2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Holistic Graph-based Motion Prediction

Daniel Grimm[1], Philip Schörner[1], Moritz Dreßler[2] and J.-Marius Zöllner[1,2]

***Abstract*— Motion prediction for automated vehicles in complex environments is a difficult task that is to be mastered when automated vehicles are to be used in arbitrary situations. Many factors influence the future motion of traffic participants starting with traffic rules and reaching from the interaction between each other to personal habits of human drivers. Therefore, we present a novel approach for a graph-based prediction based on a heterogeneous holistic graph representation that combines temporal information, properties and relations between traffic participants as well as relations with static elements such as the road network. The information is encoded through different types of nodes and edges that both are enriched with arbitrary features. We evaluated the approach on the INTERACTION and the Argoverse dataset and conducted an informative ablation study to demonstrate the benefit of different types of information for the motion prediction quality.**

I. Introduction

Machine learning has improved in recent years and excels in domains where it is hard to find an explicit mathematical description of the solution. In autonomous driving machine learning led to great improvements in perception tasks. However, driving in crowded scenes remains challenging for autonomous vehicles (AVs), mainly because the motion prediction becomes harder due to the increasing number of possible interactions among the traffic participants while paying attention to the road. This problem is not restricted to autonomous driving and can easily be transferred to other use cases where autonomous systems interact and share their space with humans, e.g., a logistic robot in a warehouse. In this work we focus on motion prediction for AVs.

In recent works Jia et al. [1] and Mo et al. [2] solve the spatio-temporal characteristic of the problem in a two staged fusion approach. Firstly, dynamic information, i.e. the past trajectory of the traffic participants, is fused over time. Secondly, information is shared between the traffic participants and the road. Yuan et al. [3] propose a simultaneous temporal and spacial fusion of the past trajectories using a masked transformer. This allows to capture the dynamic context at a higher resolution. However, map information is modelled as a birds eye view image and processed via a convolutional neural network (CNN). This is not optimal, because each agent should process only surrounding road elements and not the complete map. Works such as Jia et al. [1] and Liang et al. [4] model the map as a graph allowing traffic participants to only attend to areas of the map where they are currently driving. Those approaches use the aforementioned staged fusion approach, which motivates the problem of finding a graph representation that incorporates the spatio-temporal information from traffic participants and the environment. Thus, we propose a heterogeneous graph for simultaneous attention to past history, other agents' time-discrete trajectory and map information without using pre-fused data. The contribution of this paper includes:

- Holistic heterogeneous graph: Formulating the problem as a graph without pre-fused data makes it possible to capture interactions at a higher resolution.
- Modularity: More opportunities to encode expert-knowledge in the graph via edges and their features. The modular construction of the graph also allows for further extensions in the future.
- Benchmarking: INTERACTION and Argoverse dataset

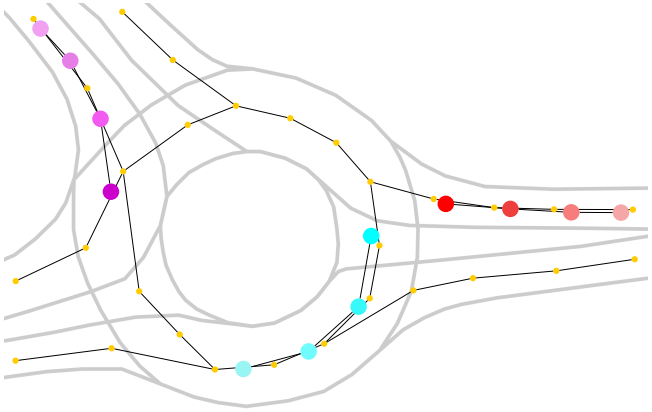


Fig. 1: Nodes in the heterogeneous graph. Map-nodes are depicted in yellow. Different colored nodes represent agent-nodes where nodes of the same color belong to the same trajectory. Time context is visualized with color fading. The high definition map (HD-Map) is depicted in light gray for better understanding of the traffic scene.

II. Related Work

Motion prediction is an ongoing research topic in the field of autonomous driving. In this section we provide an overview regarding graph neural networks (GNN) and motion prediction. As we are pursuing a learned prediction approach, we are focussing on learning-based approaches for motion prediction.

A. Graph Neural Networks

Graph neural networks are used to extract information from data which can be structured in graphs. For homogeneous graphs, there exist a wide variety of operations to exchange information between nodes, e.g. GCN [5],

[1] FZI Research Center for Information Technology, 76131 Karlsruhe, Germany. daniel.grimm, schoerner, zoellner@fzi.de
[2] Karlsruhe Institute of Technology (KIT), Germany.

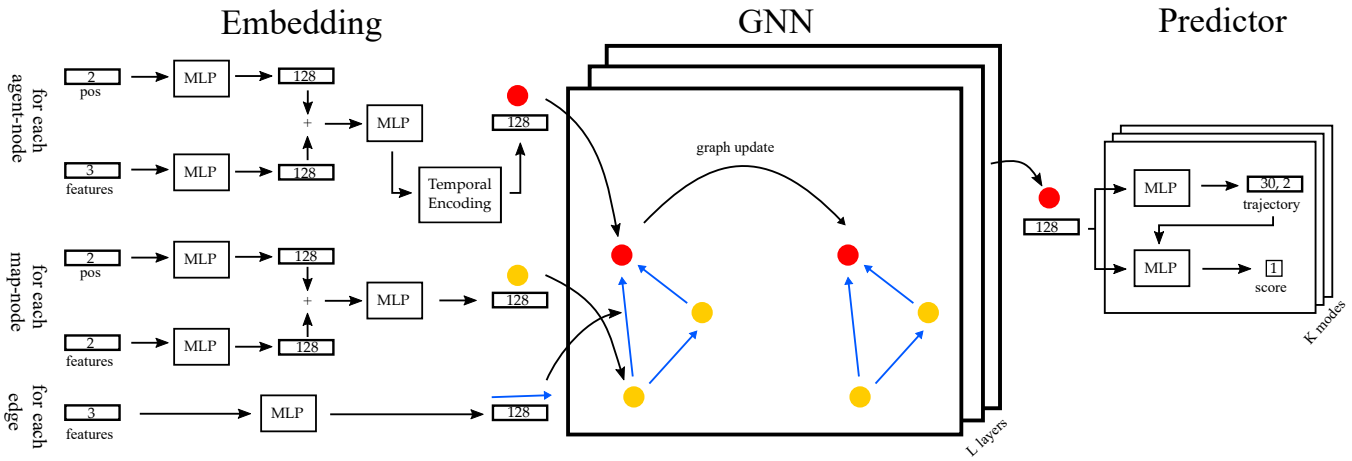


Fig. 2: Proposed concept. Inputs are embedded in separate embedding modules. Heterogeneous GNN is used to generate latent representation of all agents in the scene. Prediction head outputs a future trajectory for each agent.

GraphSAGE [6], GAT [7] and Gatv2 [8], each following the message passing scheme [9] to update the nodes in the graph. Most previous works, such as [5], [7] focus on homogeneous graphs. This is not sufficient in the field of motion prediction, where different entities, e.g., traffic participants and map elements, interact. Heterogeneous graphs consist of different node and edge types [10]. Wang et al. [11] propose to model attention in a heterogeneous graph in a two stage approach called Node-Level Attention and Semantic-Level Attention. Hu et al. [12] introduces ideas of a Transformer [13] in a heterogeneous graph. The attention matrix is calculated dependent on the edge type and node type. However, edge features are not considered.

B. Motion Prediction

The task of motion prediction is mostly formulated as a seq2seq problem. Therefore, early motion prediction models, such as Social LSTM [14] from Alahi et al., PRECOG [15] and R2P2 [16] from Rhinehart et al., rely on Recurrent Neural Network (RNN) structures, such as LSTM [17] or GRU [18]. With the success of CNNs in the domain of image classification, such as Krizhevsky et al. [19] and Simonyan et al. [20], it became possible to use a 2D birds eye view (BEV) image of the street layout in motion prediction. Hong et al. [21], Phan-Minh et al. [22] and Djuric et al. [23] encode a rich representation of the environment including road elements, dynamic context and other traffic participants in the image. Due to the success of Transformer [13] in Natural Language Processing, which is also a seq2seq problem, works, such as Ngiam et al. [24] and Mercat et al. [25] adopted the attention mechanism for motion prediction. Yuan et al. [3] combine attention over time and over other agents in one Transformer called AgentFormer. Attention is done in a fully connected fashion not regarding spatial distance between agents. To the authors knowledge, VectorNet from Gao et al. [26] and LaneGCN from Liang et al. [4] were the first models, to use a GNN for motion prediction. VectorNet uses a local graph to obtain polyline-level features for agent trajectories and lanes. Afterwards these features are used in a global interaction graph, which is fully connected, undirected and homogeneous. In contrast to Vectornet, LaneGCN uses a segment of a polyline as a node in their lane graph, hence, capturing the map at a higher resolution. DenseTNT by Gu et al. [27] adopt Vectornet and split the prediction task into goal prediction and trajectory completion. The map-nodes in the heterogeneous graph proposed in our model use a similar map representation as LaneGCN. HEAT from Mo et al. [2] and HDGT from Jia et al. [1] propose a heterogeneous interaction graph, where the nodes represent higher-level features, such as agent trajectories or lanes. HEAT constructs the street layout with a CNN from BEV images. HDGT uses a simplified PointNet [28] to encode lane features from a vectorized format. Sheng et al. [29], and Cao et al. [30] introduce a graph-based spatial-temporal convolution. Our proposed heterogeneous graph differs from above mentioned works by the differently modelled temporal information. Instead of fusing temporal information outside of the graph such as HEAT [2] and HDGT [1], or using a separate graph for each time-step such as Sheng et al. [29] and Cao et al. [30], we combine time variant information, e.g. agent trajectories, in one graph. Time information is preserved by the usage of a temporal encoding, see Sec. III-A. To the knowledge of the authors, we are the first to model the whole encoding step in a single graph for the task of motion prediction.

III. Concept

The general pipeline is depicted in Fig. 2. The model consists of an embedding part followed by an encoder-decoder structure. As encoder, we propose a spatio-temporal static heterogeneous graph, which includes encoding the past trajectory as well as social context attention and the encoding of the street layout. The graph yields a latent feature vector per agent. The decoder is a normal Multilayer Perceptron (MLP) that outputs multi-modal predictions for each agent in the scene. We use a scene-centric data representation.

A. Embedding

Firstly, agent-nodes, map-nodes and edge features are embedded to a higher dimension f using a set of MLPs each with a linear layer followed by ReLU Activation and Layer-Normalization. A detailed view of the embedding process is depicted in Fig. 2. In order to represent the timestamp of an agent-node $\boldsymbol{a}_i^t$, a temporal encoding $\boldsymbol{\tau}$ which is similar to the positional encoding in Transformers [13] is added to the agent-nodes in the last step of the embedding.

$$\tau(t, 2i) = \sin\left(t/10000^{\frac{2i}{f}}\right) \tag{1}$$

$$\tau(t, 2i+1) = \cos\left(t/10000^{\frac{2i+1}{f}}\right) \tag{2}$$

$$\boldsymbol{a}_i^t = \boldsymbol{W}_1\left(\boldsymbol{a}_i^t \parallel \boldsymbol{\tau}(t)\right) \tag{3}$$

where $\tau(t, 2i)$ and $\tau(t, 2i+1)$ refer to the even resp. odd index of feature dimension in $\boldsymbol{\tau}(t)$ and t refers to id of the time-step, e.g., $t \in \{0, 1, ..., 9\}$ in the INTERACTION dataset.

B. Hetero GNN

The heterogeneous graph is defined as $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of edges with their corresponding edge features. A scene consists of traffic participants, hereafter referred to as agents, and the HD-Map. In this work the set of nodes $\mathcal{N} = \{\mathcal{A}, \mathcal{M}\}$ consist of two types:

- The set of agent-nodes $\mathcal{A}$, where a single agent-node $\boldsymbol{a}_i^t$ refers to a measurement at time-step t of the observed past trajectory of the i-th agent and consists of the current position, velocity and orientation, so that, $\boldsymbol{a}_i = (x_i, y_i, v_{x_i}, v_{y_i}, h_i)^\intercal$.
- The set of map-nodes $\mathcal{M}$, where a single map-node $\boldsymbol{m}_i$ refers to a segment of a centerline of the vectorized HD-Map consisting of direction and position, hence, $\boldsymbol{m}_i = (x_i, y_i, \Delta x_i, \Delta y_i)^\intercal$.

The set of the different directed edge types $\mathcal{E}$ of the heterogeneous graph can be seen in Fig. 3. The connections of a specific edge type from node type j to node type i with relation r are stored in the adjacency matrix $\boldsymbol{A}_{j,r,i}$ and $\boldsymbol{e}_{j,r,i}$ denotes the corresponding edge features. The edge features for each relation consist of the relative Cartesian coordinates between two connected nodes.

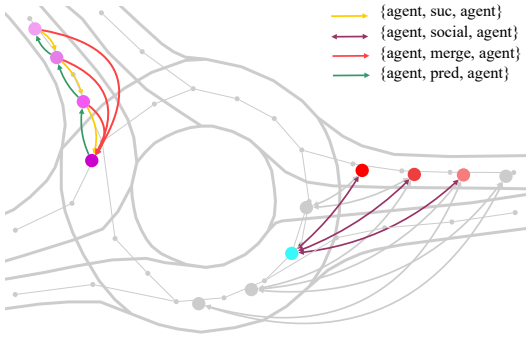
To update the node features $\boldsymbol{x}_{i,r}^{(l)} \in \mathbb{R}^F$ of node i in Layer l for a specific edge type a basic message passing scheme introduced by Fey et al. [9] is used.

$$\hat{\boldsymbol{x}}_{i,r}^{(l)} = \gamma_r^{(l)}\left(\boldsymbol{x}_i^{(l-1)}, \sum_{j \in \mathcal{N}(i)} \phi_r^{(l)}\left(\boldsymbol{x}_i^{(l-1)}, \boldsymbol{x}_j^{(l-1)}, \boldsymbol{e}_{j,r,i}^{(l-1)}\right)\right) \tag{4}$$

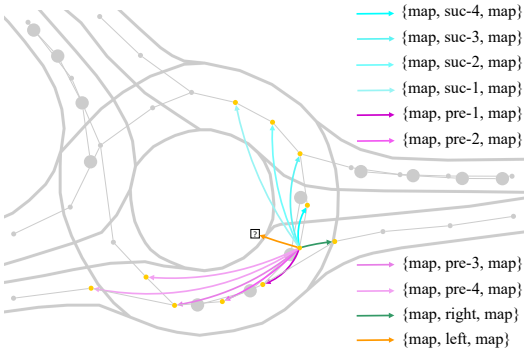
A MLP $\phi^{(l)}$ is used to calculate the messages of the neighboring nodes $\boldsymbol{x}_j$ while also using the edge features $\boldsymbol{e}_{j,r,i}$ from the edge connecting the corresponding node j to node i with relation r. The neighboring nodes are determined by the associated adjacency matrix. In Eq. 4 the messages are aggregated using a sum. The edge type specific update $\hat{\boldsymbol{x}}_{i,r}^{(l)}$ is calculated by another MLP $\gamma^{(l)}$. Afterwards all edge type specific node updates are merged by a sum followed by ReLU Activation, Residual-Connection and a Layer Normalization. to get the output of layer l:

$$\boldsymbol{x}_i^{(l)} = \text{norm}\left(\text{ReLU}\left(\sum_{r \in \mathcal{E}(i)} \hat{\boldsymbol{x}}_{i,r}^{(l)}\right) + \boldsymbol{x}_i^{(l-1)}\right) \tag{5}$$

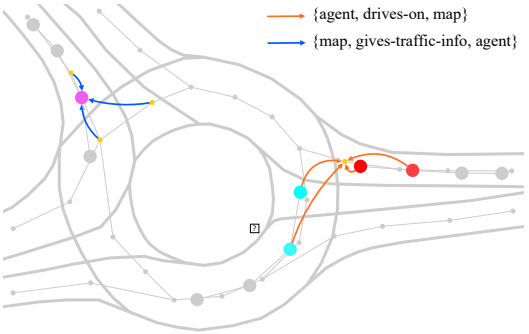
In the proposed heterogeneous graph, not every edge is used simultaneously to update the nodes. Instead, edges between the same types of nodes are used first to generate more meaningful node features. Afterwards all edges in the graph



(a) Edges among agent-nodes. Time-step information is presented with color shading. Agent-nodes belonging to one agent trajectory are presented on the left. Social context is visualized on the right.



(b) Edges among map-nodes. For a better view, only the edges from one map-node are depicted.



(c) Edges between agent-nodes and map-nodes. For a better view, one agent-node is depicted as destination on the left and on the right one map-node is selected as destination.

Fig. 3: Overview of the edges used by the heterogeneous GNN.

except {agent, merge, agent} are used to further update node features and fuse context between agents and the map.{agent, merge, agent} is used to get the output of the GNN, i.e., a latent feature vector for each agent in the scene.

1) Map Context: Map-nodes are connected to other map-nodes using the previous, successive, left and right neighbour according to the driving direction of the lane, e.g., {map, left, map} for the left neighbour. During message passing it is preferable to propagate information along the road-direction rather than perpendicular to it because most road users travel along the road and not across. We accomplish this by adding new edges along the road connecting a map node with its *i*-th predecessor respectively successor using the *i*-th power of the corresponding adjacency, e.g., {map, pre-2, map}. A detailed view of the edges between map-nodes is given in Fig. 3b and is similar to LaneGCN [4]. For message generation we propose an extension to the basic GCN-Conv [5]. We include the usage of edge features in the message generation ϕ resulting in the node updates $\hat{\boldsymbol{x}}_{i,r}^{(l)}$,

$$\hat{\boldsymbol{x}}_{i,r}^{(l)} = \sum_{j \in \mathcal{N}(i)} \frac{1}{\sqrt{\deg(i)}\sqrt{\deg(j)}} \left(\left(\boldsymbol{x}_j^{(l-1)} + \boldsymbol{e}_{j,r,i}^{(l-1)} \right) \boldsymbol{W} \right) + \boldsymbol{b} \tag{6}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ refer to learnable parameters. Edge and node features are added together, which reduces the number of learnable parameters without decreasing performance, see Seq. IV-C. To gather a good encoding of the HD-Map Data we use five layers, where each layer is constructed according to Eq. 5.

2) Agent Context: An agent-node is connected to its predecessor and successor belonging to the past trajectory of the agent. The corresponding edges are named {*agent, pre, agent*} and {*agent, suc, agent*}. For social context {*agent, social, agent*}, every agent-node is connected to agent-nodes of the previous, same and future timestamp that belong to other agents. The respective edges are shown in Fig. 3a. Updating the agent-nodes is similar to the map-nodes. In order to pass information from the first to the last agent-node of an agents trajectory, the number of used layers n corresponds to the number of time-steps of the past trajectory, e.g., Argoverse: $n = 20$, INTERACTION: $n = 10$. Messages are generated using Eq. 6. Social context is added during the last two layers with a multi head graph attention module (GATv2) [8]. Therefore, edges of type {*agent, social, agent*} are used. The node updates via GATv2 [8] for relation r are given by

$$\hat{\boldsymbol{x}}_{i,r}^{(l)} = \alpha_{i,i}^{r} \boldsymbol{x}_i^{(l-1)} \boldsymbol{W}_1 + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}^{r} \boldsymbol{x}_j^{(l-1)} \boldsymbol{W}_2 \tag{7}$$

where the attention coefficients $\alpha_{i,j}$ are calculated with the learnable parameters $\boldsymbol{w}$ and $\boldsymbol{W}_{i \in \{1,2,3\}}$ as:

$$\alpha_{i,j} = \text{softmax}\left(\text{LeakyReLU}\left(\left[\boldsymbol{x}_i \parallel \boldsymbol{x}_j \parallel \boldsymbol{e}_{j,r,i}\right] \boldsymbol{W}_3\right) \boldsymbol{w}\right) \tag{8}$$

3) Context fusion: To properly fuse the HD-Map with the past trajectories of the agents, we use edges of type {*agent, drives-on, map*} and {*map, gives-traffic-info, agent*}. These two edges use a multi head GATv2 [8] module. The source nodes are selected based on the euclidean distance d_{th} of the 2d-position to the target nodes. d_{th} is dynamically calculated using the velocity of the agent-nodes and a threshold time t_{th}. This compensates for faster moving agents. Furthermore, we include the edges introduced in Seq. III-B.2 and Seq. III-B.1. In total, we use two fusion layers.

The latent representation of an agents past trajectory is spread out between all agent-nodes belonging to this specific agent, hence, for every agent, the agent-node at t_{obs} is selected as final feature vector and updated by a multi head GATv2 [8] module using edges from the past agent-nodes of that agent. Fig. 3a shows the edges {*agent, merge, agent*} for this purpose.

C. Motion-Prediction Head

We use a combination of regression and scoring in separate MLPs to generate K possible trajectories per agent. For each mode, a new regression and classification MLP is instantiated. Input is the latent feature vector for each agent. To calculate the trajectory score we also use the predicted trajectory. The two MLPs are similar and consist of a linear layer with a residual connection, ReLU Activation, Layer Normalization; followed by another linear layer. The model outputs the predicted trajectories $\boldsymbol{\mathcal{T}}$ of shape $[A, K, T_f, 2]$ and the scores $\boldsymbol{s}$ of shape $[A, K]$, where A is the number of agents and T_f equals the number of predicted time-steps.

D. Loss

The Loss $\mathcal{L}$ consists of a regression Loss and a classification Loss.

$$\mathcal{L} = \mathcal{L}_{reg} + \lambda \mathcal{L}_{cls} \tag{9}$$

A smooth L1 Loss is used as regression Loss $\mathcal{L}_{reg}$. To prevent mode collapse, $\mathcal{L}_{reg}$ is only calculated for the mode k_{min} with minimal final displacement error (FDE) to the ground truth.

$$\mathcal{L}_{reg} = \frac{1}{AT_f} \sum_{a}^{A} \sum_{t}^{T_f} \sum_{n \in \{x,y\}} \text{smoothL1}\left(\mathcal{T}_{a,n}^{t,k_{\text{min}}}, \hat{\mathcal{T}}_{a,n}^{t}\right) \tag{10}$$

with

$$\text{smoothL1}(x,y) = \begin{cases} 0.5 * (x-y)^2, & \text{if} |x-y| < 1 \\ |x-y| - 0.5, & \text{otherwise} \end{cases} \tag{11}$$

where $\hat{\boldsymbol{\mathcal{T}}}$ refers to the ground truth. The classification loss $\mathcal{L}_{cls}$ is a max-margin loss [31] with margin m.

$$\mathcal{L}_{cls} = \frac{1}{A(K-1)} \sum_{a}^{A} \sum_{k \neq k_{\text{min}}}^{K} \max\left(0, s_{a,k} + m - s_{a,k_{\text{min}}}\right) \tag{12}$$

IV. Evaluation

In the following, we evaluate our model on the INTERACTION dataset [32] and the Argoverse motion forecast dataset [33]. Firstly, we introduce the datasets, the evaluation metrics and the used hyperparameter settings. Afterwards, we conduct ablation studies on the architecture and finally compare our model to the state-of-the-art.

A. Experimental Settings

The Argoverse Motion Forecast Dataset is a large scale collection of 323557 samples, each with a duration of 5 s, resulting in a total of 320 h. The data was collected in Miami and Pittsburgh with 10 Hz. The task is to predict the future locations of one agent for 3 s, given its history of 2 s. HD-Map data is provided in an argoverse specific format.

The INTERACTION Dataset is a highly interactive dataset recorded in 5 different locations including roundabouts, merging scenarios and intersections in Germany, USA and China. It consists of around 16.5 h of data including 40054 trajectories sampled at 10 Hz. The task is to predict the future locations of all agents in the scene for 3 s, given their history for 1 s. The HD-Map data is provided using the Lanelet2 [34] format.

To evaluate the results quantitatively on Argoverse, we use its suggested metrics: Minimum Average Displacement Error (minADE), see Eq. 13, Minimum Final Displacement Error (minFDE), see Eq. 14, and Minimum Miss Rate (minMR). Latter denotes the percentage of predictions having a minFDE greater than 2m. $\|\cdot\|^2$ refers to L2-Norm.

$$\text{minADE} = \frac{1}{AT_f} \sum_{a}^{A} \sum_{t}^{T_f} \min_{k}^{K} \left\| \mathcal{T}_a^{t,k} - \hat{\mathcal{T}}_a^{t} \right\|^2 \tag{13}$$

$$\text{minFDE} = \frac{1}{A} \sum_{a}^{A} \min_{k}^{K} \left\| \mathcal{T}_a^{T_f,k} - \hat{\mathcal{T}}_a^{T_f} \right\|^2 \tag{14}$$

For multi-modal predictions, minFDE refers to the minimum euclidean distance of the predicted trajectory and the ground truth at the prediction horizon T_F over all modes. minADE is defined as the euclidean distance between the ground truth and the predicted positions averaged by time. minMR indicates the ratio of the predictions where the final position of the trajectory of the best mode is more than a certain threshold, usually 2 m, away from the ground truth. On INTERACTION, we its proposed metrics Minimum Joint Average Displacement error (minJADE), see Eq. 15, Minimum Joint Final Displacement Error (minJFDE), see Eq. 16, and Minimum Joint Miss Rate (minJMR) as metrics to measure the performance of joint motion prediction. Latter denotes the percentage of predictions having a minJFDE greater than 2m.

$$\text{minJADE} = \min_{k}^{K} \frac{1}{AT_f} \sum_{a}^{A} \sum_{t}^{T_f} \left\| \mathcal{T}_a^{t,k} - \hat{\mathcal{T}}_a^{t} \right\|^2 \tag{15}$$

$$\text{minJFDE} = \min_{k}^{K} \frac{1}{A} \sum_{a}^{A} \left\| \mathcal{T}_a^{T_f,k} - \hat{\mathcal{T}}_a^{T_f} \right\|^2 \tag{16}$$

Training on each dataset was done on a RTX 3080 GPU for 40 epochs starting with an initial learning rate of 1e-3 and a decay of 0.5 every fifth epoch. We used the Adam [35] optimizer, a batch size of 8 and a weight decay of 0.5% for all weights which are not part of a normalization layer. All attention modules have four heads and the results of the heads are concatenated. Training took 33 h on Argoverse and 8 h on INTERACTION. For each scene in Argoverse [33], we set the position of the last time-step of the ego-agent as the origin of the local fixed coordinate system. In INTERACTION [32] the origin of a sample is set to the geometric center point of all the trajectories in the sample. For both datasets, we use a square with size of 160 m x 160 m centered around the origin to determine the relevant lanes and agents for the graph.

B. Results

Tab. I shows the results of our model in comparison to state-of-the-art approaches on the Argoverse and INTERACTION dataset. We achieve similar results as the state-of-the-art while having only 2.5 Mio parameters. In comparison to DenseTNT, HoliGraph predicts all traffic participants at once instead of only a single agent. HoliGraph has an inference time of around 77 ms on a RTX 3080, making it real-time capable. It can be assumed that the performance could be further increased by adding more semantic features to the graph, e.g., differentiate between road-bound and non-road-bound agents and adding traffic lights to the map-nodes.

TABLE I: Results on argoverse test, regular INTERACTION single and regular INTERACTION multi test dataset.

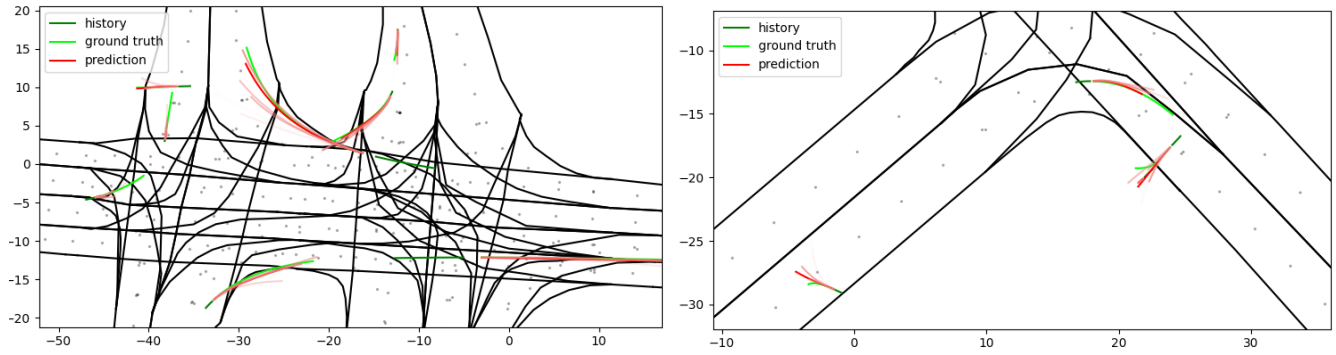
argoverse single	K=6 minADE	K=6 minFDE	K=6 minMR	No. of Parameters
HoliGraph (ours)	0.98 m	1.65 m	0.172	2.5 Mio
DenseTNT [27]	0.94 m	1.49 m	0.105	1.4 Mio
Scene Trans.[24]	0.80 m	1.23 m	0.13	15.3 Mio
LaneGCN [4]	0.87 m	1.36 m	0.16	3.6 Mio
interaction single	**K=6 minADE**	**K=6 minFDE**	**K=6 minMR**	**No. of Parameters**
HoliGraph (ours)	0.213 m	0.529 m	0.029	2.5 Mio
DenseTNT [27]	0.2819 m	0.6371 m	0.028	1.4 Mio
HDGT [1]	0.1085 m	0.3361 m	0.014	12 Mio
interaction multi	**K=6 minJADE**	**K=6 minJFDE**	**K=6 minJMR**	**No. of Parameters**
HoliGraph (ours)	0.362 m	1.043 m	0.138	2.5 Mio
ReCoG2 [36]	0.330 m	0.932 m	0.194	-
HDGT [1]	0.2162 m	0.7309 m	0.1384	12 Mio

C. Ablation Study

To investigate the effect of using different types of context information on the prediction accuracy, we conducted an ablation study. Tab. II shows that the performance on the INTERACTION validation dataset is increasing when providing the model with more context information. It also shows the importance of edge features to provide the model with additional relational information. In Tab. III we

TABLE II: Results on INTERACTION validation dataset for different types of context information as input. History means each agent is only connected to itself. Map means the usage of map-nodes. Social means that agents are also connected to other agents. Relational refers to the usage of edge features.

context information				K=6		
history	map	social	relational	minJADE	minJFDE	minJMR
✓				0.607 m	1.745 m	0.311
✓		✓		0.562 m	1.601 m	0.272
✓	✓			0.458 m	1.254 m	0.188
✓	✓	✓		0.441 m	1.212 m	0.178
✓	✓	✓	✓	0.362 m	1.043 m	0.138



(a) Dense traffic scene

(b) Pedestrian crossing

Fig. 4: Qualitative results on INTERACTION validation dataset. History is depicted in dark green, ground truth in light green, predictions in light red. The prediction with the highest score is depicted in red. The map-nodes are depicted as light red dots.

investigate the effect of residual connections during node update, the temporal encoding of agent-nodes and the way of including edge features. The residual connections improve the model performance by 11 % (mean over all metrics). Adding timestamp information directly to the agent-nodes with the temporal encoding from Eq. 3 further improves performance by 6 % (mean over all metrics). The third row in Tab. III refers to the architecture used in the final model which does not use the concatenation of edge features and node features. Thats because the concatenation only results in a small performance gain, but significantly increases the number of learnable parameters from 2.5 Mio to 4.2 Mio. Lastly we investigate the influence of different attention mechanisms. All three modules result in roughly the same number of learnable parameters, but the GATv2 module outperforms the other attention-modules.

TABLE III: Results on INTERACTION validation dataset for different architectures. Residual means residual connections during node update. Temporal refers to the temporal encoding of agent-nodes and concat refers to the concatenation of edge and node features.

architecture			K=6		
residual	temp	concat	minJADE	minJFDE	minJMR
			0.426 m	1.214 m	0.177
✓			0.383 m	1.090 m	0.151
✓	✓		0.362 m	1.043 m	0.138
✓	✓	✓	0.361 m	1.039 m	0.137

D. Qualitative Results

Some qualitative results are depicted in Fig. 4. Fig. 4a shows the performance of the model in a complex intersection with a lot of interactions between the traffic participants. In the scene are vehicles as well as pedestrians present. Our model is able to predict all agents well. For most agents, the lateral predictions are almost perfect. However, their longitudinal predictions show small deviations. On the right side of Fig. 4b a pedestrian is crossing the road. Nearly all modes indicate a light left turn. This is a result of the attention to the map-nodes, as the driving direction of the road is to the right. We will use this showcase as a motivation to distinguish in our future work between road-bound and non-road-bound users.

TABLE IV: Results on INTERACTION validation dataset for different attention mechanism.

attention modules			K=6		
GAT [7]	GATv2 [8]	Transformer [37]	minJADE	minJFDE	minJMR
✓			0.434 m	1.207 m	0.178
	✓		0.362 m	1.043 m	0.138
		✓	0.416 m	1.149 m	0.163

V. Conclusion

In this paper we have proposed a new way to represent temporal information in heterogeneous graphs for motion prediction. Instead of compressing the temporal information, we embed the whole past trajectories of all agents into the GNN. We achieve similar results as state-of-the-art approaches while having considerably less learnable parameters. We did an extensive ablation study to verify the effectiveness of each design decision. The evaluation was conducted on two different state-of-the-art datasets. As the holistic graph representation allows to include arbitrary information, we are going to further distinguish between road-bound agents such as cars, trucks and motorcycles and non-road-bound agents such as pedestrians.

Additionally, we plan to investigate the suitability of the HoliGraph representation for tracking tasks. The main problem of associating detected objects to tracks could be solved by learning the most probable edges, i.e., {agent, pre, agent}, connecting corresponding agent-nodes of the same track.

Acknowledgment

The research leading to these results was conducted within the project KIsSME (Artificial Intelligence for selective near-real-time recordings of scenario and maneuver data in testing highly automated vehicles) and was funded by the German Federal Ministry for Economic Affairs and Climate Action. Responsibility for the information and views set out in this publication lies entirely with the authors.

REFERENCES

[1] X. Jia *et al.*, *Hdgt: Heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding*, 2022.

[2] X. Mo, Y. Xing, and C. Lv, *Heterogeneous edge-enhanced graph attention network for multi-agent trajectory prediction*, 2021.

[3] Y. Yuan *et al.*, "Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 9813–9823.

[4] M. Liang *et al.*, "Learning lane graph representations for motion forecasting," in *Computer Vision – ECCV 2020*, A. Vedaldi *et al.*, Eds., Cham: Springer International Publishing, 2020, pp. 541–556, ISBN: 978-3-030-58536-5.

[5] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, 2016.

[6] W. L. Hamilton, R. Ying, and J. Leskovec, *Inductive representation learning on large graphs*, 2017.

[7] P. Veličković *et al.*, *Graph attention networks*, 2017.

[8] S. Brody, U. Alon, and E. Yahav, *How attentive are graph attention networks?* 2021.

[9] M. Fey and J. E. Lenssen, *Fast graph representation learning with pytorch geometric*, 2019.

[10] X. Wang *et al.*, "A survey on heterogeneous graph embedding: Methods, techniques, applications and sources," *IEEE Transactions on Big Data*, pp. 1–1, 2022.

[11] X. Wang *et al.*, "Heterogeneous graph attention network," in *The World Wide Web Conference*, ser. WWW '19, San Francisco, CA, USA: Association for Computing Machinery, 2019, 2022–2032, ISBN: 9781450366748.

[12] Z. Hu *et al.*, "Heterogeneous graph transformer," in *Proceedings of The Web Conference 2020*, ser. WWW '20, Taipei, Taiwan: Association for Computing Machinery, 2020, 2704–2710, ISBN: 9781450370233.

[13] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[14] A. Alahi *et al.*, "Social lstm: Human trajectory prediction in crowded spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 961–971.

[15] N. Rhinehart *et al.*, "Precog: Prediction conditioned on goals in visual multi-agent settings," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[16] N. Rhinehart, K. M. Kitani, and P. Vernaza, "R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] K. Cho *et al.*, *On the properties of neural machine translation: Encoder-decoder approaches*, 2014.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira *et al.*, Eds., vol. 25, Curran Associates, Inc., 2012.

[20] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2014.

[21] J. Hong, B. Sapp, and J. Philbin, "Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[22] T. Phan-Minh *et al.*, "Covernet: Multimodal behavior prediction using trajectory sets," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[23] N. Djuric *et al.*, "Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 2084–2093.

[24] J. Ngiam *et al.*, *Scene transformer: A unified architecture for predicting multiple agent trajectories*, 2021.

[25] J. Mercat *et al.*, *Multi-head attention for multi-modal joint vehicle motion forecasting*, 2019.

[26] J. Gao *et al.*, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[27] J. Gu, C. Sun, and H. Zhao, "Densetnt: End-to-end trajectory prediction from dense goal sets," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 15 303–15 312.

[28] C. R. Qi *et al.*, *Pointnet: Deep learning on point sets for 3d classification and segmentation*, 2016.

[29] Z. Sheng *et al.*, *Graph-based spatial-temporal convolutional network for vehicle trajectory prediction in autonomous driving*, 2021.

[30] D. Cao *et al.*, "Spectral temporal graph neural network for trajectory prediction," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1839–1845.

[31] J. Weston, C. Watkins, *et al.*, "Support vector machines for multi-class pattern recognition.," in *Esann*, vol. 99, 1999, pp. 219–224.

[32] W. Zhan *et al.*, *Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps*, 2019.

[33] M.-F. Chang *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[34] F. Poggenhans *et al.*, "Lanelet2: A high-definition map framework for the future of automated driving," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 1672–1679.

[35] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014.

[36] X. Mo, Y. Xing, and C. Lv, *Recog: A deep learning framework with heterogeneous graph for interaction-aware trajectory prediction*, 2020.

[37] Y. Shi *et al.*, *Masked label prediction: Unified message passing model for semi-supervised classification*, 2020.