction graphs. The minimizer of graph entropy is characterized in theory to guide the choice of regularization coefficient. For the temporally accumulated errors, this article proposes to mix up the ground-truth trajectories and the predicted ones in the training stage, which balances the model's ability in accurate single-step prediction and multistep prediction. The mixup training strategy is proven to enjoy a lower error bound in theory.

The contributions of this article are summarized as follows.

1) An encoder–decoder framework is proposed for heterogeneous multiagent trajectory prediction. Our method dynamically infers the interacting relations and the interacting effects among agents, characterizing the interaction structure as an edge-featured graph.
2) This article proposes an HAM of linear space complexity over the interaction graphs to model the complex heterogeneous interactions.
3) This article analyzes the error accumulation problem in multiagent trajectory prediction from both the spatial and temporal perspectives, and proposes to use the graph entropy and a mixup training strategy to reduce the two types of errors, respectively. Both the graph entropy and the mixup training strategy have theoretical justifications under some simplified assumptions.

II. Related Works

A. Interaction Modeling in Multiagent Trajectory Prediction

Traffic participants, like pedestrians and vehicles, interact with others in the surroundings and adjust their trajectories toward the destinations. The interaction is complex itself, and evolves over time, leading to the complicated trajectories of multiple traffic participants. Interacting modeling for multiagent systems generally includes interaction graph construction and information aggregation over the graph. The techniques differ in *homogeneous* and *heterogeneous* systems.

In *homogeneous* systems, grids [7], complete graphs [15], [16], and distance-based graphs [8], [9] were used as predefined interaction graphs. Since complete graphs and distance-based graphs contain superfluous edges and fail to model directed effects, some works [10], [11], [17] proposed to learn asymmetric latent graphs. Given the interaction graphs, social pooling [7], spatio-temporal graph convolutions, spatio-temporal graph attention [8], [15] were introduced to aggregate the interacting effects. Prior knowledge like sparsity [10] and obstacle avoidance [16] were used for attention computation.

In the *heterogeneous* systems, the interaction graphs can be extended to heterogeneous graphs with different types of nodes and edges. The edge types can be predefined as the concatenation of node types [4], [18], e.g., "Pedestrian-Bus," or learned by the model from observational data [5]. Relative position, velocity, and heading angle can be further incorporated as edge features [19], [20].

Given the heterogeneous interaction graphs, node-type aware modules were used for trajectory encoding, type-specific supernodes were used to share information among agents of the same type [21], [22], and edge-type aware modules were used for modeling interacting effects, which are aggregated to predict future states [4], [18]. However, there are some drawbacks of edge-type aware aggregations. When the edge type is defined by concatenating node types [4], [18], the number of edge-type aware modules grows quadratically, which can be a redundant design. When the edge type is learned by the model [5], the number of edge types is hard to decide in practice. Besides, some works [18], [21] limited themselves to a predecided set of agent types when coding the category-aware modules, which is less flexible when agent types are different in new datasets.

Unlike existing works, this article proposes to learn edge-featured graphs, where an edge indicates a directed interaction, and an edge feature describes the interacting effects without defining edge types. Besides, an HAM of linear space complexity is proposed to aggregate the interacting effects.

B. Heterogeneous Graph Neural Networks

Graphs with more than one type of nodes or edges, termed as heterogeneous graphs or heterogeneous information networks [23], [24], widely exist in the world, such as social networks, knowledge graphs, and proteins. GNNs developed for this form of data are called heterogeneous GNNs. The message passing mechanisms of homogeneous GNNs [25] were substantially extended by introducing category-aware modules for different types of nodes [26], edges [27], and their compositions, meta-paths [28]. Representative works include relation-type aware aggregators [27], heterogeneous graph transformer [29], hierarchical aggregators defined over node types [26], and hierarchical attention defined over both node-level and semantic-level [28].

Despite the great success of heterogeneous GNNs in heterogeneous graphs, these methods are not applicable to heterogeneous multiagent trajectory prediction. The key difference is that there are no natural well-defined relations between two agents. Even when the categories of the agents and the relative position between them are known, the interactions can be variable. Besides, these methods are not designed to integrate possible edge features. Our method can infer the relations between agents and model their interactions over an edge-featured graph.

C. Learning Graphs From Data

Graphs provide a structured representation of data. When the graph representation is not readily available, researchers seeked to learn it from observational data [30], with application to multivariate time series forecasting [31] and reasoning over physical systems [17]. Edge directions [17], [31], edge types [17], and coexistence of edges [32] were considered for graph learning. The learning is often guided by a task-specific objective, while the complexity of the graph is not explicitly controlled. Sparsity [17] is an intuitive choice to penalize the graph complexity. However, a simpler graph is not necessarily sparser, and a sparser graph may not necessarily lead to better performance. Instead, this article introduces another metric, the graph entropy, to reduce the graph complexity, which helps reduce the prediction error.

III. Method: HIMRAE

A. Problem Definition

In an N-agent heterogeneous dynamical system, each agent i belongs to a category c_i out of C categories. The number of agents may vary case by case. The 2-D coordinate of agent i at time t is recorded as $\mathbf{x}_i^t = (x_i^t, y_i^t)$. Let T_h and T_f denote the number of historical steps and future steps, respectively. The problem of trajectory prediction is that, given the historical trajectories of all agents $\mathbf{X}^{1:T_h} = \{\mathbf{x}_i^{1:T_h}\}_{i=1}^N$, predict their future trajectories $\widehat{\mathbf{X}}^{T_h+1:T_h+T_f}$ that best match the ground-truth trajectories $\mathbf{X}^{T_h+1:T_h+T_f}$. A summary of the notations used in this article can be found in *Appendix A*.

B. Hypotheses, Motivation and Overview of HIMRAE

The trajectory of each agent is affected by the agents it interacts with, and the interactions vary over time. The interactions among agents at time t can be abstracted as a directed graph $\mathcal{G}^t = (\mathcal{V}, \mathcal{E}^t)$, with each agent $i \in \mathcal{V}$ as a node and the interacting relation from agent i to agent j as an directed edge $(i, j) \in \mathcal{E}^t$. Generally, the underlying interaction graphs are unobservable, but they can be inferred from the historical trajectories and used for future trajectory prediction.

This article hypothesizes that the interacting relations among agents within a small time window are relatively stable. Therefore, a θ-parameterized distribution of the full trajectories can be factorized as

$$p_\theta\left(\mathbf{X}^{1:T_h+T_f}\right) = p_\theta\left(\mathbf{X}^{1:\tau}\right)\prod_{m=1}^{M} p_\theta\left(\mathbf{X}^{1+m\tau:(m+1)\tau}\middle|\mathbf{X}^{1:m\tau}\right) \times p_\theta\left(\mathbf{X}^{1+M\tau:T_h+T_f}\middle|\mathbf{X}^{1:M\tau}\right) \quad (1)$$

where τ is the window size, and $M = \lfloor (T_h + T_f)/\tau \rfloor$ is the maximum number of the time windows. An appropriate τ may be related to the number of agents N, since the interactions change more frequently when N is large. Nonetheless, τ is assumed to be case agnostic in this article for simplicity, and a more sophisticated design is left for future work.

To motivate the design of the interaction graph, the authors observe that: in practice, one has little prior knowledge about the semantics of the interaction between two agents, e.g., the type of the interaction and its effect, even when the agent types and their relative position are known. Therefore, this article seeks to infer the existence of interactions and a hidden state that implicitly encodes the interacting effects. Formally, given the trajectories in the previous time window, one can infer the interacting relations and interacting effects among agents, which lead to the following factorization:

$$\begin{aligned} &p_\theta\left(\mathbf{X}^{1+m\tau:(m+1)\tau}\middle|\mathbf{X}^{1:m\tau}\right) \\ &= \int p_\theta\left(\mathbf{X}^{1+m\tau:(m+1)\tau}\middle|\mathbf{X}^{1:m\tau}, \mathbf{Z}^m, \mathbf{E}^m\right) \\ &\quad \times p_\theta\left(\mathbf{Z}^m, \mathbf{E}^m\middle|\mathbf{X}^{1:m\tau}\right) \mathrm{d}\mathbf{Z}^m\, \mathrm{d}\mathbf{E}^m \end{aligned} \quad (2)$$

where $\mathbf{Z}^m$ and $\mathbf{E}^m$ are the interacting relations and interacting effects in the mth time window, respectively. $\mathbf{Z}^m \in \{0, 1\}^{N\times N}$ is a zero-diagonal binary matrix with $z_{ij}^m = 1$ indicating

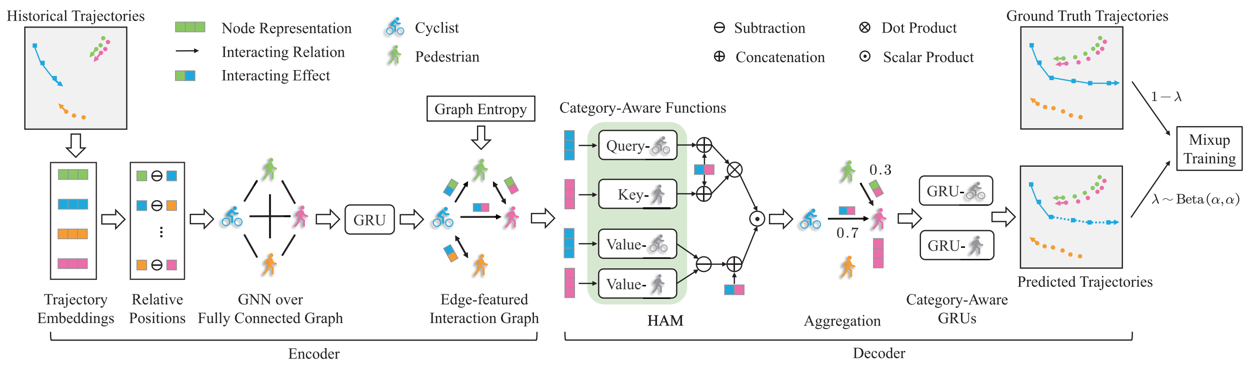


Fig. 2. Overview of HIMRAE. HIMRAE takes the form of an autoencoder. The encoder applies a GNN with a GRU over the historical trajectories to infer dynamic interaction graphs. The interaction graph is an edge-featured graph, where an edge represents a directed interacting relation and an edge feature represents the interacting effect. The graph entropy is used to control the structural complexity of the interaction graph. The decoder defines an HAM to model the interactions among heterogeneous agents, and predicts future trajectories recursively. The predicted trajectories and the ground-truth trajectories are mixed with a coefficient λ sampled from a beta distribution $\mathrm{Beta}(\alpha, \alpha)$ to reduce the temporally accumulated error.

an directed edge (i, j). $\mathbf{E}^m \in \mathbb{R}^{N\times N\times D}$ is a real-valued tensor with $\mathbf{e}_{ij}^m$ representing the impact of agent i on agent j. Regarding $\mathbf{E}^m$ as the edge features, $\mathbf{Z}^m$ and $\mathbf{E}^m$ jointly define an edge-featured interaction graph.

Directly evaluating $p_\theta(\mathbf{Z}^m, \mathbf{E}^m|\mathbf{X}^{1:m\tau})$ in intractable [33], therefore, a neural network $q_\phi(\mathbf{Z}^m, \mathbf{E}^m|\mathbf{X}^{1:m\tau})$ parameterized by ϕ is used as an approximation. Interpreting q_ϕ and p_θ as an encoder and a decoder, respectively, our method falls in the encoder–decoder framework.

The predicted trajectories $\widehat{\mathbf{X}}$ are assumed to obey an isotropic Gaussian distribution. Therefore, maximizing the log-likelihood conditioned on the historical trajectories leads to minimizing the expected squared loss

$$\mathcal{L} = \mathop{\mathbb{E}}_{\mathbf{Z}^{1:M},\mathbf{E}^{1:M}|\mathbf{X}^{1:T_h}} \left[\frac{1}{NT_f}\sum_{t=T_h+1}^{T_h+T_f}\left\|\mathbf{X}^t - \widehat{\mathbf{X}}^t\right\|_F^2\right]. \tag{3}$$

Though intuitive, the formulation above raises two issues: 1) unlike the variational autoencoder (VAE) [33], it lacks a natural choice for constraining the latent variables, as there is little prior knowledge about the interaction graph in practice and 2) directly optimizing $\mathcal{L}$ will face a severe problem of error accumulation from the recursive nature of the decoder. Since a simplified interaction graph can be more controllable, this article adopts the graph entropy to penalize the structural complexity of the graph, which also alleviates the error propagation problem in the spatial domain. For the temporally accumulated errors, this article introduces a mixup training strategy.

An overview of the proposed method is shown in Fig. 2. The details of the encoder and decoder are described in Sections III-C and III-D, respectively. Section III-E analyzes the error sources from the spatial and temporal domains, and describes the usage of the graph entropy. Section III-F describes the mixup training strategy. Section III-G summarizes the characteristics of our method.

C. Encoder

The encoder aims at inferring the dynamically evolving interaction graph in each time window. Since the underlying graph is unknown, a GNN is applied over a fully connected graph to learn the latent interaction structure, and a gated recurrent unit (GRU) [34] is used to capture the temporal dependence of the latent graphs. The structure of the encoder is visualized in Fig. 2 (left).

First, the historical trajectories are embedded to a latent space to obtain the node representation of each agent

$$\mathbf{v}_j^m = f_{\mathrm{emb}}\left(\mathbf{x}_j^{1+(m-1)\tau:m\tau}\right) \tag{4}$$

where f_{emb} is a multilayer perceptron (MLP). Then, one can use the relative position $\mathbf{v}_i^m - \mathbf{v}_j^m$ between two agents in the latent space to describe their spatial relationship. $\mathbf{v}_i^m - \mathbf{v}_j^m$ measures the correlation between the trajectories of two agents, which can be a good indicator of their interaction. This is advantageous over the stepwise relative position in the original space [9], a local metric that may fail to capture the global correlation.

The relative position can be viewed as a message over an edge. Following the message passing formulation of GNNs [25], a two-layer GNN is designed as follows:

$$\widetilde{\mathbf{v}}_j^m = f_v\left(\sum_{i\neq j} f_e\left(\mathbf{v}_i^m - \mathbf{v}_j^m\right)\right) \tag{5}$$

$$\widetilde{\mathbf{e}}_{ij}^m = \widetilde{f}_e\left(\widetilde{\mathbf{v}}_i^m - \widetilde{\mathbf{v}}_j^m\right) \tag{6}$$

where f_v is an MLP updating the node embeddings, f_e and $\widetilde{f}_e$ are MLPs that update the edge embeddings. Considering the uncertainty of the interacting effect, $\mathbf{e}_{ij}^m$ is sampled from a Gaussian distribution $\mathcal{N}(\widetilde{\mathbf{e}}_{ij}^m, \mathbf{I})$.

As for the interacting relation, a GRU is used to model its evolution over time, i.e.,

$$\mathbf{r}_{ij}^m = \mathrm{GRU}\left(\widetilde{\mathbf{e}}_{ij}^m, \mathbf{r}_{ij}^{m-1}\right) \tag{7}$$

where $\mathbf{r}_{ij}^m$ is an edge representation encoding the dynamics of the interacting relation. By projecting $\mathbf{r}_{ij}^m$ to a scalar, one can calculate the interacting probability from one agent to another. However, sampling the interacting relation is nondifferentiable since z_{ij}^m is a discrete variable. Fortunately, the Bernoulli distribution has a continuous approximation named the binary concrete distribution [35] that allows back-propagation. Formally, z_{ij}^m is sampled via the following reparameterization trick:

$$z_{ij}^m = \mathrm{Sigmoid}\left(\left(f_{\mathrm{proj}}\left(\mathbf{r}_{ij}^m\right) + \ln\delta - \ln(1-\delta)\right)\Big/ T\right) \quad (8)$$

where $\delta \in \mathbb{R}$ is drawn from the Gumbel(0, 1) distribution and T is a temperature parameter that controls the "smoothness" of the samples. f_{proj} maps $\mathbf{r}_{ij}^m$ to a scalar.

D. Decoder

The decoder is intended for modeling the interactions among heterogeneous agents based on the interaction graphs, and predicting their future trajectories. An HAM is used to capture spatial dependence among heterogeneous agents, and category-aware GRUs are used to capture agents' intrinsic dynamics. The procedure of the decoder is visualized in Fig. 2 (right).

To distinguish the importance of different types of agents, this article extends the scaled dot-product attention in Transformer [36] by introducing category-aware modules. The attention mechanism is defined as mapping a query and a set of key-value pairs to an output. Given an edge (i, j), this article treats the source node i as a "query" and the target node j as a "key" to compute the attention score. The relative position concatenated with the interacting effect is regarded as the "value" corresponding to the key. The sum of the values from adjacent agents weighted by the attention scores represents the overall influence of interactions. To model the effect of heterogeneous agents, this article simply applies category-aware mappings to the agent representations before the aforementioned computation.

Let $\mathbf{h}_j^t$ be the hidden vector encoding the dynamics of agent j at time t in the decoder. An HAM is formulated as follows:

$$\mathbf{m}_j^t = \sum_{i\neq j} \alpha_{ij}^t \cdot f_V\left(\left[g_V^{c_i}\left(\mathbf{h}_i^t\right) - g_V^{c_j}\left(\mathbf{h}_j^t\right), \mathbf{e}_{ij}^m\right]\right) \quad (9)$$

where $\mathbf{m}_j^t$ is the aggregated interacting effects from agent j' s neighbors, and $[\cdot,\cdot]$ is the concatenation operator. f_V is an MLP updating the value vector and $g_V^{c_i}$ is a category-aware single-layer perceptron that maps different types of agents to a common space. α_{ij}^t is the attention score defined in a softmax-like form

$$\alpha_{ij}^t = \frac{z_{ij}^m \cdot \exp\left(a_{ij}^t\right)}{\sum_{z_{ij}^m > 1/2} \quad z_{ij}^m \cdot \exp\left(a_{ij}^t\right)}. \quad (10)$$

In the training stage, z_{ij}^m sampled via (8) ranges in [0, 1], and only edges with $z_{ij}^m > 1/2$ are involved in the computation. In the testing stage, (10) is the normal graph attention over the inferred edges without ambiguity.

In (10), a_{ij}^t is calculated via the scaled inner product

$$a_{ij}^t = \frac{1}{\sqrt{D}} f_Q\left(\left[g_Q^{c_i}(\mathbf{h}_i^t), \mathbf{e}_{ij}^m\right]\right)^T f_K\left(\left[g_K^{c_j}(\mathbf{h}_j^t), \mathbf{e}_{ij}^m\right]\right). \quad (11)$$

f_Q and f_K are single-layer perceptrons for updating the query vector and the key vector, respectively. $g_Q^{c_i}$ and $g_K^{c_i}$ are category-aware mappings like $g_V^{c_i}$. D is the dimension of the query vector. In (9)–(11), $\mathbf{e}_{ij}^m$ implicitly decides the interacting effect without explicitly predefining a finite set of interacting relations. HAM can also be extended to incorporate subcategories for modeling finer-grained or personalized trajectory patterns. The subcategories can be learned by clustering [37] or contrastive learning [38] that encourages discriminative representations, which are left for future works.

Note that HAM is a natural extension of the self-attention in transformer to an edge-featured graph containing multiple types of nodes. By dropping the category-aware mappings $g_{Q/K/V}^{c_i}$, HAM reduces to a homogeneous attention mechanism. By further ignoring the edge feature $\mathbf{e}_{ij}^m$, the original self-attention is recovered. Besides, HAM is a lightweight design for heterogeneous interaction modeling since the number of category-aware modules is linear of the agent types.

After modeling the spatial dependence among agents in HAM, our method use category-aware GRUs [5] to capture the intrinsic behavior patterns for different types of agents

$$\mathbf{h}_j^{t+1} = \mathrm{GRU}_{c_j}\left(\left[\mathbf{m}_j^t, \widehat{\mathbf{x}}_j^t\right], \mathbf{h}_j^t\right). \quad (12)$$

The hidden vector $\mathbf{h}_j^{t+1}$ is used to predict future trajectories

$$\boldsymbol{\mu}_j^{t+1} = \widehat{\mathbf{x}}_j^t + f_{\mathrm{out}}\left(\mathbf{h}_j^{t+1} + \boldsymbol{\epsilon}\right) \quad (13)$$

where f_{out} is an MLP outputting the change in positions, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a noise vector to increase diversity. $\widehat{\mathbf{x}}_j^{t+1}$ is assumed to follow an isotropic Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_j^{t+1}, \sigma^2\mathbf{I})$ with σ^2 as a fixed variance. The mean $\boldsymbol{\mu}_j^{t+1}$ is used as the predicted positions. As is done in EvolveGraph [5], the predicted position $\widehat{\mathbf{x}}_j^t$ in (12) and (13) is replaced by the ground-truth $\mathbf{x}_j^t$ during the historical steps.

A recent work, reinforced hybrid attention inference network (RAIN) [39], shares a similar pipeline on latent graph learning with our method. RAIN first uses reinforcement learning to identify important relations that decrease the prediction error, and then assigns the relations different attention weights. Unlike RAIN, our method uses the reparameterization trick to allow back-propagation, offering a simpler option for end-to-end training without a specific design for reinforcement learning. Besides, the soft attention mechanism in RAIN is still homogeneous, while ours is heterogeneous.

E. Loss Function With Graph Entropy Regularization

The decoder defined in Section III-D makes multistep predictions recursively, which can be viewed as the forward pass of a multilayer GNN. Fey et al. [40] showed that under some mild assumptions, the errors of the node embeddings grow exponentially on the Lipschitz constants of the GNN model as well as the node degrees with respect to the number of layers. As shown in Fig. 3(a), this result can be intuitively understood in the trajectory prediction problem, since the errors from interacting agents will propagate spatially and accumulate over time. Particularly, the spatially propagated error is a unique challenge in spatio-temporal time series forecasting like trajectory prediction, which is not covered by traditional research focusing on temporally accumulated error.

The number of agents that affect a target agent can be termed as the in-degree of a node. The distribution of the in-degrees can be imbalanced within an interaction graph,

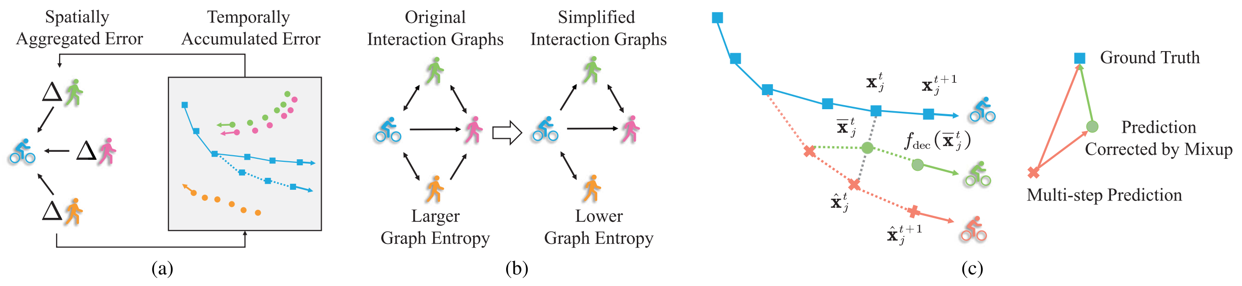

Spatially Aggregated Error
Temporally Accumulated Error
Original Interaction Graphs
Simplified Interaction Graphs
Larger Graph Entropy
Lower Graph Entropy
$\mathbf{x}_j^t$
$\mathbf{x}_j^{t+1}$
$\overline{\mathbf{x}}_j^t$
$f_{\text{dec}}(\overline{\mathbf{x}}_j^t)$
$\hat{\mathbf{x}}_j^t$
$\hat{\mathbf{x}}_j^{t+1}$
Ground Truth
Prediction Corrected by Mixup
Multi-step Prediction
(a)
(b)
(c)

Fig. 3. (a) Visualization of the error sources from both the spatial and temporal perspectives. (b) Lower graph entropy favors a simplified interaction graph. (c) Predicted position and the ground-truth position at time t are mixed up to correct the multistep prediction.

and varies over different graphs. Directly restricting the in-degrees may lead to a degenerate interaction graph that hurts the expressiveness of the model. Instead, this article turns to the graph entropy [41], a global measure for the complexity of graphs with wide applications in sociology, chemistry, etc. It also receives increasing interest in graph machine learning for deciding the node-embedding dimension [42] and designing graph-pooling modules [43].

The graph entropy is originally defined on undirected graphs [41] by introducing a node distribution. The definition can be naturally extended to our case for directed graphs by considering the in-degrees [41]. Let $d_j^m = \sum_{i=1}^N z_{ij}^m$ denote the in-degree of node j, and $|\mathcal{E}^m| = \sum_{i \neq j} z_{ij}^m$ denote the total number of edges. Then, a graph entropy is defined as follows:

$$\mathrm{H}_{\mathcal{G}}\left[\mathbf{Z}^m\right] = -\frac{1}{\ln N}\sum_{j=1}^{N}\left(d_j^m/|\mathcal{E}^m|\right)\ln\left(d_j^m/|\mathcal{E}^m|\right) \tag{14}$$

where $\ln N$ is a normalization constant for the entropy. The graph entropy weighted by a nonnegative coefficient γ is added to the original loss function to penalize the graph complexity

$$\mathcal{L}' = \mathcal{L} + \frac{\gamma}{M}\sum_{m=1}^{M}\mathrm{H}_{\mathcal{G}}\left[\mathbf{Z}^m\right]. \tag{15}$$

Since $\mathcal{L}$ represents the average prediction error of a single agent per step and $\mathrm{H}_{\mathcal{G}}[\mathbf{Z}^m]$ lies in $[0, 1]$, γ stands for the strength of graph complexity penalization to a single node. Equation (15) constrains the latent graphs from the probabilistic perspective, and is a meaningful extension of the entropy regularization to graphs, an important class of irregular data.

Ignoring the reconstruction error, the graph entropy is maximized when all in-degrees are equal. The characterization of the minimizer is trickier. For simplicity, it is informally stated as the following theorem.

Theorem 1 (Informal): Given the numbers of nodes and edges, the graph entropy is minimized when the edges are centered on a few nodes.

Obviously, the minimizer given by Theorem 1 can be rarely reached when considering the reconstruction error. This article derives a more practical corollary that helps us understand the optima.

Corollary 1: Given the number of nodes, a graph with fewer edges has a smaller lower bound of graph entropy.

The corollary points out that the graph entropy and the sparsity are different metrics for graph complexity. A graph with lower graph entropy is not necessarily sparser, while a sparser graph has a potentially lower graph entropy. An empirical study of graph entropy with sparsity constraints is presented in Section V-B. A formal statement of Theorem 1 and the proofs for Theorem 1 and Corollary 1 using the majorization theory [44] are shown in *Appendix B*.

F. Mixup Training Strategy

To reduce the temporally accumulated errors, this section introduces the mixup training strategy. Mixup is a simple and data-agnostic data augmentation trick [45]. It is originally proposed to improve the generalization of deep neural networks by training the models on the linear interpolation (convex combination) of two random training samples and their labels. Liang et al. [46] employed mixup to generalize from simulation datasets to real-world datasets in trajectory prediction. Apart from generalization, mixup can also be used to correct the predicted values. As shown in Fig. 3(c), mixing up the true position and the predicted position yields a more accurate prediction. This motivates us to reduce the accumulated error via mixup.

Given the true position $\mathbf{X}^t$ and its predicted value $\widehat{\mathbf{X}}^t$, a convex combination of them is defined as

$$\bar{\mathbf{X}}^t = \mathrm{Mix}_\lambda\left(\widehat{\mathbf{X}}^t, \mathbf{X}^t\right) \triangleq \lambda\widehat{\mathbf{X}}^t + (1-\lambda)\mathbf{X}^t \tag{16}$$

where the mixing coefficient $\lambda \in [0, 1]$ is sampled from a beta distribution $\mathrm{Beta}(\alpha, \alpha)$ parameterized by a positive number α.

$\bar{\mathbf{X}}^t$ can be regarded as a correction of the predicted position $\widehat{\mathbf{X}}^t$. Thus, $\bar{\mathbf{X}}^t$ can serve as a more accurate starting point for multistep prediction, which in turn helps to infer a more accurate interaction graph. Furthermore, by simplifying the notation of multistep predictions as $\widehat{\mathbf{X}}^{t+1} = f_{\text{dec}}(\widehat{\mathbf{X}}^t)$, this article conjectures that the gap between $\widehat{\mathbf{X}}^{t+1}$ and $f_{\text{dec}}(\bar{\mathbf{X}}^t)$ is smaller than that between $\widehat{\mathbf{X}}^{t+1}$ and $\mathbf{X}^{t+1}$, which is visualized in Fig. 3(c). Under this assumption, alternatively minimizing two intermediate objectives $\|f_{\text{dec}}(\bar{\mathbf{X}}^t) - \mathbf{X}^{t+1}\|_F^2$ and $\|f_{\text{dec}}(\bar{\mathbf{X}}^t) - \widehat{\mathbf{X}}^{t+1}\|_F^2$ can be easier than directly optimizing the original objective $\|\mathbf{X}^{t+1} - \widehat{\mathbf{X}}^{t+1}\|_F^2$.

However, correcting the prediction at each time step may prevent the model from learning to make multistep predictions. Without loss of generality, this article proposes to correct the final prediction of each time window. It can balance the model's ability in single-step and multistep prediction instead of focusing on single-step prediction like teacher forcing. Let $f_{\text{dec}}^n(\bar{\mathbf{X}}^t)$ denotes the nth step prediction from time t. Then,

Algorithm 1 Mixup Training Strategy

```
Input: decoder $f_{\text{dec}}$ with the parameter $\theta$,
       hyperparameter $\alpha$ for the beta distribution,
       optimization algorithm $A$.
Output: decoder $f_{\text{dec}}$.
1 Initialize $\theta$ with random weights;
  /* First update.                                  */
2 Compute $\widehat{\mathbf{X}}^t$;
3 Sample $\lambda \sim \text{Beta}(\alpha, \alpha)$;
4 Compute $\bar{\mathbf{X}}^t = \text{Mix}_\lambda(\text{StopGrad}(\widehat{\mathbf{X}}^t), \mathbf{X}^t)$;
5 Compute $\mathcal{L}_1 = \sum_{n=1}^{\tau} \|\mathbf{X}^{t+n} - f_{\text{dec}}^n(\bar{\mathbf{X}}^t)\|_F^2$;
6 $\theta := A(\mathcal{L}_1; \theta)$;
  /* Second update.                                 */
7 Compute
  $\mathcal{L}_2 = \sum_{n=1}^{\tau} \| f_{\text{dec}}^n(\widehat{\mathbf{X}}^t) - \text{StopGrad}(f_{\text{dec}}^n(\bar{\mathbf{X}}^t))\|_F^2$;
8 $\theta := A(\mathcal{L}_2; \theta)$;
9 return decoder $f_{dec}$.
```

the mixup training strategy can be briefly described in the decoding procedure within a time window. The corresponding pseudo code is shown in Algorithm 1.

StopGrad$(\cdot)$ in line 4 of Algorithm 1 is a function that treats $\widehat{\mathbf{X}}^t$ as a constant without back-propagation. The parameter α decays over the training epochs, resulting in a gradually flattened beta distribution [45] that allows more combinations of the predicted positions and the ground truth. This increases the difficulties in training and encourages the model to make more accurate multistep predictions. The graph entropy introduced in Section III-E can be added to the loss function to further reduce the accumulated error.

Note that the mixup training strategy is fundamentally different from Zhang et al.'s method [14]. Our method is designed for the continuous domain while theirs is limited to the discrete domain. As for the key techniques, the ground truth is combined with the predicted value via interpolation rather than random substitution. Moreover, an alternating optimization procedure is tailored for our problem instead of minimizing the original objective.

Finally, this article theoretically analyzes the effectiveness of the proposed optimization algorithm under some simplified assumptions [47], summarized in the following theorem.

Theorem 2: Let f be a learned model with bounded single-step prediction error $\|f(\mathbf{X}^t) - \mathbf{X}^{t+1}\| \le \epsilon$. Assume that f is Lipschitz continuous with constant L_f under the norm $\|\cdot\|$. Then, the following upper bounds hold.

1) $\|f^n(\widehat{\mathbf{X}}^t) - \mathbf{X}^{t+n}\| \le L_f^n \|\widehat{\mathbf{X}}^t - \mathbf{X}^t\| + (L_f^n - 1)/(L_f - 1)\epsilon$.
2) $\mathbb{E}_\lambda[\|f^n(\bar{\mathbf{X}}^t) - \mathbf{X}^{t+n}\|] \le (1/2)L_f^n \|\widehat{\mathbf{X}}^t - \mathbf{X}^t\| + (L_f^n - 1)/(L_f - 1)\epsilon$.
3) $\mathbb{E}_\lambda[\|f^n(\widehat{\mathbf{X}}^t) - f^n(\bar{\mathbf{X}}^t)\|] \le (1/2)L_f^n \|\widehat{\mathbf{X}}^t - \mathbf{X}^t\|$.

When $L_f > 1$, the accumulated error grows exponentially with respect to the steps, which agrees with empirical results. The two intermediate losses in the mixup training strategy enjoy smaller upper bounds than the original loss, which helps alleviate the error accumulation problem in the training stage. The proof of this theorem is shown in *Appendix C*.

G. Characteristics of HIMRAE

HIMRAE is characterized by the following features.

1) It learns edge-featured interaction graphs, where a directed edge indicates the existence of interaction and an edge feature implicitly decides the interacting effect. It does not rely on predefined adjacency matrices or edge types that may fail to describe heterogeneous interactions precisely.
2) The HAM can discriminate the importance of different types of agents, requiring only linear space complexity.
3) The graph entropy, an analogy of Shannon entropy on graphs, reduces the spatially propagated error by penalizing the graph complexity. Unlike sparsity constraints, it controls the graph complexity appropriately without learning an over sparse graph that hurts the prediction precision.
4) The mixup training strategy can balance the model's ability of single-step and multistep prediction, and thereby reduce the temporally accumulated error. Unlike teacher forcing, it can effectively narrow the gap between training and testing.
5) Both graph entropy and mixup training strategy have certain theoretical justifications that can guide the hyperparameter selection and help with experimental analysis.

IV. Experiments

A. Datasets

Following Li et al. [5], the proposed method is evaluated on three real-world datasets involving heterogeneous agents, NBA SportVU Dataset (NBA), Honda 3D Dataset [48], and Stanford Drone Dataset (SDD) [49], described as follows.

1) *NBA:* A trajectory dataset collected by NBA with the SportVU tracking system, containing the trajectories of the ball and ten players, with five from the home team and the rest from the visiting team.
2) *H3D:* A large-scale full-surround 3-D multiobject detection and tracking dataset, which provides the trajectories of eight types of traffic participants, including pedestrians, animals, cyclists, motorcyclists, cars, buses, trucks, and other vehicles.
3) *SDD:* A university campus trajectory dataset, containing the trajectories of six types of traffic participants, pedestrians, cars, buses, bikers, skaters, and carts.

The raw data are preprocessed by ourselves according to the original paper [5] with our best efforts. Details of the data preprocessing are described in *Appendix D*. Some important statistics of the datasets are listed in Table I.

B. Evaluation Metrics

Following previous works [7], [15], the average displacement error (ADE) and the final displacement error (FDE) are used as the evaluation metrics. The ADE and FDE for the predicted trajectory of a single agent are defined as follows.

1) *ADE:* The average Euclidean distance between the predicted positions and the ground-truth positions over all future steps.

TABLE I
STATISTICS OF DATASETS

Datasets	NBA	H3D	SDD
# Scenes	100	121	56
# Categories	3	8	6
# Agents	11	[4, 30]	[5, 61]
# Samples	10k	10k	27k
Sampling rate	2.5Hz	2.5Hz	2.5Hz
Length unit	Meter	Meter	Pixel
Numerical range of x	[0, 28.65]	[-16, 40]	[0, 1960]
Numerical range of y	[0, 15.24]	[-40, 40]	[0, 1977]
Historical steps/Future steps	5/10	5/10	8/12

2) *FDE:* The Euclidean distance between the predicted position and the ground-truth position at the final time step.

To compare the performance of different generative models, 20 samples are drawn, and both the minimum and the mean of ADE and FDE are calculated. Most of the existing works [5], [9], [15], [49] are evaluated in minimum ADE and FDE, which may be unreliable in real-world applications since little prior knowledge is available to select the best prediction [4]. Therefore, the mean ADE and FDE can serve as important complemental metrics that measure the average performance of generative models.

C. Baselines

Our method is compared with four GNN-based interaction modeling methods to validate its effectiveness.

1) *STGAT [15]:* This method applies the graph attention network to a fully connected graph to model the interactions among pedestrians.
2) *Social-STGCNN [9]:* This method defines a distance-based weighted interaction graph at each time step, and applies a spatio-temporal graph convolutional neural networks to capture the spatio-temporal dynamics of pedestrians.
3) *STAR [8]:* This method defines interaction graphs for pedestrians by distance thresholding, and adopts the Transformer for both spatial and temporal modeling.
4) *EvolveGraph [5]:* This method infers dynamic multirelational interaction graphs to model interactions among heterogeneous agents.

The codes of STGAT,[1] Social-STGCNN[2] and STAR[3] are public and thus directly used in our experiments. EvolveGraph is coded by ourselves according to the original paper. Implementation details of our method can be found in *Appendix D*.

Some variants of our method are introduced as follows to demonstrate the effectiveness of the HAM, the graph entropy, and the mixup training strategy.

1) *HIMRAE$_{HOMO}$:* This variant replaces the category-aware mappings of HAM with an identity mapping.
2) *HIMRAE w/GE:* A variant trained only with the graph entropy.
3) *HIMRAE w/mixup:* A variant trained only with the mixup training strategy.
4) *HIMRAE w/GE & Mixup:* A variant trained with both the graph entropy and the mixup training strategy.

D. Comparison With Baselines

As shown in Table II, our method HIMRAE w/GE & mixup achieves the lowest mean ADEs/FDEs on all datasets, and the lowest minimum ADEs/FDEs on the H3D dataset. The overall results validate the superiority of the proposed method.

All methods perform well on the NBA dataset, where most agents (players) are of the same type except the ball. STGAT achieves the lowest minimum ADEs/FDEs, while its mean ADEs and FDEs are larger than other baselines. STGAT is trained on the variety loss [15], i.e., evaluating the training batch several times and back-propagating through the lowest loss. This training strategy encourages STGAT to make diversified predictions with a possible large variance, leading to relatively poor performance on average. The problem becomes more severe on heterogeneous datasets like H3D and SDD. Our method outperforms other baselines on the NBA dataset since the category information is considered and the two training strategies further reduce the errors. Lower mean ADEs/FDEs show that our method can make more accurate predictions with lower variance, which is more reliable in practical scenarios.

On the H3D dataset, both our method and EvolveGraph achieve significantly lower predicting errors than the rest methods that are originally designed for pedestrian trajectory prediction. The results show that heterogeneous agents present different behavior patterns, which can not be effectively captured by homogeneous modeling. EvolveGraph, a baseline that considers both heterogeneous interaction modeling and decoding, makes even less accurate predictions than HIMRAE. Maybe the HAM in our method can distinguish the importance of heterogeneous neighbors, while EvolveGraph assigns equal weights to all neighbors in the decoder.

On the SDD dataset, methods designed for homogeneous systems achieve comparable predicting errors as our method and EvolveGraph. A possible reason is that the dominant class of agents on the SDD dataset is the pedestrian. Compared with EvolveGraph, our method achieves significantly lower mean ADEs/FDEs, because the graph entropy and the mixup training strategy can effectively reduce the accumulated error.

The predicted trajectories of HIMRAE w/GE & mixup and the most related baseline EvolveGraph are visualized in *Appendix D* for qualitative analysis.

E. Ablation Study

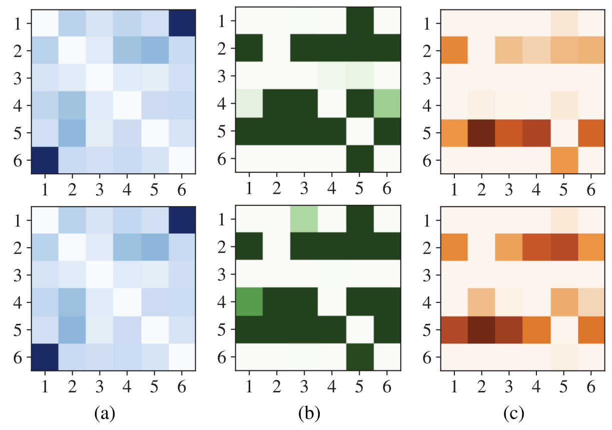
1) Effect of Heterogeneous Attention Mechanism: As shown in Table II, HIMRAE consistently outperforms HIMRAE$_{\text{HOMO}}$ on H3D and SDD datasets, while their performances are comparable on the NBA dataset. The results validate that the category-aware mappings are key components of HAM,

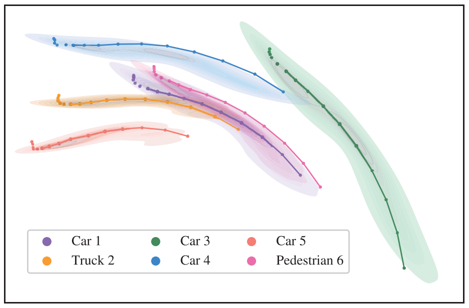
[1]https://github.com/huang-xx/STGAT
[2]https://github.com/abduallahmohamed/Social-STGCNN
[3]https://github.com/Majiker/STAR

TABLE II

ADEs/FDEs of Trajectory Prediction on Different Datasets. The Minimum and the Mean of the Metrics Calculated on 20 Random Samples Are Reported. For All Evaluation Metrics, Lower Is Better. The Values May Differ From Those Reported by Li et al. [5] Since the Experiments Are Rerun on All Datasets Using Our Data Preprocessing. Results Highlighted With * Are Statistically Significant With $p < 0.01$ by Comparing With EvolveGraph, the Best Baseline Overall. Details of the p-Values Can Be Found in Appendix D

Datasets	NBA		H3D		SDD	
Units	Meters		Meters		Pixels	
Metrics	$\min_{20}\downarrow$	$\text{mean}_{20}\downarrow$	$\min_{20}\downarrow$	$\text{mean}_{20}\downarrow$	$\min_{20}\downarrow$	$\text{mean}_{20}\downarrow$
STGAT	**0.07/0.13**	0.20/0.42	0.74/1.45	3.34/8.34	**12.7/23.3**	27.9/58.8
Social-STGCNN	0.10/0.18	0.18/0.34	1.51/2.46	2.61/5.01	21.2/32.7	40.5/73.6
STAR	0.08/0.18	0.25/0.62	0.90/2.09	2.56/6.92	15.9/32.4	31.1/70.6
EvolveGraph	0.12/0.21	0.19/0.34	0.58/1.00	0.81/1.49	20.9/39.8	28.3/54.0
HIMRAE w/ TF	0.55/0.72	1.75/2.92	5.43/10.17	12.05/25.81	178.0/332.4	315.8/592.9
HIMRAE w/ TF+	0.11/0.20	0.19/0.36	0.63/1.12	1.04/2.08	20.7/36.9	27.5/51.7
$\text{HIMRAE}_{\text{HOMO}}$	0.09/0.16	0.17/0.31	0.41/0.60	0.70/1.32	17.2/31.2	29.2/54.9
HIMRAE	0.09/0.16	0.16/0.29	0.40/0.57	0.69/1.28	16.9/30.5	29.0/54.3
HIMRAE w/ GE	0.09/0.16	0.16/0.29	0.37/0.54	0.64/1.22	18.0/33.6	26.8/51.2
HIMRAE w/ mixup	0.09/0.18	**0.14/0.28**	0.36/0.54	0.58/1.04	19.3/37.5	**25.2**/49.9
HIMRAE w/ GE & mixup	0.09*/0.17*	0.15*/**0.28***	**0.35*/0.50***	**0.56*/0.99***	18.8*/36.4*	**25.2***/**49.4***





(d)

Fig. 4. For matrices in (a)–(c), lighter indicates stronger connections. In (d), historical trajectories are in dots, ground-truth trajectories to be predicted are in solid lines, while the predicted trajectories are visualized using the kernel density estimation. (a) Distance based adjacency matrices at $t = 8$ and $t = 13$. (b) Interacting probabilities for $t \in [6, 10]$ and $t \in [11, 15]$. (c) Attention weights at $t = 8$ and $t = 13$.

which extends the normal attention mechanism to handle the interactions among heterogeneous agents.

To gain an intuitive understanding of HAM, this article visualizes the distance-based adjacency matrices, the probability matrices of the interacting relations, the attention weights, and the predicted trajectories in Fig. 4. Compared with the distance-based adjacency matrices, the inferred interaction graph can capture asymmetric relations between agents. For example, during $6 \le t \le 10$, Car 1 moves away from Truck 2, while Truck 2 is not affected by Car 1. During the same period, although the interacting graph tells Car 5 is affected by almost all other agents except Car 3, HAM helps to identify Pedestrian 6 as the most important agent to avoid. Similar results can be found for $t \in [11, 15]$. From this case, one finds that the interaction graph within a time window describes the coarsened interactions, while the attention weights describe the fine-grained interactions at each time step.

TABLE III

Average Graph Entropy and Density on Different Datasets

Datasets	NBA		H3D		SDD	
Metrics	GE	Density	GE	Density	GE	Density
HIMRAE	**0.95**	0.50	0.87	0.39	0.90	0.57
HIMRAE w/ GE	**0.95**	**0.43**	**0.84**	**0.38**	**0.88**	**0.56**

2) Effect of Graph Entropy: As shown in Table II, by adding the graph entropy, the mean ADEs/FDEs decreases on H3D and SDD datasets, while there is no significant difference on the NBA dataset. To see how graph entropy reduces the graph complexity, this article reports the average graph entropy and average graph density $|\mathcal{E}|/(N(N-1))$ of HIMRAE and HIMRAE w/GE in Table III.

From Table III, HIMRAE w/GE achieves lower graph complexity than HIMRAE on all datasets except that there is no significant difference in graph entropy on the NBA dataset. The results also agree with the prediction errors reported in Table II, where graph entropy is more effective on the H3D dataset and SDD dataset. However, as discussed in Section III-E, a graph with lower graph entropy is not necessarily sparser. A case study is conducted on the H3D dataset to illustrate that HIMRAE w/GE can infer graphs with lower graph entropy despite the graph density. As shown Fig. 5, graphs inferred by HIMRAE w/GE can achieve a lower

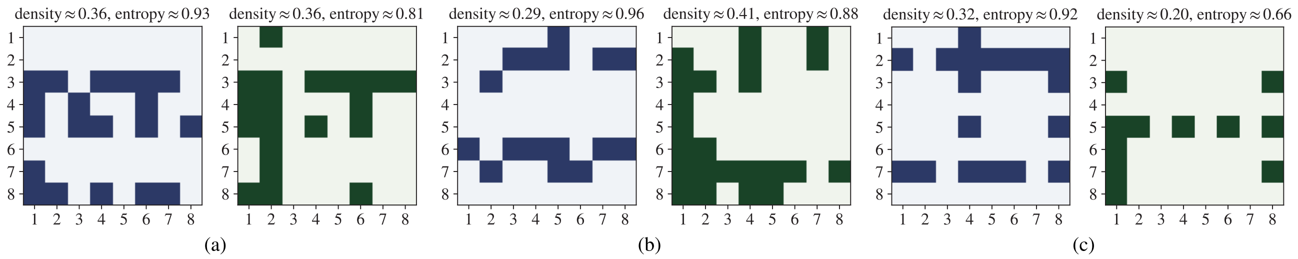


Fig. 5. Reduction of graph entropy at different graph density. Graphs inferred by HIMRAE and HIMRAE w/GE are in blue and green, respectively. (a) Case I: Equal density with smaller entropy. (b) Case II: Larger density with smaller entropy. (c) Case III: Smaller density with smaller entropy.

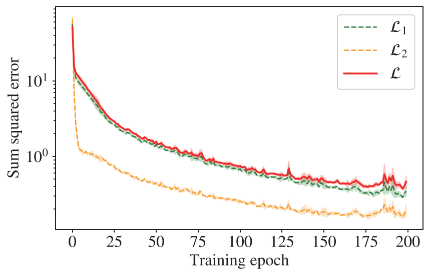


Fig. 6. Training loss of mixup strategy.

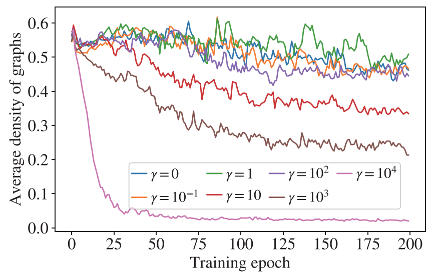


Fig. 7. Density versus γ.

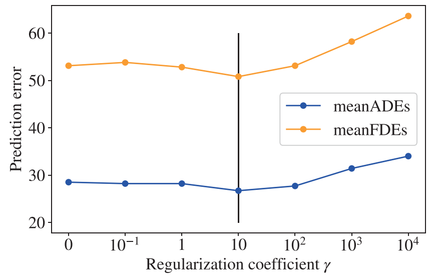


Fig. 8. Prediction error versus γ.

graph entropy when the graph densities are equal, larger or smaller, because the node distributions are less spread out.

3) Effect of Mixup Strategy: As shown in Table II, the mixup training strategy consistently reduces the mean ADEs/FDEs of HIMRAE on different datasets. The combination of mixup and the graph entropy can further improve the performance. Nonetheless, the minimum ADEs/FDEs increase on the NBA dataset and the SDD dataset. Maybe there exists a trade-off between minimum errors and mean errors, and the mixup training strategy tends to reduce the mean errors at the cost of increasing the minimum errors.

To explore how mixup eases the training of the model, this article visualizes the training loss and the two intermediate losses in Fig. 6. Besides, the intermediate loss $\mathcal{L}_1$ is slightly lower than the original loss $\mathcal{L}$ and $\mathcal{L}_2$ is much smaller. The result agrees with the theoretical analysis, and verifies that mixup does reduce the accumulated errors in the training stage. Note that $\mathcal{L}_2$ is significantly lower than $\mathcal{L}_1$, which means that imitating the behavior of multistep predictions based on corrected values is easier than directly learning from the ground-truth trajectories. Despite the advantages mentioned above, HIMRAE w/mixup requires more training time, as it includes extra forward and backward passes in one epoch.

V. Discussion

A. Selection of Regularization Coefficient γ

The hyperparameter γ controls the strengths of complexity regularization. The best γ varies with datasets since the units of positions and the dynamics are different. Without loss of generality, this article illustrates the choice of γ by training HIMRAE w/GE on the SDD dataset with γ selected from $\{0, 10^{-1}, 1, 10, 10^2, 10^3, 10^4\}$. The average graph densities during the training procedure are shown in Fig. 7, and the mean ADEs/FDEs are shown in Fig. 8. From Fig. 7, a larger γ approximately leads to a sparser graph, which agrees with the theoretical results in Corollary 1. When $\gamma = 10^4$, the average graph density declines dramatically to zero, resulting in an almost empty graph. Therefore, a suitable γ should be chosen from the rest candidates. From Fig. 8, the model reaches the minimum predictor error at $\gamma = 10$.

B. Comparing Graph Entropy With Sparsity Constraints

Apart from graph entropy, there are other choices of regularization to penalize the graph complexity like the sparsity constraints. Without loss of generality, this article considers two simple and differentiable candidates, the density constraint, and the maximum node degree constraint, i.e.,

$$R_{\text{density}}\left(\mathbf{Z}^{1:M}\right) = \frac{\gamma}{M}\sum_{m=1}^{M}\frac{1}{N(N-1)}\sum_{i\neq j} z_{ij}^m$$

$$R_{\text{degree}}\left(\mathbf{Z}^{1:M}\right) = \frac{\gamma}{M}\sum_{m=1}^{M}\frac{\max_j d_j^m}{N}.$$

The resulting variants of our method are denoted as **HIMRAE w/R_{density}** and **HIMRAE w/R_{degree}**.

A comparative experiment is conducted on the H3D dataset to show the advantages of graph entropy over the sparsity constraints in trajectory prediction. The regularization coefficients for HIMRAE w/R_{density} and HIMRAE w/R_{degree} are chosen as 10^{-4} and 10^{-4}, respectively, via cross-validation. The

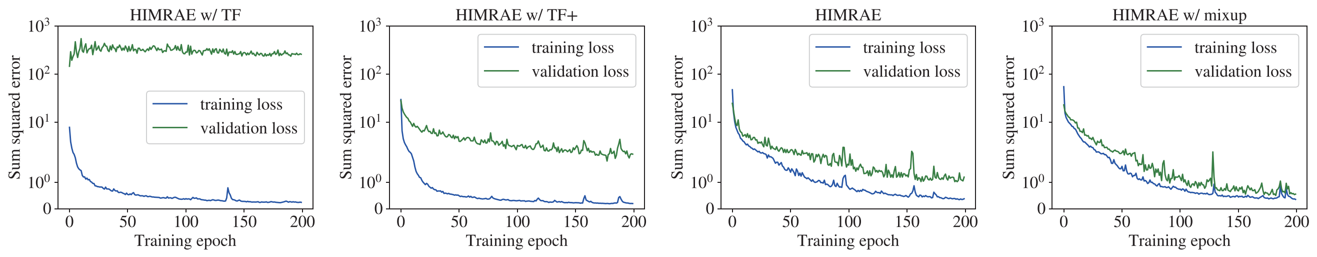


Fig. 9. Training loss and validation loss of different methods.

TABLE IV
EFFECT OF DIFFERENT REGULARIZATION TERMS

Metrics	$\min_{20} \downarrow$	$\text{mean}_{20} \downarrow$	Density (%)
HIMRAE	0.40/0.57	0.69/1.28	38.6
HIMRAE w/ GE	**0.37/0.54**	**0.64/1.22**	37.9
HIMRAE w/ R_{density}	0.40/0.58	0.69/1.30	**36.0**
HIMRAE w/ R_{degree}	0.42/0.62	0.71/1.33	37.2

minimum ADEs/FDEs, mean ADEs/FDEs, and the average graph density are reported in Table IV.

From Table IV, HIMRAE w/R_{density} and HIMRAE w/R_{degree} achieve lower densities, while HIMRAE w/GE achieves the lowest prediction errors. The results demonstrate that sparer graphs do not necessarily lead to lower prediction error. Compared with HIMRAE w/R_{density}, HIMRAE w/GE can rationally reduce the graph complexity without learning over sparse graphs that hurt the prediction precision. Compared with HIMRAE w/R_{degree}, our method can impose a global constraint to the interactions instead of only paying attention to the node with the maximum degree.

C. Comparing Mixup Training Strategy With Teacher Forcing

This section compares the performance of mixup training strategy with teacher forcing to show our advantages. Unfortunately, we find teacher forcing fails to train satisfactory models in our dataset [see Table II and Fig. 9(a)]. For a fair comparison, this article also considers a variant of teacher forcing that feeds the ground-truth positions at every τ steps. The methods with teacher forcing and its variant are denoted as **HIMRAE w/TF** and **HIMRAE w/TF+**, respectively. This article evaluates the prediction errors of teacher forcing and its variant on all datasets, and the results are shown in Table II.

The results show that models trained under teacher forcing fail to make accurate predictions. HIMRAE w/TF+ performs much better, but on the NBA dataset and H3D dataset, its performance is even worse than HIMRAE, which is trained without the mixup strategy. Therefore, though sharing similar ideas with teacher forcing, the mixup strategy is substantially different. It can better balance the model's ability for both single-step and multistep prediction.

This article further visualizes the training loss and validation loss of HIMRAE, HIMRAE w/mixup, HIMRAE w/TF and HIMRAE w/TF+ on the H3D dataset. As shown in Fig. 9, teacher forcing and its variant suffer from a large gap between training loss and validation loss, which is even larger than that of HIMRAE. By contrast, HIMRAE w/mixup effectively reduces the training-validation gap.

D. Quality of the Interaction Graphs

This section makes an attempt to evaluate the quality of the learned interaction graphs. In our model, the edges in a graph are conditionally independent given the historical trajectories. Therefore, it suffices to evaluate the quality of each edge, which indicates the existence of a directed interaction and its effect. Since the interacting effect is encoded by a continuous hidden variable, it is hard to tell its quality. Therefore, this article tries to check if our model can identify the existence of an edge.

The problem is studied from two aspects.

1) *Necessity of the Edges:* Whether or not the inferred edges contain no redundancy.
2) *Sufficiency of the Edges:* Whether or not the inferred edges contain all useful edges.

Two types of tests are introduced to examine the two aspects above, respectively.

1) Remove an edge at a time to see if the prediction error increases significantly. Edges that do not affect the prediction error are thought redundant.
2) Add an edge at a time to see if the prediction error decreases significantly. Edges that help reduce the prediction error are thought missing.

Let E, E_1, E_2 denote the number of inferred edges, the number of redundant edges, and the number of missing edges, respectively. Then, $E - E_1 + E_2$ is an approximate number of all useful edges. Consequently, the redundant rate and the missing rate are defined as follows.

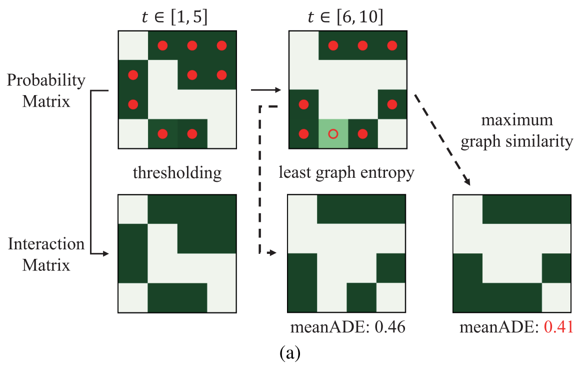
1) *Redundant Rate:* $E_1/(E - E_1 + E_2)$.
2) *Missing Rate:* $E_2/(E - E_1 + E_2)$.

By definition, a lower redundant rate and missing rate indicate higher qualities. This article evaluates the two metrics on 256 random samples from the testing set of all datasets. For each sample, this article first infers the interaction graph via a learned model HIMRAE, and then, removes or adds an edge at a time to see if the mean ADE increases significantly. The difference is measured by a statistical significance test over 20 generated trajectories at the level of $p < 0.05$.

From Table V, our method can learn satisfactory interaction graphs with the redundant rate and the missing rate lower than 10%. Still, the missing rate is relatively high on the H3D dataset, indicating that our method needs to consider potentially missing edges for further improvement.

E. Choices of the Interaction Graphs

In the experiments, the interaction graphs are sampled randomly to evaluate the performance, while in practical



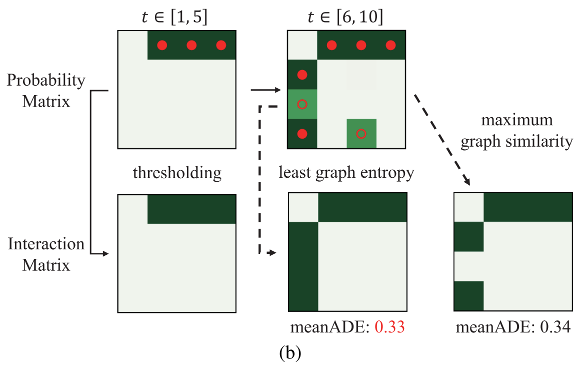


Fig. 10. Choices of interaction graphs. In the probability matrices, red dots stand for probabilities larger than 0.8, while red circles stand for probabilities in [0.2, 0.8]. The lower mean ADEs are highlighted in red. (a) Case I: Maximum graph similarity leads to lower prediction error. (b) Case II: Least graph entropy leads to lower prediction error.

TABLE V
QUALITY OF INFERRED INTERACTION GRAPHS ON DIFFERENT DATASETS

Datasets	NBA	H3D	SDD
Redundant Rate (%)	2.34	3.11	2.91
Missing Rate (%)	2.32	7.16	1.79

scenarios, users might be interested in how to choose a “most possible” graph. First of all, there is generally no gold standard because the ground-truth graphs are unobservable. When the future trajectories are available, one can select the graph that minimizes the prediction error. However, when they are unknown, e.g., in the testing stage, an alternative criterion should be chosen before selecting the best graph.

In our model, edges in a graph are conditionally independent. Once the encoder infers the edge probabilities, the existence of each edge can be decided independently. Edges with small uncertainty are selected deterministically via thresholding, while the rest are selected based on some heuristics. An exemplar procedure is described as follows.

1) Filter out edges with probabilities smaller than a given threshold θ_{low}.
2) Select edges with probabilities larger than another given threshold θ_{high}.
3) For edges with probabilities in $[\theta_{\text{low}}, \theta_{\text{high}}]$, this article adopts the following heuristics.
    a) Select the graph with the least graph entropy.
    b) Select the graph that is most consistent with the previous one by measuring the graph similarity, e.g., the ℓ_1-norm.

Other heuristics may be explored based on users’ prior knowledge.

Two cases from the H3D dataset are used to illustrate the procedure above. This article sets $\theta_{\text{low}} = 0.2$ and $\theta_{\text{high}} = 0.8$ in the experiments. The results are shown in Fig. 10. In Case I, heuristics b) is better than heuristics a) in terms of prediction error, while the situation reverses in Case II. The case study demonstrates that neither heuristics can ensure the best graph that leads to the lowest reconstruction error.

VI. CONCLUSION

This article proposes an encoder–decoder framework HIMRAE that models the interactions among heterogeneous agents and reduces the accumulated errors of multiagent trajectory prediction. The encoder can use the historical trajectories to infer dynamic interaction graphs, featured by the interacting relations and corresponding interacting effects. The decoder applies an HAM of linear space complexity to the inferred graphs to model the fine-grained heterogeneous interactions. This article further figures out the error sources of recursive multistep prediction from both the spatial and the temporal aspects. The graph entropy is adopted to control the spatially propagated errors and the mixup strategy is proposed to reduce the temporally accumulated errors. Both strategies have theoretical justifications. Extensive experiments on real-world heterogeneous datasets validate the effectiveness of HIMRAE. The results also show the advantages of graph entropy over sparsity constraints, and the advantages of mixup strategy over teacher forcing.

In practical scenarios, the model may be required to predict the trajectories of agents from an unseen category. To tackle this challenge, future work includes using meta-learning to transfer the knowledge of interaction modeling to new agent types. Besides, the interaction graphs inferred by the neural networks lack interpretability. It is worthwhile to explore interpretable interaction graph generation.

REFERENCES

[1] H.-X. Hu, C. Wen, and G. Wen, “A distributed Lyapunov-based redesign approach for heterogeneous uncertain agents with cooperation–competition interactions,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6946–6960, Nov. 2022.
[2] H. Zhang, H. Liang, Z. Wang, and T. Feng, “Optimal output regulation for heterogeneous multiagent systems via adaptive dynamic programming,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 1, pp. 18–29, Jan. 2017.
[3] X. Du, J. Wang, S. Chen, and Z. Liu, “Multi-agent deep reinforcement learning with spatio-temporal feature fusion for traffic signal control,” in *Proc. ECML-PKDD*, 2021, pp. 470–485.
[4] B. Ivanovic and M. Pavone, “The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2375–2384.
[5] J. Li, F. Yang, M. Tomizuka, and C. Choi, “EvolveGraph: Multi-agent trajectory prediction with dynamic relational reasoning,” in *Proc. NIPS*, 2020, pp. 19783–19794.
[6] P. Kothari, S. Kreiss, and A. Alahi, “Human trajectory forecasting in crowds: A deep learning perspective,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 7386–7400, Jul. 2022.
[7] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social LSTM: Human trajectory prediction in crowded spaces,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 961–971.

[8] C. Yu, X. Ma, J. Ren, H. Zhao, and S. Yi, "Spatio-temporal graph transformer networks for pedestrian trajectory prediction," in *Proc. ECCV*, 2020, pp. 507–523.

[9] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-STGCNN: A social spatio-temporal graph convolutional neural network for human trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14424–14432.

[10] L. Shi et al., "SGCN: Sparse graph convolution network for pedestrian trajectory prediction," in *Proc. CVPR*, 2021, pp. 8994–9003.

[11] H. Zhou, D. Ren, H. Xia, M. Fan, X. Yang, and H. Huang, "AST-GNN: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction," *Neurocomputing*, vol. 445, pp. 298–308, Jul. 2021.

[12] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, pp. 270–280, Jun. 1989.

[13] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," 2015, *arXiv:1511.06732*.

[14] W. Zhang, Y. Feng, F. Meng, D. You, and Q. Liu, "Bridging the gap between training and inference for neural machine translation," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 4334–4343.

[15] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, "STGAT: Modeling spatial–temporal interactions for human trajectory prediction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6272–6281.

[16] B. Yang, G. Yan, P. Wang, C.-Y. Chan, X. Song, and Y. Chen, "A novel graph-based trajectory predictor with pseudo-oracle," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 4, 2021, doi: 10.1109/TNNLS.2021.3084143.

[17] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," in *Proc. ICML*, 2018, pp. 2688–2697.

[18] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Proc. ECCV*, 2020, pp. 683–700.

[19] J. Li, H. Ma, Z. Zhang, J. Li, and M. Tomizuka, "Spatio-temporal graph dual-attention network for multi-agent prediction and tracking," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 10556–10569, Aug. 2022.

[20] X. Mo, Z. Huang, Y. Xing, and C. Lv, "Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9554–9567, Jul. 2022.

[21] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "TrafficPredict: Trajectory prediction for heterogeneous traffic-agents," in *Proc. AAAI*, 2019, pp. 6120–6127.

[22] Z. Li, C. Lu, Y. Yi, and J. Gong, "A hierarchical framework for interactive behaviour prediction of heterogeneous traffic participants based on graph neural network," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9102–9114, Jul. 2022.

[23] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, "A survey of heterogeneous information network analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 17–37, Jan. 2017.

[24] H. Ji, X. Wang, C. Shi, B. Wang, and P. Yu, "Heterogeneous graph propagation network," *IEEE Trans. Knowl. Data Eng.*, early access, May 11, 2021, doi: 10.1109/TKDE.2021.3079239.

[25] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. ICML*, 2017, pp. 1263–1272.

[26] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proc. KDD*, 2019, pp. 793–803.

[27] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. ESWC*, 2018, pp. 593–607.

[28] X. Wang et al., "Heterogeneous graph attention network," in *Proc. WWW*, 2019, pp. 2022–2032.

[29] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proc. Web Conf.*, Apr. 2020, pp. 2704–2710.

[30] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 44–63, May 2019.

[31] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 753–763.

[32] S. Chen, J. Wang, and G. Li, "Neural relational inference with efficient message passing mechanisms," in *Proc. AAAI*, 2021, pp. 7055–7063.

[33] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. ICLR*, 2014, pp. 1–14.

[34] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.

[35] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *Proc. ICLR*, 2017, pp. 1–20.

[36] A. Vaswani et al., "Attention is all you need," in *Proc. NIPS*, 2017, pp. 5998–6008.

[37] Y. Xing, C. Lv, and D. Cao, "Personalized vehicle trajectory prediction based on joint time-series modeling for connected vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1341–1352, Feb. 2020.

[38] G. Chen, J. Li, N. Zhou, L. Ren, and J. Lu, "Personalized trajectory prediction via distribution discrimination," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 15560–15569.

[39] J. Li, F. Yang, H. Ma, S. Malla, M. Tomizuka, and C. Choi, "RAIN: Reinforced hybrid attention inference network for motion forecasting," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 16076–16086.

[40] M. Fey, J. E. Lenssen, F. Weichert, and J. Leskovec, "GNNAutoScale: Scalable and expressive graph neural networks via historical embeddings," in *Proc. ICML*, 2021, pp. 3294–3304.

[41] M. Dehmer and A. Mowshowitz, "A history of graph entropy measures," *Inf. Sci.*, vol. 181, no. 1, pp. 57–78, 2011.

[42] G. Luo et al., "Graph entropy guided node embedding dimension selection for graph neural networks," in *Proc. IJCAI*, 2021, pp. 2767–2774.

[43] J. Wu, X. Chen, K. Xu, and S. Li, "Structural entropy guided graph hierarchical pooling," in *Proc. ICML*, 2022, pp. 24017–24030.

[44] A. W. Marshall, I. Olkin, and B. C. Arnold, *Inequalities: Theory of Majorization and Its Applications*. New York, NY, USA: Springer, 2011, ch. 4, pp. 156–157.

[45] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *Proc. ICLR*, 2018, pp. 1–13.

[46] J. Liang, L. Jiang, and A. Hauptmann, "SimAug: Learning robust representations from simulation for trajectory prediction," in *Proc. ECCV*, 2020, pp. 275–292.

[47] A. Venkatraman, "Training strategies for time series: Learning for prediction, filtering, and reinforcement learning," Ph.D. dissertation, Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2017.

[48] A. Patil, S. Malla, H. Gang, and Y.-T. Chen, "The H3D dataset for full-surround 3D multi-object detection and tracking in crowded urban scenes," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 9552–9557.

[49] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *Proc. ECCV*, 2016, pp. 549–565.

Siyuan Chen received the B.S. degree in mathematics and applied mathematics from Sun Yat-sen University, Guangzhou, China, in 2018, where he is currently pursuing the Ph.D. degree in computer science and technology with the School of Computer Science and Engineering.

His research interests include graph neural networks, crowd intelligence, and social network analysis.

Jiahai Wang (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Toyama, Toyama, Japan, in 2005.

In 2005, he joined Sun Yat-sen University, Guangzhou, China, where he is currently a Full Professor with the School of Computer Science and Engineering. He is also leading an Intelligent Optimization and Learning Laboratory, Sun Yat-sen University. He has authored a series of papers in leading conferences and top journals, including AAAI, ACL, and IEEE TRANSACTIONS. His main research interests include computational intelligence (deep neural networks and metaheuristics) and its applications.

Dr. Wang is a Distinguished Member of CCF, China.

Supplementary file-Heterogeneous Interaction Modeling With Reduced Accumulated Error for Multi-Agent Trajectory Prediction

Siyuan Chen, and Jiahai Wang, *Senior Member, IEEE*

***Abstract*—This is the supplementary material of the paper "Heterogeneous Interaction Modeling With Reduced Accumulated Error for Multi-Agent Trajectory Prediction" for IEEE Transactions on Neural Networks and Learning Systems. A summary of notations, proofs of theoretical results and extra experimental details are shown in this supplementary material due to page of limit of paper.**

List of Tables

List of Figures

This work is supported by the National Key R&D Program of China (2018AAA0101203), the National Natural Science Foundation of China (62072483), and the Guangdong Basic and Applied Basic Research Foundation (2022A1515011690, 2021A1515012298). *(Corresponding author: Jiahai Wang.)*

Siyuan Chen and Jiahai Wang are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China (e-mail: chensy47@mail2.sysu.edu.cn, wangjiah@mail.sysu.edu.cn).

Appendix A
Notations

Notations used in the main paper are summarized in Table A1.

Appendix B
Theoretical Analysis for the Graph Entropy

This section analyzes the effect of the graph entropy from the optimization perspective. There are two main results: (1) For a graph with a given number of edges, the graph entropy is minimized when the edges are centered on a few nodes. (2) A graph with fewer edges has a smaller lower bound of graph entropy. The proof relies on a useful inequality from the majorization theory [1].

Recall that the graph entropy used in this paper is defined as

$$\mathrm{H}_{\mathcal{G}}[\mathbf{Z}^m] = -\frac{1}{\ln N}\sum\nolimits_{j=1}^{N}(d_j^m/|\mathcal{E}^m|)\ln(d_j^m/|\mathcal{E}^m|), \tag{A1}$$

where $\mathbf{Z}^m$ is the adjacency matrix of N nodes in the m-th time window, d_j^m is the in-degree of node j. For simplicity, this paper omits the superscript m and the normalization constant $\frac{1}{\ln N}$ that does not affect the optimization in the following text.

Let $\mathbf{p} = \frac{1}{|\mathcal{E}|}(d_1, \ldots, d_N)$, the graph entropy in Eq. (A1) can be rewritten as,

$$\mathrm{H}_{\mathcal{G}}[\mathbf{Z}] = -\sum_{i=1}^{N} p_i \ln p_i, \tag{A2}$$

where p_i is the i-th element of $\mathbf{p}$.

The graph entropy defined in Eq. (A2) is essentially the Shannon entropy. It is well-known that Shannon entropy is maximized for the uniform distribution and minimized for the single-point distribution. Therefore, it can be conjectured that a graph with less spread-out edges has a lower entropy. This paper formalizes the intuition via the majorization theory.

For any vector $\mathbf{x} \in \mathbb{R}^N$, let $\mathbf{x}^{\downarrow} \triangleq (x_{[1]}, \ldots, x_{[N]})$ denote its decreasing order, i.e., $x_{[1]} \geq \cdots \geq x_{[N]}$. Then, the definition of majorization and a useful inequality are introduced are follows.

Definition 1 (Majorization)**.** *For any vectors* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$*, we say* $\mathbf{x}$ *majorizes* $\mathbf{y}$ *(or* $\mathbf{y}$ *is majorized by* $\mathbf{x}$*), denoted by* $\mathbf{x} \succ \mathbf{y}$

TABLE A1
Summary of Notations.

Notations	Descriptions
$\mathbf{x}_i^t = (x_i^t, y_i^t)$	The 2D coordinate of agent i at time t.
c_i	The category of agent i.
T_h, T_f	Historical steps and future steps.
$\mathbf{X}^{1:T_h} = \{\mathbf{x}_i^{1:T_h}\}_{i=1}^N$	The trajectories of N agents from time $t=1$ to $t=T_h$.
$\widehat{\mathbf{X}}^t$	Predicted positions at time t.
$\mathcal{G}^t = (\mathcal{V}, \mathcal{E}^t)$	A directed interaction graph at time t, with each agent $i \in \mathcal{V}$ as a node and the interacting relation from agent i to agent j as an directed edge $(i,j) \in \mathcal{E}^t$.
τ	The size for a time window.
$M = \lfloor (T_h + T_f)/\tau \rfloor$	The maximum number of the time window.
$\mathbf{Z}^m \in \{0,1\}^{N\times N}$	The interacting relations at time t, with $z_{ij}^m = 1$ indicating a directed edge (i,j).
$\mathbf{E}^m \in \mathbb{R}^{N\times N\times D}$	The interacting effects at time t, with $\mathbf{e}_{ij}^m$ representing the impact of agent i on agent j.
$q_\phi\left(\mathbf{Z}^m, \mathbf{E}^m \mid \mathbf{X}^{1:m\tau}\right)$	The encoder parameterized by ϕ.
$p_\theta\left(\mathbf{X}^{1+m\tau:(m+1)\tau} \mid \mathbf{X}^{1:m\tau}, \mathbf{Z}^m, \mathbf{E}^m\right)$	The decoder parameterized by θ.
f_{dec}	A simplified notation of the decoder.
f_{emb}	An MLP for trajectory embedding.
f_v	An MLP updating the node embeddings.
$f_e, \widetilde{f}_e, g_e$	MLPs updating the edge embeddings.
f_{proj}	An MLP projecting the edge representations to a 1D space.
f_V	An MLP updating the value vector.
f_Q, f_K	Single-layer perceptrons updating the query vector and the key vector respectively.
$g_Q^{c_i}, g_K^{c_i}, g_V^{c_i}$	Category-aware single-layer perceptrons in the heterogeneous attention mechanism.
GRU_{c_j}	A category-aware gated recurrent unit.
f_{out}	An MLP predicting the change in positions.
$\mathbf{v}_j^m$	The trajectory embedding of agent j in the m-th time window.
$\widetilde{\mathbf{v}}_j^m, \widetilde{\mathbf{e}}_{ij}^m$	Intermediate node representations and edge representations in the encoder.
$\mathbf{r}_{ij}^m$	The hidden state for the interacting relations of edge (i,j).
$\mathbf{m}_j^t$	The aggregated effects from the neighbors of agent j at time t.
$\mathbf{h}_j^t$	The hidden state of agent j at time t in the decoder.
$\mathrm{H}_{\mathcal{G}}[\mathbf{Z}^m]$	The graph entropy.
γ	The regularization coefficient for the graph entropy.
$\mathrm{Mix}_\lambda(\cdot,\cdot)$	The mixup operator, with λ as the mixing coefficient.
$\bar{\mathbf{X}}^t$	A convex combination of the predicted position $\widehat{\mathbf{X}}^t$ and the ground truth $\mathbf{X}^t$.

(or $\mathbf{y} \prec \mathbf{x}$*), if*

$$\sum_{i=1}^{k} x_{[i]} \geq \sum_{i=1}^{k} y_{[i]}, 1 \leq k < N,$$
$$\sum_{i=1}^{N} x_{[i]} = \sum_{i=1}^{N} y_{[i]}.$$

Lemma 1 (Hardy-Littlewood-Pólya inequality [1])**.** *Let* ϕ *be a concave function over* $\Omega \subseteq \mathbb{R}^N$*. For any vectors* $\mathbf{x}, \mathbf{y} \in \Omega$*, if* $\mathbf{x} \succ \mathbf{y}$*, then*

$$\sum_{i=1}^{N} \phi(x_i) \leq \sum_{i=1}^{N} \phi(y_i).$$

In our case, $\Omega = \{\mathbf{x} \in \mathbb{R}^N_{\geq 0} : \mathbf{1}^T\mathbf{x} = 1\}$ is a probability simplex and $\phi(x) = -x \ln x$ is the entropy function with $\phi(0) = 0$. Then, the problem of finding the minimizer of the graph entropy reduces to finding the minimum elements of Ω if they exist. Denote $\Omega(N, |\mathcal{E}|) = \left\{\frac{1}{|\mathcal{E}|}(d_1, \ldots, d_N) \in \mathbb{R}^N_{\geq 0} : |\mathcal{E}| = \sum_{i=1}^N d_i\right\}$ as the probability simplex for a graph with N nodes and $|\mathcal{E}|$ edges. The minimizer and the maximizer of the graph entropy are characterized as follows.

Theorem 1 (Minimizer of Graph Entropy)**.** *For an* N*-node directed graph with edge counts* $|\mathcal{E}|$*,* $\min \mathrm{H}_{\mathcal{G}}[\mathbf{Z}] = \frac{k(N-1)}{|\mathcal{E}|} \ln \frac{|\mathcal{E}|}{N-1} - \frac{e}{|\mathcal{E}|} \ln \frac{e}{|\mathcal{E}|}$*, where* k *and* e *are determined by* $|\mathcal{E}| = k(N-1) + e, 0 \leq e < N-1$*. The minimum is achieved when*

$$\mathbf{p}^{\downarrow} = \Big(\underbrace{\frac{N-1}{|\mathcal{E}|}, \ldots, \frac{N-1}{|\mathcal{E}|}}_{k}, \frac{e}{|\mathcal{E}|}, \underbrace{0, \ldots, 0}_{N-k-1}\Big). \tag{A3}$$

Specifically, $\min \mathrm{H}_{\mathcal{G}}[\mathbf{Z}] = 0$ *for* $|\mathcal{E}| \leq N-1$*. The minimum is achieved when* $\mathbf{p}^{\downarrow} = (1, 0, \ldots, 0)$*.*

Before the proof of Theorem 1, this paper first introduces a useful lemma that will be used repeatedly.

Lemma 2. *For any vectors* $\mathbf{x}, \mathbf{y} \in \Omega_c = \{\mathbf{x} \in \mathbb{R}^N_{\geq 0} : \mathbf{1}^T\mathbf{x} = c\}$ *with* c *as a constant, if there exists an integer* ℓ *such that* $x_{[i]} \geq y_{[i]}, 1 \leq i \leq \ell$ *and* $x_{[i]} \leq y_{[i]}, \ell+1 \leq i \leq N$*, then* $\mathbf{x} \succ \mathbf{y}$*.*

Proof. Since $x_{[i]} \geq y_{[i]}, 1 \leq i \leq \ell$

$$\sum_{i=1}^{j} x_{[i]} \geq \sum_{i=1}^{j} y_{[i]}, 1 \leq j \leq \ell.$$

On the other hand, $x_{[i]} \leq y_{[i]}, \ell+1 \leq i \leq N$. Thus,

$$\sum_{i=j}^{N} x_{[i]} \leq \sum_{i=j}^{N} y_{[i]}, \ell < j \leq N,$$

which implies that

$$\sum_{i=1}^{j} x_{[i]} = c - \sum_{i=j+1}^{N} x_{[i]} \geq c - \sum_{i=j+1}^{N} y_{[i]} = \sum_{i=1}^{j} y_{[i]}, \ell < j \leq N.$$

Hence, we prove that $\mathbf{x} \succ \mathbf{y}$. □

The constant c in Lemma 2 is equal to 1 for probability simplex. The proof of Theorem 1 is presented as follows.

Proof. For any probability vector $\mathbf{q} \in \Omega(N, |\mathcal{E}|)$, this paper will prove that $\mathbf{p} \succ \mathbf{q}$. Firstly, note that $p_{[i]} \geq q_{[i]}$ for $1 \leq i \leq k$ and $p_{[i]} \leq q_{[i]}$ for $k < i \leq N$. Set

$$\ell = \begin{cases} k & \text{if } p_{[k+1]} \leq q_{[k+1]}, \\ k+1 & \text{otherwise.} \end{cases}$$

Then ℓ satisfies the condition in Lemma 2, which implies that $\mathbf{p} \succ \mathbf{q}$. It can be evaluated that

$$\begin{aligned} &\min \mathrm{H}_{\mathcal{G}}[\mathbf{Z}] \\ &= -\sum_{i=1}^{k} \frac{N-1}{|\mathcal{E}|} \ln \frac{N-1}{|\mathcal{E}|} - \frac{e}{|\mathcal{E}|} \ln \frac{e}{|\mathcal{E}|} - \sum_{i=1}^{N-k-1} \phi(0) \\ &= \frac{k(N-1)}{|\mathcal{E}|} \ln \frac{|\mathcal{E}|}{N-1} - \frac{e}{|\mathcal{E}|} \ln \frac{e}{|\mathcal{E}|}. \end{aligned}$$

When $|\mathcal{E}| \leq N-1$, one finds $k = 0$ and $e = |\mathcal{E}|$. In this case, $\mathbf{p}^{\downarrow} = (1, 0, \ldots, 0)$ and $\min \mathrm{H}_{\mathcal{G}}[\mathbf{Z}] = 0$. □

Corollary 1. *Given the number of nodes, a graph with fewer edges has a smaller lower bound of graph entropy.*

Proof. Let $\mathbf{p}$ and $\mathbf{q}$ be the minimum elements of $\Omega(N, |\mathcal{E}|)$ and $\Omega(N, |\mathcal{E}'|)$, respectively. Suppose that $|\mathcal{E}| = k(N-1) + e \leq |\mathcal{E}'| = k'(N-1) + e'$. One can verify that $k' \geq k$, $p_{[i]} \geq q_{[i]}, 1 \leq i \leq k$ and $p_{[i]} \leq q_{[i]}, k+2 \leq i \leq N$. Set

$$\ell = \begin{cases} k & \text{if } p_{[k+1]} \leq q_{[k+1]}, \\ k+1 & \text{otherwise.} \end{cases}$$

Then ℓ satisfies the condition in Lemma 2, which implies that $\mathbf{p} \succ \mathbf{q}$. Hence $\min \mathrm{H}_{\mathcal{G}}[\mathbf{Z}] \leq \min \mathrm{H}_{\mathcal{G}}[\mathbf{Z}']$. □

Appendix C
Proof of Theorem 2

Theorem 2. *Let* f *be a learned model with bounded single-step prediction error* $\|f(\mathbf{X}^t) - \mathbf{X}^{t+1}\| \leq \epsilon$*. Assume that* f *is Lipschitz continuous with constant* L_f *under the norm* $\|\cdot\|$*. Then, the following upper bounds hold,*

(1) $\|f^n(\widehat{\mathbf{X}}^t) - \mathbf{X}^{t+n}\| \leq L_f^n \|\widehat{\mathbf{X}}^t - \mathbf{X}^t\| + \frac{L_f^n - 1}{L_f - 1}\epsilon$;

(2) $\mathbb{E}_{\lambda}[\|f^n(\bar{\mathbf{X}}^t) - \mathbf{X}^{t+n}\|] \leq \frac{1}{2} L_f^n \|\widehat{\mathbf{X}}^t - \mathbf{X}^t\| + \frac{L_f^n - 1}{L_f - 1}\epsilon$;

(3) $\mathbb{E}_{\lambda}[\|f^n(\widehat{\mathbf{X}}^t) - f^n(\bar{\mathbf{X}}^t)\|] \leq \frac{1}{2} L_f^n \|\widehat{\mathbf{X}}^t - \mathbf{X}^t\|$.

Proof. Proposition (1) can be proved by induction. Firstly,

$$\begin{aligned} &\|f^n(\widehat{\mathbf{X}}^t) - \mathbf{X}^{t+n}\| \\ &= \|f^n(\widehat{\mathbf{X}}^t) - f(\mathbf{X}^{t+n-1}) + f(\mathbf{X}^{t+n-1}) - \mathbf{X}^{t+n}\| \\ &\leq \|f^n(\widehat{\mathbf{X}}^t) - f(\mathbf{X}^{t+n-1})\| + \|f(\mathbf{X}^{t+n-1}) - \mathbf{X}^{t+n}\| \\ &\leq L_f \|f^{n-1}(\widehat{\mathbf{X}}^t) - \mathbf{X}^{t+n-1}\| + \epsilon \end{aligned}$$

where the first inequality is due to the triangle inequality, and the second inequality is due to the assumptions in the theorem. Denoting $\epsilon_n = \|f^n(\widehat{\mathbf{X}}^t) - \mathbf{X}^{t+n}\|$, the result above can be rewritten as

$$\epsilon_n \leq L_f \epsilon_{n-1} + \epsilon, n \geq 1.$$

By induction, it can be proved that

$$\|f^n(\widehat{\mathbf{X}}^t) - \mathbf{X}^{t+n}\| \leq L_f^n \|\widehat{\mathbf{X}}^t - \mathbf{X}^t\| + \frac{L_f^n - 1}{L_f - 1}\epsilon.$$

This completes the proof of proposition (1). A similar result can be obtained for proposition (2) as follows,

$$\begin{aligned}\|f^n(\widehat{\mathbf{X}}^t) - \mathbf{X}^{t+n}\| &\leq L_f^n \|\bar{\mathbf{X}}^t - \mathbf{X}^t\| + \frac{L_f^n - 1}{L_f - 1}\epsilon \\ &= \lambda L_f^n \|\widehat{\mathbf{X}}^t - \mathbf{X}^t\| + \frac{L_f^n - 1}{L_f - 1}\epsilon,\end{aligned}$$

where the equality is by the definition of $\bar{\mathbf{X}}^t$. Note that $\mathop{\mathbb{E}}_{\lambda}[\lambda] = \frac{1}{2}$ for $\lambda \sim \text{Beta}(\alpha, \alpha)$. Taking expectation w.r.t. λ from both sides, one has

$$\mathop{\mathbb{E}}_{\lambda}[\|f^n(\widehat{\mathbf{X}}^t) - \mathbf{X}^{t+n}\|] \leq \frac{1}{2} L_f^n \|\widehat{\mathbf{X}}^t - \mathbf{X}^t\| + \frac{L_f^n - 1}{L_f - 1}\epsilon.$$

For proposition (3), by recursively using the triangle inequality,

$$\|f^n(\widehat{\mathbf{X}}^t) - f^n(\bar{\mathbf{X}}^t)\| \leq (1-\lambda) L_f^n \|\widehat{\mathbf{X}}^t - \mathbf{X}^t\|.$$

Again, by taking expectation w.r.t. λ from both sides, one has

$$\mathop{\mathbb{E}}_{\lambda}[\|f^n(\widehat{\mathbf{X}}^t) - f^n(\bar{\mathbf{X}}^t)\|] \leq \frac{1}{2} L_f^n \|\widehat{\mathbf{X}}^t - \mathbf{X}^t\|.$$

$\square$

APPENDIX D EXPERIMENTS

A. Data Preprocessing

This subsection describes the supplementary details of data preprocessing. The raw data of NBA[1] and SDD[2] can be directly downloaded as they are public online. The raw data of H3D[3] along with necessary codes for object detection and tracking can be requested from the Honda Research Institute.

The raw data are preprocessed by ourselves according to the instructions of the original paper [2]. Regarding each game in NBA as a scene, all three datasets contain multiple scenes. For each scene, the trajectories are down-sampled at the rate of 2.5Hz. All types of agents are preserved and the trajectories of all agents within a given period (15 for NBA and H3D datasets, and 20 for the SDD dataset) are treated as a sample. The number of agents in a sample does not change over time, while the agent counts may vary in different samples. Non-overlapping trajectories from each scene constitute the training set, validation set and testing set with ratios 0.65, 0.10 and 0.25, respectively.

As shown in Table I, the numerical ranges of the 2D coordinates vary significantly across different datasets. To ensure numerical stability, the coordinates are normalized to $[-1, 1]$ via the min-max normalization, and they are recovered while making a prediction.

Context information like the top-down-view images and point cloud density maps are not included in the experiments. Incorporating the context information is left for future work.

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

[2]https://cvgl.stanford.edu/projects/uav_data/

[3]https://usa.honda-ri.com/h3d

TABLE A2
NETWORK STRUCTURES OF THE FUNCTIONS USED IN HIMRAE.

Function	Network Structure
$f_{\text{emb}}, f_v, f_e, \widetilde{f}_e$	[Linear, ELU, BatchNorm1d]×2
$f_Q, f_K, g_Q^{c_i}, g_K^{c_i}, g_V^{c_i}$	[Linear, Tanh]
f_V	[Linear, Tanh]×2
f_{proj}	[[Linear, ELU, BatchNorm1d]×2, Linear]
f_{out}	[[Linear, ReLU]×2, Linear]

TABLE A3
REGULARIZATION COEFFICIENT γ FOR GRAPH ENTROPY.

Dataset	Candidate Values for γ	Best γ	Best γ w/ mixup
NBA	$\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$	10^{-4}	10^{-5}
H3D	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$	10^{-3}	10^{-4}
SDD	$\{0.1, 1, 10, 10^2\}$	10	10

B. Implementation Details of HIMRAE

Most functions used in HIMRAE are single-layer perceptrons or MLPs. The network structure of these functions are listed in Table A2. Other implementation details of HIMRAE are listed as follows.

- Following Li *et al.* [2], two-layer GRUs are used in both the encoder and the decoder.
- The temperature parameter T in Eq. (8) is set to 0.5.
- The variance σ^2 is set to 1 for the distribution of $\widehat{\mathbf{X}}^t$.

C. Experimental Settings

- All experiments are run 5 times on a machine containing 128GB of RAM, and 8 NVIDIA TITANXP graphics cards with PyTorch 1.9 and CUDA 11.5 in Ubuntu 20.04.
- Graphs of different sizes are batched into a single graph with different connected components, which allows better usage of the memory and significantly reduces the training time.
- The dimensions of all hidden layers are set to 128.
- Our model is trained with a batch size of 128 for 200 epochs. The Adam optimizer is adopted with a learning rate of 10^{-3}.
- Empirically, the windows size τ is set to 5 for NBA and H3D datasets, and it is set to 4 for the SDD dataset. Experiments by Li *et al.* [2] showed that, a smaller τ leads to lower predicting errors, while more inference time is required. Therefore, an intermediate value can achieve a good trade-off.
- For the graph entropy, the regularization coefficient γ is chosen via cross-validation. The candidate set, the best γ, and the best γ used in HIMRAE w/ GE & mixup are listed in Table A3.
- For the mixup training, the parameter α is empirically initialized as 10 and decays by 0.5 in every 10 epochs. α controls the variance of the distribution Beta(α, α) for the mixing coefficient λ. When α is large, $\lambda \approx 1/2$, leading to an approximately equal-weighted mixture. In this case, the accumulated errors are more controllable. When α is small, more choices are allowed for λ, and the errors are

TABLE A4
STATISTICAL SIGNIFICANCE TEST FOR MINIMUM ADES/FDES.

Datasets	NBA		H3D		SDD	
Units	Meters		Meters		Pixels	
Metrics	minADE	minFDE	minADE	minFDE	minADE	minFDE
EvolveGraph	0.12	0.21	0.58	1.00	20.9	39.8
HIMRAE w/ GE & mixup	**0.09**	**0.17**	**0.35**	**0.50**	**18.8**	**36.4**
p-values	2.22×10^{-4}	1.84×10^{-5}	5.51×10^{-11}	4.50×10^{-10}	1.09×10^{-5}	1.23×10^{-4}

TABLE A5
STATISTICAL SIGNIFICANCE TEST FOR MEAN ADES/FDES.

Datasets	NBA		H3D		SDD	
Units	Meters		Meters		Pixels	
Metrics	minADE	minFDE	minADE	minFDE	minADE	minFDE
EvolveGraph	0.19	0.34	0.81	1.49	28.3	54.0
HIMRAE w/ GE & mixup	**0.15**	**0.28**	**0.56**	**0.99**	**25.2**	**49.4**
p-values	9.75×10^{-4}	1.88×10^{-4}	5.95×10^{-12}	1.23×10^{-10}	1.13×10^{-8}	5.27×10^{-8}

TABLE A6
MEAN ADES/FDES (METERS) OF HETEROGENEOUS AGENTS ON THE H3D DATASET.

Type	Car	Pedestrian	Truck	Cyclist	Other vehicle	Bus	Motorcyclist	Animal
Percentage (%)	50.05	36.62	8.93	2.36	1.43	0.32	0.26	0.03
HIMRAE$_{\text{HOMO}}$	0.75/1.44	0.69/1.23	0.73/**1.33**	0.92/1.64	0.71/1.34	1.32/1.63	**1.14/2.17**	0.19/0.28
HIMRAE	**0.73/1.39**	**0.66/1.16**	**0.72**/1.34	**0.79/1.14**	**0.69/1.19**	**0.76/1.14**	1.32/2.68	**0.15/0.20**

TABLE A7
MEAN ADES/FDES (PIXELS) OF HETEROGENEOUS AGENTS ON THE SDD DATASET.

Type	Pedestrian	Biker	Car	Cart	Skater	Bus
Percentage (%)	65.87	17.29	14.85	0.73	0.66	0.60
HIMRAE$_{\text{HOMO}}$	23.8/43.7	67.9/130.7	10.7/18.6	**60.1/112.5**	**69.6/132.8**	69.6/132.8
HIMRAE	**23.3/42.7**	**62.7/122.4**	**9.5/17.0**	62.9/116.5	77.1/143.3	**19.7/34.2**

TABLE A8
MEAN ADES/FDES (METERS) OF HETEROGENEOUS AGENTS ON THE NBA DATASET.

Type	Ball	Home Team Player	Visiting Team Player
Percentage	1/11	5/11	5/11
HIMRAE$_{\text{HOMO}}$	0.44/0.81	0.15/0.26	0.15/0.26
HIMRAE	**0.43/0.79**	**0.13/0.24**	**0.13/0.23**

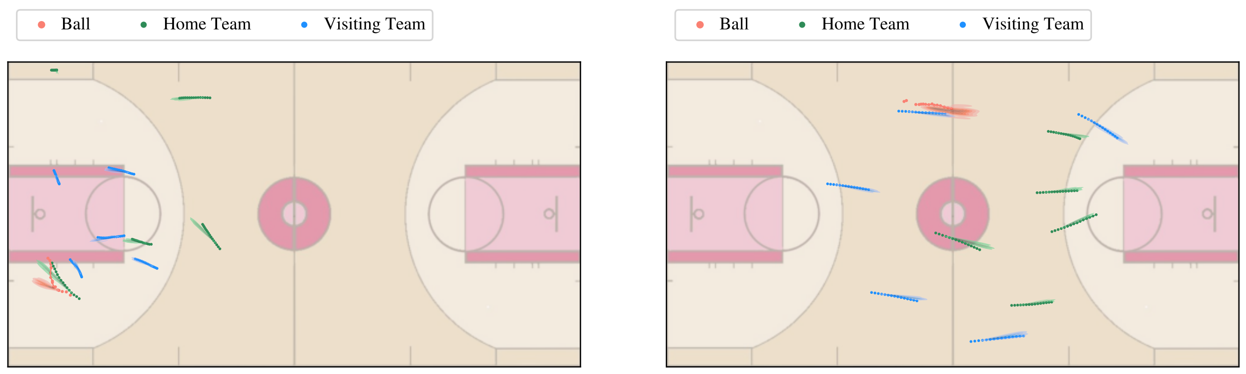
less controllable, which increases the training difficulty. By starting with a relatively large α and decreasing it linearly, the model progressively learns to reduce the accumulated errors. Other choices of α's initial value and more designs for the decaying strategy may be explored to further improve the performance.

D. Statistical Significance Test

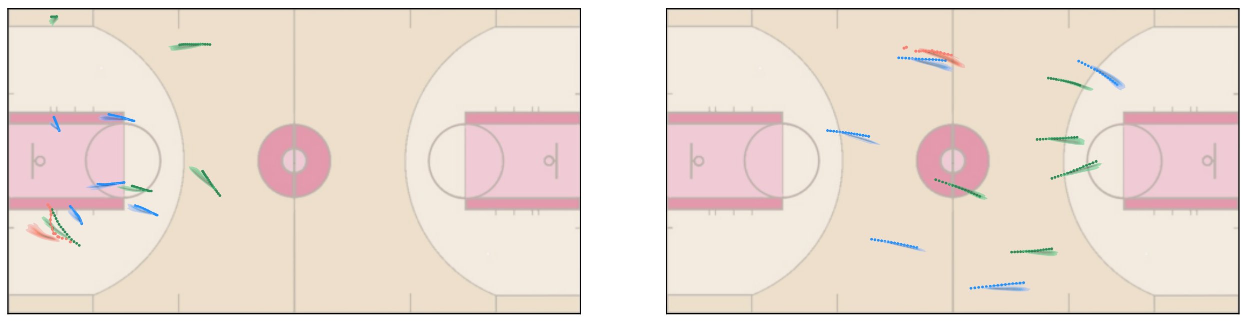
Statistical significance tests are conducted on all datasets by comparing HIMRAE w/ GE & mixup and EvolveGraph, the best baseline overall. The p-values of minimum ADEs/FDEs and mean ADEs/FDEs are reported in Table A4 and Table A5, respectively. $p < 0.01$ indicates that two distributions are significantly different.

E. Category Specific ADEs/FDEs

To further demonstrate the effectiveness of HAM, this paper reports the mean ADEs/FDEs of HIMRAE and HIMRAE$_{\text{HOMO}}$ on heterogeneous agents. The results on H3D, SDD and NBA datasets are shown in Table A6, Table A7 and Table A8, respectively. The results show that the predicting errors for almost all types of agents decrease by using HAM. On the H3D dataset, the ADEs/FDEs of some minor classes like cyclists, buses and animals decrease more significantly, which shows that the fine-grained interactions learned by HAM can help to make better predictions for rare classes. Similar results are shown on the SDD dataset except that the predicting errors for carts and skaters are larger. On the NBA dataset, the predicting error for the ball is significantly larger than that of a player. Maybe the ball moves with larger uncertainty, and



(a) HIMRAE w/ GE & mixup



(b) EvolveGraph

Fig. A1. Visualization of the predicted trajectories on the NBA dataset. Each column corresponds to the results on a single sample. The trajectories of the ball, home team players and visiting team players are in orange, green and blue, respectively. Historical trajectories are in dots, ground truth trajectories to be predicted are in solid lines, while the predicted trajectories are visualized using the kernel density estimation.)

its trajectory is harder to predict.

F. Visualizing the Predicted Trajectories

This subsection visualizes the predicted trajectories of our method and EvolveGraph, a closely related work for heterogeneous multi-agent trajectory prediction. Randomly selected samples from NBA, H3D and SDD datasets are visualized in Fig. A1, Fig. A2 and Fig. A3, respectively.

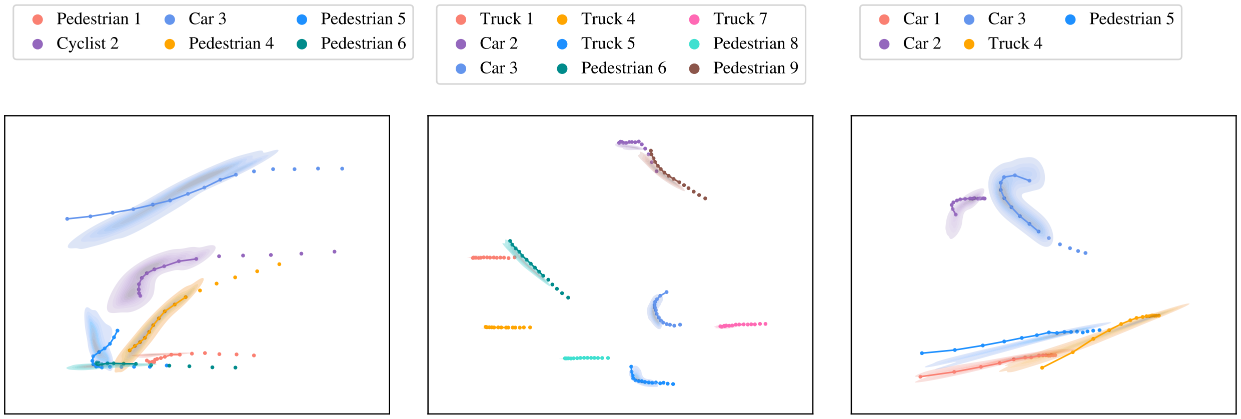
As shown in Fig. A1, our method can better predict the moving directions of both the ball and the players. In both cases, EvolveGraph fails to make accurate predictions for the ball that generally moves with large uncertainty. Consequently, the predicted trajectories of players around the ball suffer from a larger deviation than our predictions. Maybe our method can distinguish the ball-player interaction and the player-player interaction, and make more accurate predictions.

The trajectories on the H3D dataset and the SDD dataset are more complex since more types of traffic participants are involved. Compared with EvolveGraph, the predictions of our method can better cover the target trajectories of all types of agents. Specifically, in the third case of Fig. A2, our method successfully predicts the turning behavior of Car 3. Nonetheless, in the second case of Fig. A3, the predictions of both methods for Biker 7 suffer from a large deviation from the ground truth. Such a case is difficult to handle since the sudden change of intention can be hardly inferred from only the historical trajectories.

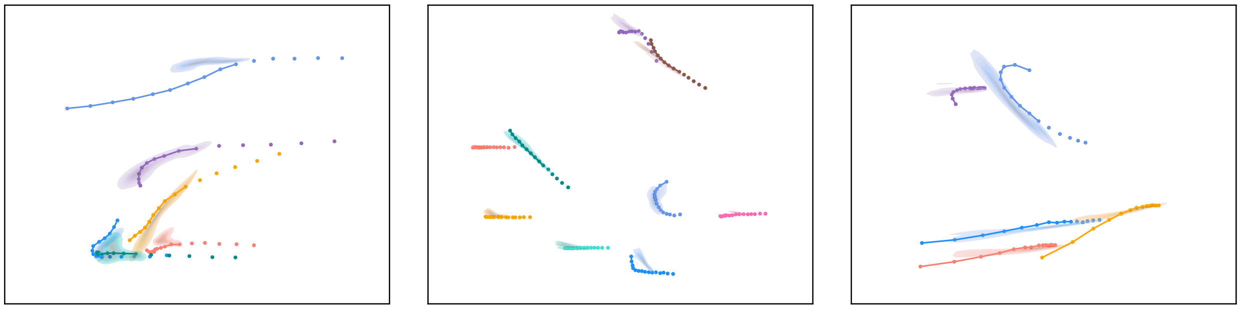
References

[1] A. W. Marshall, I. Olkin, and B. C. Arnold, *Inequalities: Theory of Majorization and Its Applications*. Springer New York, NY, 2011, ch. 4, pp. 156–157.

[2] J. Li, F. Yang, M. Tomizuka, and C. Choi, "EvolveGraph: Multi-agent trajectory prediction with dynamic relational reasoning," in *Proc. NeurIPS*, 2020, pp. 19 783–19 794.

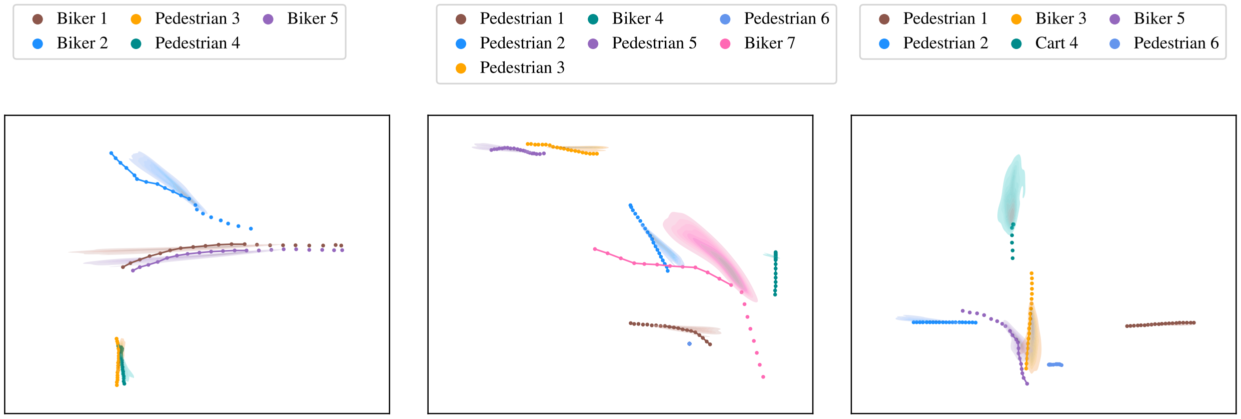


(a) HIMRAE w/ GE & mixup

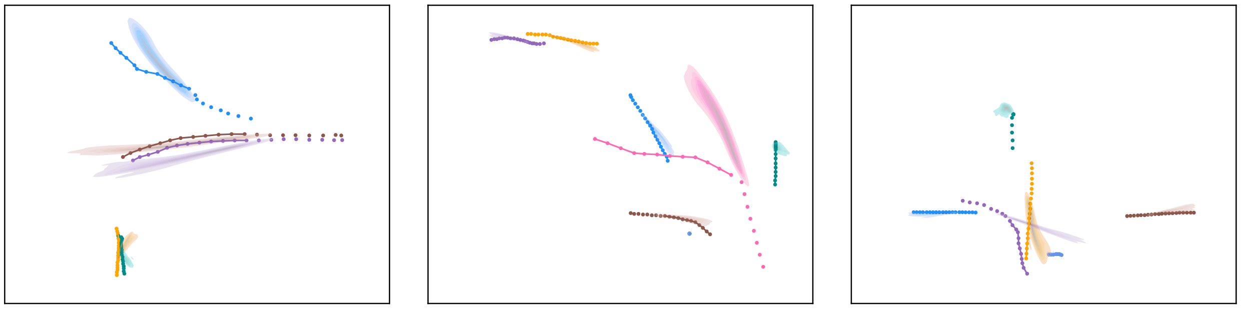


(b) EvolveGraph

Fig. A2. Visualization of the predicted trajectories on the H3D dataset. Each column corresponds to the results on a single sample. Historical trajectories are in dots, ground truth trajectories to be predicted are in solid lines, while the predicted trajectories are visualized using the kernel density estimation.



(a) HIMRAE w/ GE & mixup



(b) EvolveGraph

Fig. A3. Visualization of the predicted trajectories on the SDD dataset. Each column corresponds to the results on a single sample. Historical trajectories are in dots, ground truth trajectories to be predicted are in solid lines, while the predicted trajectories are visualized using the kernel density estimation.