FJMP: Factorized Joint Multi-Agent Motion Prediction

by

Luke Rowe

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2023

Author's Declaration

Abstract

Multi-agent motion prediction is an important problem in an autonomous driving pipeline, and it involves forecasting the future behaviour of multiple agents in complex driving environments. Autonomous vehicles (AVs) should produce accurate predictions of future agent behaviour in order to make safe and informed plans in safety-critical scenarios. Importantly, AVs should generate scene-consistent future predictions that predict the joint future behaviour of multiple agents, as this enables reasoning about potential future multi-agent interactions, which are critical for downstream planning.

In this thesis, we address the problem of generating a set of scene-level, or joint, future trajectory predictions in multi-agent driving scenarios. To this end, we propose FJMP, a Factorized Joint Motion Prediction framework for multi-agent interactive driving scenarios. FJMP models the future scene interaction dynamics as a sparse directed interaction graph, where nodes represent agents and edges denote explicit interactions between agents. We then prune the graph into a directed acyclic graph (DAG) and decompose the joint prediction task into a sequence of marginal and conditional predictions according to the partial ordering of the DAG, where joint future trajectories are decoded using a directed acyclic graph neural network (DAGNN). We conduct experiments on two autonomous driving datasets and demonstrate that FJMP produces more accurate and scene-consistent joint trajectory predictions than existing approaches. Importantly, we show that FJMP produces superior joint forecasts compared to non-factorized approaches on the most interactive and kinematically interesting agents, which highlights the benefit of our proposed factorization.

Acknowledgements

I would like to thank my advisor, Prof. Krzysztof Czarnecki, for the invaluable guidance, encouragement, and support he provided throughout my graduate studies. I am incredibly thankful for the freedom he granted me to explore and pursue my research interests, which has not only greatly enriched my graduate experience but also fostered my growth and confidence as a researcher.

I would also like to thank all the members of the WISE lab for their kindness and support throughout my studies. I am especially thankful to Prarthana Bhattacharyya, Martin Ethier, Eli-Henry Dykhne, Prof. Sean Sedwards, and Dr. Vahdat Abdelzad for their engagement with my research and constructive feedback at our weekly meetings. Finally, I would like to thank Benjamin Thérien for the numerous insightful discussions and brainstorming sessions we have had within our office space.

To my family – Mom, Dad, Daniel, Juliet, Felix, and Ellis – thank you for being a constant source of company and support throughout my graduate studies. I am incredibly lucky to have you all in my life. Finally, to my wife, Charithe, thank you for being a constant source of love, support, and laughter throughout my graduate studies and for being by my side every day. Without you, this thesis would not have been possible.

Dedication

To Charithe

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multi-agent motion prediction is an important task in a self-driving pipeline, and it involves forecasting the future positions of multiple agents in complex driving environments. It is important that an autonomous vehicle (AV) generates plausible and diverse future predictions. This capability enables AVs to plan collision-free trajectories while safely and effectively interacting with other road agents. While humans exhibit a remarkable ability to predict the future behaviour of other road users while driving, the task of multi-agent motion prediction remains a challenging problem for AVs. This difficulty arises primarily from AVs' inability to directly observe the intent of other road agents, leading to a substantial amount of aleatoric uncertainty in future trajectory estimation. This is a problem for humans too; however, humans often rely on subtle cues and signals to indicate intent, which is challenging for AVs to reliably interpret. Moreover, multi-agent motion prediction requires reasoning effectively about the interactions and negotiations with other human road users, which requires a sophisticated understanding of social norms on the road. Human drivers have an intuitive grasp of these norms, but imbuing AVs with such social awareness is a complex endeavour [57]. Addressing these open challenges in multi-agent motion prediction is crucial for ensuring more safe and robust interactions between AVs and human agents on the road.

Most existing works in multi-agent motion prediction predict a set of marginal trajectories for each agent [34, 19, 61, 35, 14, 53, 23], and thus fail to explicitly account for agent interactions in the future. This results in trajectory predictions that are not consistent with each other. For example, the most likely marginal prediction for two interacting agents may collide with each other, when in reality a negotiation between agents to avoid collision is far more likely. As scene-consistent future predictions are critical for downstream planning, recent work has shifted toward generating a set of scene-level, or joint,

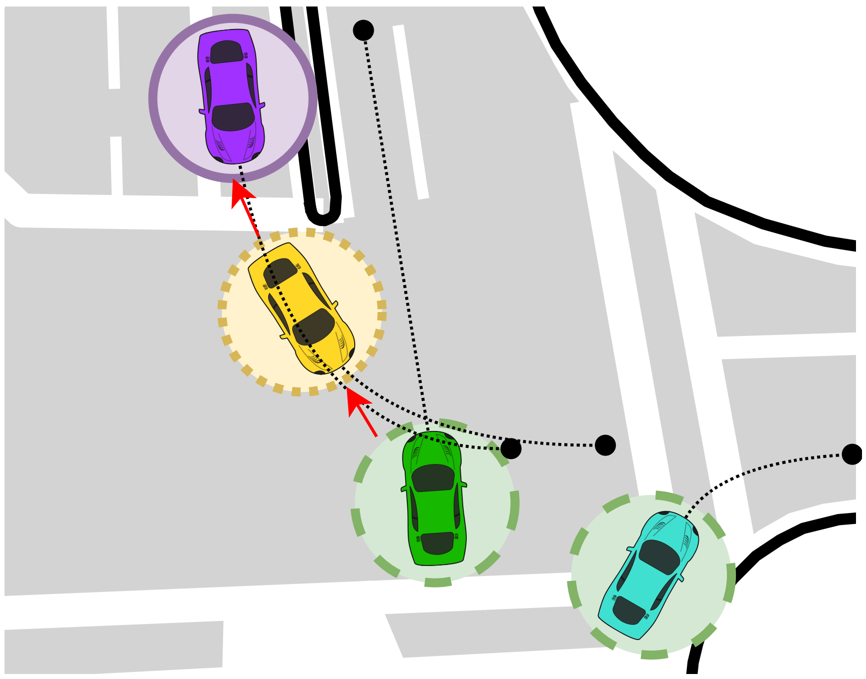


Figure 1.1: An illustration of the directed acyclic interaction graph, comprised of colored nodes and red arrows. The dotted black lines denote the ground-truth futures over a short time horizon. FJMP first produces marginal predictions for the green (dashed) nodes, followed by a conditional prediction for the yellow (dotted) node and a conditional prediction for the purple (solid) node, with conditioning on the predicted future of the parent nodes in the graph.

future trajectory predictions [49, 17, 5, 42, 7, 8, 18, 47, 66, 37, 68, 20], whereby each mode consists of a future trajectory prediction for each agent and the predicted trajectories are consistent with each other.

In this thesis, we focus on the problem of generating a set of joint future trajectory predictions in multi-agent driving scenarios. Unlike marginal prediction, the joint trajectory prediction space grows exponentially with the number of agents in the scene, which makes this prediction setting particularly challenging. A common approach for this setting is to simultaneously predict the joint futures for all agents in the scene [17, 5, 42, 18, 8, 66, 68]; however, this approach fails to explicitly reason about future interactions in the joint predictions. To address this limitation, recent work has shown that decomposing the joint prediction task of two interacting agents into a marginal prediction for the *influencer* agent and a conditional prediction for the *reactor* agent, where the reactor's prediction conditions on the predicted future of the influencer, can generate more accurate and scene-consistent joint predictions than methods that generate marginal predictions or simultaneous joint predictions [49, 32]. However, these methods are optimized for the joint prediction of only two interacting agents, and they do not efficiently scale to scenes with a large number of interacting agents.

To address these limitations of existing joint motion predictors, in this thesis we propose **FJMP** – a **F**actorized **J**oint **M**otion **P**rediction framework that efficiently generates joint predictions for driving scenarios with an arbitrarily large number of agents by *factorizing* the joint prediction task into a sequence of marginal and conditional predictions. FJMP models the future scene interaction dynamics as a sparse directed interaction graph, where an edge denotes an explicit interaction between a pair of agents, and the direction of the edge is determined by their influencer-reactor relationship [49, 28, 31], as can be seen in Figure 1.1. We propose a mechanism to efficiently prune the interaction graph into a directed acyclic graph (DAG). Joint future trajectory predictions are then decoded as a sequence of marginal and conditional predictions according to the partial ordering of the DAG, whereby marginal predictions are generated for the source node(s) in the DAG and conditional predictions are generated for non-source nodes that condition on the predicted future of their parents in the DAG. To enable this sequential trajectory decoding, we adapt a lightweight directed acyclic graph neural network (DAGNN) [51] architecture for efficiently processing predicted future information through the DAG and decoding the marginal and conditional trajectory predictions. Our main contributions can be summarized as follows:

- We propose FJMP, a novel joint motion prediction framework that generates factorized joint trajectory predictions over sparse directed acyclic interaction graphs. To our knowledge, FJMP is the first framework that enables scalable factorized joint prediction on scenes with arbitrarily many interacting agents.
- We validate our proposed method on both the multi-agent INTERACTION dataset and the Argoverse 2 dataset and demonstrate that FJMP produces scene-consistent joint predictions for scenes with up to 50 agents that outperform non-factorized approaches, especially on the most interactive and kinematically complex agents. FJMP achieves state-of-the-art performance across several metrics on the challenging multi-agent prediction benchmark of the INTERACTION dataset and ranks 1st on the official leaderboard.

This thesis is structured into six chapters as follows:

- Chapter 2 provides a necessary background. In this chapter, we formally define the multi-agent motion prediction problem and articulate the rationale behind favoring joint motion prediction over marginal motion prediction. We also present a concise overview of graph neural networks (GNNs).

- Chapter 3 surveys relevant literature in motion prediction for autonomous driving, interaction modeling for autonomous driving, and joint motion prediction. This chapter is adapted from [45].

- Chapter 4 presents the proposed FJMP method in detail, elucidating the architectural details of the feature encoder, DAG predictor, and factorized joint decoder. This chapter is adapted from [45].

- Chapter 5 conducts a detailed analysis of the performance of FJMP across two popular autonomous driving datasets, as well as a comprehensive ablation study that offers insights into the impact of the core components of FJMP on its overall performance. This chapter is adapted from [45].

- Chapter 6 summarizes our contributions and closes with limitations and several interesting directions for future work.

Chapter 2

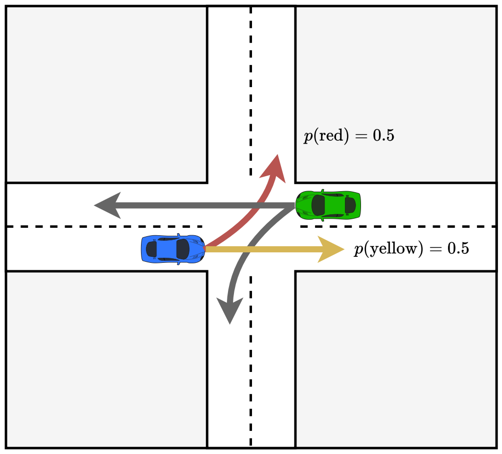
Background

In this chapter, we begin by formally defining the multi-agent motion prediction problem, and we explain the rationale behind addressing the task of joint motion prediction. Next, we provide a concise background of GNNs and cover a GNN variant that is utilized in FJMP: the directed acyclic graph neural network (DAGNN) [51].

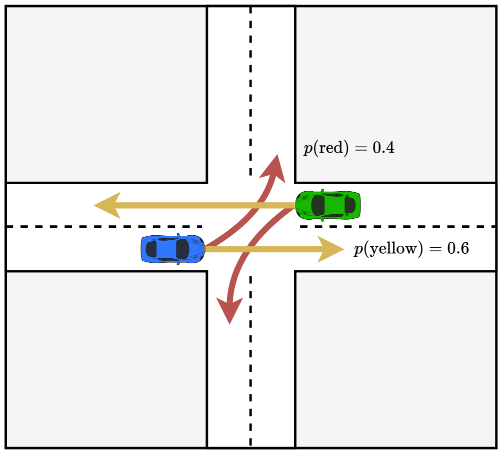
2.1 Multi-Agent Motion Prediction

The goal of multi-agent motion prediction is to predict the future T_{fut} timesteps of N dynamic agents in a scene given the past motion of the N agents and the structure of the High-Definition (HD) Map. As there are multiple possible futures for a given past, the joint motion prediction task involves predicting $K > 1$ modalities, whereby each modality consists of a predicted future for each agent in the scene. We let X and Y denote the past trajectories and future trajectories for all N agents in the scene, respectively, where $X_{\mathcal{S}}$ denotes the past trajectories for all agents in the set $\mathcal{S} \subseteq [N]$, and $Y_{\mathcal{S}}$ is defined similarly.

Most existing work in multi-agent motion prediction is focused on the task of marginal prediction. Specifically, for each agent i, the marginal prediction task involves predicting K future trajectories with optional associated probabilities: $\{\hat{Y}^k_{\{i\}}, p(\hat{Y}^k_{\{i\}})\}_{k=1}^K$, where $\sum_{k=1}^K p(\hat{Y}^k_{\{i\}}) = 1$. The primary reason for such focus is that the dominant evaluation metrics driving most popular motion prediction benchmarks [6, 3, 58, 13] are based on minimum trajectory displacement errors computed independently for each agent of interest. This thesis, however, is focused on the task of joint motion prediction. Concretely, the joint prediction task involves predicting a set of K future scene evolutions, or joint trajectory



(a) marginal prediction



(b) joint prediction

Figure 2.1: (a) An illustration of a marginal prediction output for $K = 2$, where we focus on the blue agent. Each colour represents an agent trajectory mode, with associated trajectory likelihoods. (b) An illustration of a joint prediction output, with $K = 2$. Each colour represents a joint trajectory mode, with associated joint likelihoods.

predictions, with optional associated probabilities: $\{\hat{Y}^k, p(\hat{Y}^k)\}_{k=1}^K$, where $\sum_{k=1}^K p(\hat{Y}^k) = 1$. Importantly, in the marginal case, we assign a probability to each *agent trajectory prediction*, whereas in the joint case, we assign a probability to each *joint trajectory prediction*, where a joint trajectory prediction consists of an agent trajectory prediction for every agent of interest. Figure 2.1 illustrates the difference between marginal prediction and joint prediction.

2.1.1 Why Joint Prediction?

Although marginal prediction is more extensively studied in the motion prediction literature, it has clear drawbacks compared with joint motion prediction. In particular, marginal predictions are not associated across multiple agents, which results in scene-inconsistent predictions. This makes it difficult to reason about the multi-agent interactions that may evolve in the future, which is critical for downstream planning. Although a naive solution could be to enumerate all possible combinations of marginal predictions to produce joint predictions, this approach is computationally intractable for large scenes, as the output space grows exponentially in the number of agents in the scene. That is, for N agents and K marginal predictions per agent, there are N^K such combinations. Instead, joint prediction offers a more tractable approach that involves directly generating K joint future predictions that account for multi-agent interactions. For these reasons, we study the joint motion prediction task in this thesis. Moreover, many popular benchmarks have recently released a corresponding joint prediction benchmark, where minimum trajectory-displacement errors are calculated at the scene-level, rather than at the agent-level [13, 64, 58].

2.2 Graph Neural Networks

Graphs, which are sets of nodes connected by edges, naturally arise in a wide variety of disciplines – including biology, sociology, computer science, and medicine. Given the modern advancements in deep learning, there has been a recent surge of interest in applying deep learning methods in the graph-structured domain. These new graph algorithms, generally grouped under the name graph neural networks (GNNs), have shown significant improvements in both performance and efficiency in a variety of graph-related tasks, such as node, edge, and graph classification [60]. In this section, we provide a brief overview of graph neural networks (GNNs) and a GNN variant that is used in the FJMP architecture for learning representations over DAGs: the directed acyclic graph neural network (DAGNN).

2.2.1 Graph Theory Basics

We define a graph by a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of N vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of directed edges between pairs of vertices in $\mathcal{V}$.

We define a *walk* on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a finite sequence of edges $(e_1, e_2, \ldots, e_k)$ in $\mathcal{E}$ that join a sequence of vertices $(v_0, v_1, \ldots, v_k)$ in $\mathcal{V}$ such that $e_i = (v_{i-1}, v_i)$ for all i. A *trail* is a walk in which all edges are distinct, and a *cycle* is a non-empty trail in which only the first and last vertices are equal. A graph $\mathcal{G}$ is a directed acyclic graph (DAG) if there are no cycles in $\mathcal{G}$. Every DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ admits a unique topological ordering of its vertices, which is an ordering of the vertices such that for every edge $e = (u, v) \in \mathcal{E}$, u comes before v in the ordering.

As we are interested in learning representations over the vertices in $\mathcal{G}$, we assume that each vertex v in $\mathcal{V}$ has an associated feature vector $\mathbf{h}_v \in \mathbb{R}^D$, which can be aggregated to form a feature matrix $\mathbf{H} \in \mathbb{R}^{N \times D}$. The connectivity of the graph can be represented as an $N \times N$ adjacency matrix A, where $A_{uv} = 1$ for all $(u, v) \in \mathcal{E}$. Given a vertex $v \in \mathcal{V}$, we define the neighbourhood of v by $\mathcal{N}(v) = \{u : (u, v) \in \mathcal{E}\}$. It is important to note that it is possible to attach additional features to the edges $\mathcal{E}$; however, for simplicity, we assume that all features lie on the vertices $\mathcal{V}$ of $\mathcal{G}$. At a high level, a GNN is a function parameterized by a neural network $f_\theta(\mathcal{G}, \mathbf{H})$ that leverages the graph structure of $\mathcal{G}$ to learn meaningful representations over the vertices $\mathcal{V}$, given initial features $\mathbf{H}$. A GNN can be decomposed into a sequence of *neural message passing* layers, which we cover in the following section.

2.2.2 Neural Message Passing

A neural message passing layer ℓ takes the following form:

$$\mathbf{h}_v^\ell = \mathrm{COMB}^\ell(\mathbf{h}_v^{\ell-1}, \mathrm{AGG}^\ell(\{\mathbf{h}_u^{\ell-1} | u \in \mathcal{N}(v)\})), \tag{2.1}$$

where $\mathbf{h}_v^0 = \mathbf{h}_v$ is the initial node feature for node v, and AGG^ℓ and COMB^ℓ are parameterized neural networks. At a high level, a neural message passing layer first applies a permutation-invariant pooling layer AGG^ℓ that aggregates the features from the local neighbourhood of v and then updates the node representation of v with an update function COMB^ℓ. The output of AGG^ℓ is commonly referred to as a *message*, which is a representation of the information coming from the local neighbourhood of v. We apply a neural message passing layer ℓ independently to all nodes $v \in \mathcal{V}$. We emphasize that a neural message passing layer exploits the graph structure of $\mathcal{G}$ by updating the representation of v with features from the local neighbourhood of v. This is similar to how a convolutional neural network computes local features through the convolution operation [30], but in the case of GNNs the convolutional window is defined by the vertex neighbourhood.

Common examples of the pooling layer AGG^ℓ include summation and mean; however, AGG^ℓ may also be parameterized by a neural network, as we will see in Section 2.2.3. The update function COMB^ℓ is typically represented by a simple multi-layer perceptron (MLP) or recurrent neural network (RNN), as we will see in Section 2.2.3.

A GNN is simply a sequence of L neural message passing layers, where each neural message passing layer is parameterized with its own set of neural network weights θ_ℓ. After applying multiple layers of neural message passing, the final node representations can be used for tasks such as node classification and graph classification [60]. For example, for node classification, typically a differentiable loss function is computed for all nodes concurrently, which enables updating the weights $\{\theta_\ell\}_{\ell=1}^L$ of the GNN via backpropagation.

2.2.3 Directed Acyclic Graph Neural Network (DAGNN)

A DAGNN is a recently proposed GNN architecture that is tailored specifically for DAG classification tasks [51]. While a typical GNN applies a neural message passing layer to all vertices in parallel, DAGs offer a unique topological ordering that can serve as an inductive bias, enabling the learning of more powerful representations over DAGs. The core principle behind a DAGNN is to leverage this partial ordering by updating node representations sequentially, following the topological ordering of the DAG. Specifically, neural message passing layer ℓ in a DAGNN takes the following form:

$$\mathbf{h}_v^\ell = \text{COMB}^\ell(\mathbf{h}_v^{\ell-1}, \text{AGG}^\ell(\{\mathbf{h}_u^\ell | u \in \text{pa}_\mathcal{G}(v)\}, \mathbf{h}_v^{\ell-1})), \tag{2.2}$$

where $\text{pa}_\mathcal{G}$ denotes the parents of v in $\mathcal{G}$. Processing information according to the topological ordering enables using more recently updated information to update the node representations. Notably, AGG^ℓ aggregates features $\mathbf{h}_u^\ell$ from the parents of v that have already been updated in layer ℓ.

In practice, the DAGNN parameterizes the aggregation function with additive attention. Specifically, the message for node v in layer ℓ is computed as:

$$\mathbf{m}_v^\ell := \text{AGG}^\ell(\{\mathbf{h}_u^\ell | u \in \text{pa}_\mathcal{G}(v)\}, \mathbf{h}_v^{\ell-1}) = \sum_{u \in \text{pa}_\mathcal{G}(v)} \alpha_{vu}^\ell(\mathbf{h}_v^{\ell-1}, \mathbf{h}_u^\ell)\mathbf{h}_u^\ell. \tag{2.3}$$

The attention coefficients α_{vu}^ℓ assign varying importance to the different neighbours of v and they are computed as follows:

$$\alpha_{vu}^\ell(\mathbf{h}_v^{\ell-1}, \mathbf{h}_u^\ell) = \text{softmax}_{u \in \text{pa}_\mathcal{G}(v)}\left(\mathbf{w}_1^{\ell T}\mathbf{h}_v^{\ell-1} + \mathbf{w}_2^{\ell T}\mathbf{h}_u^\ell\right), \tag{2.4}$$

where $\mathbf{w}_1^{\ell T}$ and $\mathbf{w}_2^{\ell T}$ are learnable weight vectors. By definition, the source nodes $s \in \mathcal{S}$ in the DAG contain no parents; that is $\text{pa}_\mathcal{G}(s) = \emptyset$. In this case, we let $\text{AGG}^\ell(\{\mathbf{h}_u^\ell | u \in \text{pa}_\mathcal{G}(s)\}, \mathbf{h}_s^{\ell-1}) = \text{AGG}^\ell(\emptyset, \mathbf{h}_s^{\ell-1}) := \mathbf{0}$.

The DAGNN introduces a parameterization of the update function using a gated-recurrent unit (GRU), which is commonly employed for sequential data processing, but in this context, it proves effective for processing data in topological order as well. Specifically, the updated node representation is computed as:

$$\mathbf{h}_v^\ell = \text{COMB}^\ell(\mathbf{h}_v^{\ell-1}, \mathbf{m}_v^\ell) = \text{GRU}^\ell(\mathbf{h}_v^{\ell-1}, \mathbf{m}_v^\ell). \tag{2.5}$$

We emphasize that the input to the GRU is the node representation for v in the previous layer $\mathbf{h}_v^{\ell-1}$, and the hidden state is the message $\mathbf{m}_v^\ell$. The reason is that the message effectively keeps track of the section of the DAG that has already been processed, thus taking on the role of the hidden state.

After performing L layers of DAGNN neural message passing, the DAGNN derives a graph representation for the DAG that can be used for DAG classification. Specifically, we max-pool the representations on the leaf nodes $\mathcal{F}$ of the DAG. We note that due to the sequential processing through the partial order, the leaf nodes effectively contain information about the full DAG. Concretely, the graph representation is computed as:

$$\mathbf{h}_{\mathcal{G}} = \mathrm{MLP}(\text{Max-Pool}_{v \in \mathcal{F}}(\|_{\ell=0}^{L} \mathbf{h}_v^{\ell})), \tag{2.6}$$

where $\|$ denotes the concatenation operation. The DAGNN captures the dependencies and relationships among nodes in the directed acyclic graph and leverages the topological order to enhance its representation learning capability. This formulation proves to be useful for achieving superior performance on several DAG classification tasks [51], and will be utilized for performing factorized joint trajectory decoding, as we will see in Section 4.4.

Chapter 3

Related Work

In this chapter, we survey the relevant related work, including related work in motion prediction for driving scenarios, interaction modeling for motion prediction, and joint motion prediction.

3.1 Motion Prediction in Driving Scenarios

Given the recent growing interest in autonomous driving, many large-scale motion prediction driving datasets [6, 58, 13, 3, 64] have been publically released, which has enabled rapid progress in the development of data-driven motion prediction methods. RNNs are a popular choice for encoding agent trajectories [17, 15, 16, 10, 26, 7] and convolutional neural networks (CNNs) are widely used in earlier works to process the birds-eye view (BEV) rasterized encoding of the HD map [18, 5, 43, 40, 7, 15]. As rasterized HD-map encodings do not explicitly capture the topological structure of the lanes and are constrained by a limited receptive field, recent methods have proposed vectorized [14, 19, 35, 46], lane graph [34, 16, 9], and point cloud [61] representations for the HD-map encoding. Inspired by the success of transformers in both natural language processing [54, 11] and vision [12], several end-to-end transformer-based methods have recently been proposed for motion prediction [18, 42, 35, 46, 21, 67, 23]. However, many of these transformer-based methods are extremely costly in model size and inference speed, which makes them impractical for use in real-world settings. FJMP adopts a LaneGCN-inspired architecture [34] due to its strong performance on competitive benchmarks [6], while retaining a small model size and fast inference speed. LaneGCN represents the HD-map as a lane graph. A lane graph is a graph $\mathcal{G} = (\mathcal{V}, \{\mathcal{E}_{pre}, \mathcal{E}_{suc}, \mathcal{E}_{left}, \mathcal{E}_{right}\})$, where the vertex set $\mathcal{V}$ consists of densely sampled lane

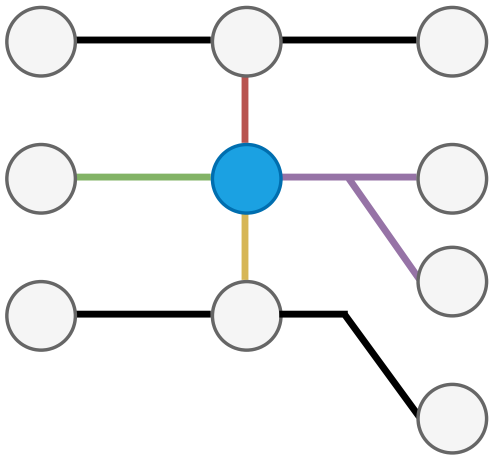


Figure 3.1: An example of a lane graph, where we focus on the blue lane node. The lane nodes can be thought of as representations of lane sections. The predecessor, successor, left, and right connections are denoted by the green, purple, red, and yellow connections, respectively.

nodes along the lane centerlines of the HD-map structure, and $\mathcal{E}_i, i \in \{\text{pre, suc, left, right}\}$, correspond to the predecessor, successor, left and right connections in the lane graph. Typically, a GNN learns feature representations over the lane nodes in $\mathcal{G}$, which are informed by the local connectivity of the lane graph. An example of a lane graph is shown in Figure 3.1.

3.2 Interaction Modeling for Motion Prediction

Data-driven methods typically use attention-based mechanisms [35, 42, 18, 34, 17, 67] or graph neural networks (GNNs) [5, 63, 33, 36, 14, 23, 28, 4] to model agent interactions for motion prediction. Recent works have demonstrated the importance of not only modeling agent interactions in the observed agent histories but also reasoning explicitly about the agent interactions that may occur in the future [46, 49, 1, 32, 29, 31]. MTR [46] proposes to generate future trajectory hypotheses as an auxiliary task, where the future hypotheses are fed into the interaction module so that it can better reason about future interactions. QCNeXt [66] performs self-attention on learnable per-agent future embeddings that are decoded into joint trajectory predictions, which enables reasoning about the self-consistency of the corresponding future predictions. GameFormer [20] formulates joint trajectory decoding as a multi-level game, where joint predictions from level $i-1$ are used together with the scene context to refine the joint predictions at level i; however, such iterative refinement can be costly at runtime. JFP [37] uses a probabilistic graphical model on top of the marginal trajectory candidates and learned potentials to generate scene-consistent joint predictions. Multiple works reason about future interactions through predicted pairwise influencer-reactor relationships [49, 32, 31], where the agent who reaches the conflict point first is defined as the influencer, and the reactor otherwise. Other works predict an explicit interaction graph for modelling future interactions between agents [37, 1, 33, 7, 28]. FJMP uses attention to model interactions in the agent histories and constructs a sparse interaction graph based on pairwise influencer-reactor relationships to model future interactions.

3.3 Joint Motion Prediction

The majority of existing motion prediction systems generate marginal predictions for each agent [53, 16, 48, 10, 14, 43, 40, 26, 21, 15, 2, 62, 34, 19, 61, 67, 35, 56, 23, 46, 63, 9]; however, marginal predictions lack an association of futures across agents. Recent works have explored generating simultaneous joint predictions [42, 5, 8, 17, 18], but these methods do not explicitly reason about future interactions in the joint predictions. Other works generate joint predictions for two-agent interactive scenarios by selecting K joint futures among all K^2 possible combinations of the marginal predictions [46, 59], which quickly becomes intractable as the number of agents in the scene increases. ScePT [7] proposes to handle the exponentially growing joint prediction space by decomposing joint prediction into the prediction of interactive cliques. However, the density of large cliques imposes a severe computational burden at inference time, which requires ScePT to upper-bound the maximum clique size to 4. To avoid the computational burden associated with dense interaction graphs, FJMP models future interactions as a *sparse* interaction graph consisting only of the strongest interactions, which enables efficient joint decoding over interactive scenarios with many interacting agents.

Our proposed method is most closely related to M2I [49], which first predicts the influencer-reactor relationship between a pair of interacting agents and then generates a marginal prediction for the influencer agent followed by a conditional prediction for the reactor agent. However, we differ from M2I in three critical ways. First, M2I is designed specifically to perform joint prediction of two interacting agents, as their model design assumes one influencer agent and one reactor agent. In contrast, FJMP naturally scales to an arbitrary number of interacting agents, where an agent may have multiple influencers and influence multiple reactors. Second, M2I requires a costly inference-time procedure that does not scale to multiple agents whereby N conditional predictions are generated for each marginal prediction, resulting in N^2 joint predictions that are pruned to $K = 6$ based on predicted likelihood. On the contrary, FJMP coherently aligns the joint predictions of a given modality through the DAG and directly produces $K = 6$ factorized joint predictions without any required pruning, which allows the system to seamlessly scale to scenes with an arbitrarily large number of agents. Third, M2I uses separate decoders for the marginal and conditional prediction, whereas FJMP decodes both marginal and conditional predictions using the same decoder, making it more parameter-efficient.

Chapter 4

Method

In this section, we describe our proposed factorized joint motion prediction framework, illustrated in Figure 4.1.

4.1 Preliminaries

4.1.1 Proposed Joint Factorization

We let X and Y denote the past trajectories and future trajectories for all N agents in the scene, respectively, where $X_{\mathcal{S}}$ denotes the past trajectories for all agents in the set $\mathcal{S} \subseteq [N]$, and $Y_{\mathcal{S}}$ is defined similarly. Moreover, we let C be an encoding of the HD-map context. We first propose to model the future scene interaction dynamics as a DAG $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where the vertices $\mathcal{V} = [N]$ correspond to the N dynamic agents in the scene, and a directed edge $e_{mn} \in \mathcal{E}$; $m, n \in [N]$, denotes an explicit interaction between agents m and n whereby m is the influencer and n is the reactor of the interaction. We propose to factorize the joint future trajectory distribution $P(Y|X, C)$ over the DAG $\mathcal{G}$ as follows:

$$P(Y|X,C) = \prod_{n=0}^{N-1} P(Y_{\{n\}}|Y_{\mathrm{pa}_{\mathcal{G}}(n)}, X, C), \tag{4.1}$$

where $\mathrm{pa}_{\mathcal{G}}(n)$ denotes the set containing the parents of node n in $\mathcal{G}$. Intuitively, the proposed joint factorization can be interpreted as an inductive bias that encourages accounting for the predicted future of the agent(s) that influence agent n when predicting the future

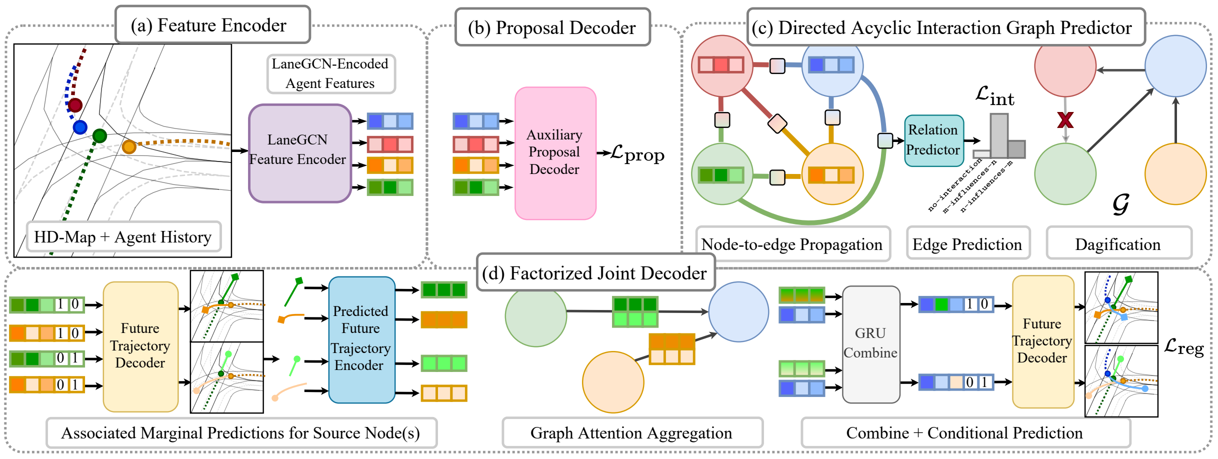

(a) Feature Encoder
HD-Map + Agent History
LaneGCN-Encoded Agent Features
LaneGCN Feature Encoder
(b) Proposal Decoder
Auxiliary Proposal Decoder
$\mathcal{L}_{\text{prop}}$
(c) Directed Acyclic Interaction Graph Predictor
Relation Predictor
$\mathcal{L}_{\text{int}}$
no-interaction
m-influences-n
n-influences-m
$\mathcal{G}$
Node-to-edge Propagation
Edge Prediction
Dagification
(d) Factorized Joint Decoder
Future Trajectory Decoder
Predicted Future Trajectory Encoder
Associated Marginal Predictions for Source Node(s)
Graph Attention Aggregation
GRU Combine
Future Trajectory Decoder
$\mathcal{L}_{\text{reg}}$
Combine + Conditional Prediction

Figure 4.1: Illustration of the proposed FJMP framework. (a) Agent histories and the HD-Map are first processed by a LaneGCN-inspired feature encoder. (b) During training, the LaneGCN-encoded features are fed into an auxiliary future proposal decoder trained with a regression loss to encourage the LaneGCN features to be *future-aware*. (c) The future-aware LaneGCN-features are processed by a GNN that predicts the pairwise influencer-reactor relationships supervised by a focal loss. A directed interaction graph $\mathcal{G}$ is constructed from the predicted edge probabilities and cycles are removed via an efficient "dagification" procedure. (d). The predicted DAG and future-aware LaneGCN features are fed into a factorized DAGNN-based trajectory decoder (red agent removed for simplicity), which produces K ($K = 2$ shown above) factorized joint futures in parallel and is supervised by a joint regression loss.

of agent n. We hypothesize that this inductive bias will ease the complexity of learning the joint distribution when compared to methods that produce a joint prediction for all N agents simultaneously.

4.1.2 Input Preprocessing

The past trajectory of a given agent is expressed as a sequence of T_{obs} states, which contains the 2D position, the velocity, and the heading of the agent at each timestep. We denote the past state of agent n at timestep $t, t \in [T_{\text{obs}}]$, by $\mathbf{x}_t^n = [\mathbf{p}_t^n, \mathbf{v}_t^n, \psi_t^n]$, where $\mathbf{p}_t^n \in \mathbb{R}^2$ is the position, $\mathbf{v}_t^n \in \mathbb{R}^2$ is the velocity, and $\psi_t^n \in \mathbb{R}$ is the yaw angle. We are also provided the agent type a^n. As in LaneGCN [34], we convert the sequence of 2D positional coordinates of each agent n to a sequence of coordinate displacements: $\hat{\mathbf{p}}_t^n = \mathbf{p}_t^n - \mathbf{p}_{t-1}^n$ for all t. We encode the HD-Map as a lane graph with M nodes, each denoting the location of the midpoint of a lane centerline segment. Using the lane graph construction proposed in LaneGCN, four adjacency matrices, $\{\mathbf{A}_i\}_{i \in \{\text{pre, suc, left, right}\}}$, $\mathbf{A}_i \in \mathbb{R}^{M \times M}$, are calculated to represent the predecessor, successor, left, and right node connectivities in the lane graph, respectively. Our system takes as input the M lane node positional coordinates, the lane node connectivities $\{\mathbf{A}_i\}_{i \in \{\text{pre, suc, left, right}\}}$, and the preprocessed agent history states $\hat{\mathbf{x}}_t^n := [\hat{\mathbf{p}}_t^n, \mathbf{v}_t^n, \psi_t^n]$ for all $n \in [N], t \in [T_{\text{obs}}]$. We let $\hat{\mathbf{X}}_{\{n\}} := [\hat{\mathbf{x}}_0^n, \ldots, \hat{\mathbf{x}}_{T_{\text{obs}}-1}^n]$ be the sequence of preprocessed observed states for agent n.

4.2 Feature Encoder

4.2.1 LaneGCN Encoder

To encode the agent history and HD-map data, we employ a LaneGCN backbone [34] with a few key modifications. For processing the agent histories, we replace LaneGCN's proposed ActorNet architecture with a gated recurrent unit (GRU) module, as shown in Equation 4.2:

$$\mathbf{h}_n^a = \text{GRU}(\hat{\mathbf{X}}_{\{n\}}), \tag{4.2}$$

for all $n \in [N]$, where the output of GRU$(\cdot)$ is the final hidden state. For processing the HD-Map, we employ LaneGCN's MapNet architecture, which consists of L graph convolutional operators that enrich the lane node features by propagating them through the lane graph. Specifically, we parameterize layer ℓ in the MapNet architecture by

$\boldsymbol{\theta}_\ell := \{\mathbf{W}_\ell^0, \mathbf{W}_\ell^{\text{left}}, \mathbf{W}_\ell^{\text{right}}\} \cup \{\mathbf{W}_{\ell,2^h}^{\text{pre}}, \mathbf{W}_{\ell,2^h}^{\text{suc}}\}_{h=0}^{H-1}$, where H is the maximum dilation size. We update the map embeddings $\mathbf{C} \in \mathbb{R}^{M \times D_m}$, where M is the number of lane nodes and D_m is the lane node feature dimension, with the graph convolutional operator defined as follows:

$$\mathbf{C} \leftarrow \mathbf{C}\mathbf{W}_\ell^0 + \sum_{h=0}^{H-1} \mathbf{A}_{\text{pre}}^{2^h}\mathbf{C}\mathbf{W}_{\ell,2^h}^{\text{pre}} + \mathbf{A}_{\text{suc}}^{2^h}\mathbf{C}\mathbf{W}_{\ell,2^h}^{\text{suc}} + \sum_{\text{dir}\in\{\text{left, right}\}} \mathbf{A}_{\text{dir}}\mathbf{C}\mathbf{W}_\ell^{\text{dir}}. \tag{4.3}$$

Here, $\mathbf{A}_{\text{pre}}^{2^h}$ and $\mathbf{A}_{\text{suc}}^{2^h}$ represent the predecessor and successor adjacency matrices for nodes that are 2^h hops away, respectively. After L applications of Equation 4.3, where each layer ℓ is parameterized with its own weights $\boldsymbol{\theta}_\ell$, we get a set of lane-node features $\mathbf{H}^m$ that capture the spatial relationships between lane nodes in the HD-map structure.

We then employ the FusionNet architecture introduced in LaneGCN [34] for fusing the map and actor features, but we remove the actor-to-lane (A2L) and lane-to-lane (L2L) modules, keeping only the lane-to-actor (L2A) and actor-to-actor (A2A) modules. We observed a minimal loss in performance when removing the A2L and L2L modules, and we benefited from the reduced parameter count, which is consistent with [2]. Concretely, the FusionNet architecture first fuses the map information into the GRU-encoded agent features via K_{L2A} applications of linear attention, where each linear attention layer is parameterized by a set of weights $\boldsymbol{\theta}_k^{\text{L2A}} := \{\mathbf{W}_{k,0}^{\text{L2A}}, \mathbf{W}_{k,1}^{\text{L2A}}, \mathbf{W}_{k,2}^{\text{L2A}}\}$ and the linear attention mechanism is defined by:

$$\mathbf{h}_n^a \leftarrow \mathbf{h}_n^a\mathbf{W}_{k,0}^{\text{L2A}} + \sum_p \phi((\mathbf{h}_n^a \parallel \Delta_{np} \parallel \mathbf{h}_p^m)\mathbf{W}_{k,1}^{\text{L2A}})\mathbf{W}_{k,2}^{\text{L2A}}, \tag{4.4}$$

where $\mathbf{h}_p^m$ is the p'th lane node feature, $\Delta_{np} = \text{MLP}(\mathbf{p}_{t_c}^p - \mathbf{p}_{t_c}^n)$ where $t_c := T_{\text{obs}} - 1$ is the present timestep, $\parallel$ denotes concatenation along the feature dimension, and ϕ is the composition of layer normalization and a ReLU activation. The summation is over the lane indices p that are within 20 meters of agent n's position at the present timestep.

Next, the FusionNet architecture models agent-to-agent interactions in the observed histories via K_{A2A} applications of linear attention, where each linear attention layer is parameterized by a set of weights $\boldsymbol{\theta}_k^{\text{A2A}} := \{\mathbf{W}_{k,0}^{\text{A2A}}, \mathbf{W}_{k,1}^{\text{A2A}}, \mathbf{W}_{k,2}^{\text{A2A}}\}$ and the linear attention mechanism is defined by:

$$\mathbf{h}_n^a \leftarrow \mathbf{h}_n^a\mathbf{W}_{k,0}^{\text{A2A}} + \sum_p \phi((\mathbf{h}_n^a \parallel \Delta_{np} \parallel \mathbf{h}_p^a)\mathbf{W}_{k,1}^{\text{A2A}})\mathbf{W}_{k,2}^{\text{A2A}}. \tag{4.5}$$

The summation is over the agent indices p that are within 100 meters of the agent n's position at the present timestep. The output of the LaneGCN FusionNet architecture is a set of agent features $H = \{\mathbf{h}_n\}_{n\in[N]}$ for each agent.

4.2.2 Auxiliary Proposal Decoder

While the output of the LaneGCN feature encoder provides informative map-aware agent features, the A2A module only considers agent interactions in the observed past trajectories. However, these features will be used downstream to reason about agent interactions in the future, and thus we desire agent feature representations that are *future-aware* – agent features that are predictive of the future. To this end, we propose to regularize the LaneGCN agent feature representations with an auxiliary pretext task that predicts joint future trajectories on top of the LaneGCN-encoded agent features. We adopt a proposal decoder f_{prop}, which decodes K joint future trajectories from the LaneGCN-encoded features $\{\hat{\mathbf{y}}^n_{\text{prop},k}\}_{k\in[K]} = f_{\text{prop}}(\mathbf{h}_n)$ and it is supervised by a joint regression loss $\mathcal{L}_{\text{prop}}$. More details of the joint regression loss can be found in Section 4.5. We hypothesize that the proposed pretext task will regularize the LaneGCN feature representations so that it contains future context that will be useful for reasoning about future interactions in the downstream modules. We note that the proposal decoder is discarded at inference time and is only used to regularize features during training.

4.3 DAG Predictor

4.3.1 Interaction Graph Predictor

In order to construct the directed acyclic interaction graph, we first must classify the future interaction label between every pair of agents in the scene. This task can be formulated as a classification task where we classify every edge in a fully-connected undirected interaction graph $\mathcal{G}_U = \{\mathcal{V}, \mathcal{E}_U\}$, where each agent corresponds to a node in $\mathcal{V}$. Similar to [49, 31, 28, 32], given an edge $e_{m,n} \in \mathcal{E}_U$, the classification task assumes three labels: `no-interaction`, `m-influences-n`, and `n-influences-m`, where the ground-truth future interaction label is heuristically determined using their ground-truth future trajectories. Concretely, we employ a collision checker to check for a collision between agents m and n at all pairs of future timesteps (t_m, t_n) where $|t_m - t_n| \leq \epsilon_I$ for some threshold ϵ_I. Details of the collision checker can be found in Section A.2. We let $\mathcal{C}$ denote the set of timestep pairs where a collision is detected. If $|\mathcal{C}| = 0$, then m and n are not interacting and the edge is labeled `no-interaction`. Otherwise, we identify the first such pair of timesteps $(\hat{t}_m, \hat{t}_n) \in \mathcal{C}$ where a collision is detected:

$$(\hat{t}_m, \hat{t}_n) = \underset{(t_m,t_n)\in\mathcal{C}}{\arg\min} \quad \min\{t_m, t_n\}. \tag{4.6}$$

If $|\mathcal{C}| > 0$, then there exists a conflict point between the two agents, and the influencer agent is defined as the agent who reaches the conflict point first. Specifically, if $\hat{t}_m < \hat{t}_n$, then we assign the edge the label `m-influences-n`, and otherwise we assign the edge the label `n-influences-m`.

With the heuristic interaction labels, we train a classifier to predict the interaction type on each edge of $\mathcal{G}_U$. We first initialize the node features of $\mathcal{G}_U$ to the future-aware LaneGCN agent features $\mathbf{h}_n$. We then perform a node-to-edge feature propagation step, where for each edge $e_{m,n}$:

$$\mathbf{h}^e_{m,n} = f_{\text{edge}}\Big(\big[\mathbf{h}_m || \mathbf{h}_n || f_{\text{dist}}(\mathbf{p}^m_{t_c} - \mathbf{p}^n_{t_c}) || \mathbf{a}_{m,n}\big] \Big), \tag{4.7}$$

where f_{edge} and f_{dist} are 2-layer MLPs, $t_{\text{c}} := T_{\text{obs}} - 1$ is the present timestep, and $\mathbf{a}_{m,n} = f_{\text{type}}([a_m, a_n])$ is the output of a 2-layer MLP f_{type} applied to the agent types a_m, a_n. We then classify the interaction label using a 2-layer MLP f_{int} with a softmax activation, which outputs a 3-dimensional softmax probability vector:

$$\hat{r}_{m,n} = \text{softmax}(f_{\text{int}}(\mathbf{h}^e_{m,n})). \tag{4.8}$$

The interaction classifier is trained with a focal loss $\mathcal{L}_{\text{int}} = \mathcal{L}^{\gamma,\alpha}_{\text{focal}}(R, \hat{R})$ with hyperparameters γ and α, where $\hat{R}$ is the predicted interaction label distributions and R is the ground-truth interaction labels. From the predicted interaction label distributions, we can construct a directed interaction graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ by selecting the interaction label on each edge with the highest predicted probability. For each pair of agents, we add a directed edge from the predicted influencer to the predicted reactor if an interaction is predicted to exist, and no edge is added otherwise.

4.3.2 Dagification

In order to perform factorized joint prediction over the learned directed interaction graph $\mathcal{G}$, we require $\mathcal{G}$ to be a DAG. We propose to remove cycles from $\mathcal{G}$, or "dagify" $\mathcal{G}$, by iterating through the cycles in $\mathcal{G}$ and removing the edges with the lowest predicted probability. We efficiently enumerate the cycles in $\mathcal{G}$ using Johnson's algorithm [24], which has time complexity $O((|\mathcal{V}| + |\mathcal{E}|)(c + 1))$, where c is the number of cycles in $\mathcal{G}$. As the directed interaction graphs are typically sparse ($|\mathcal{V}| \approx |\mathcal{E}|$) with a small number of cycles, for our application Johnson's algorithm runs approximately linear in the number of agents in the scene.

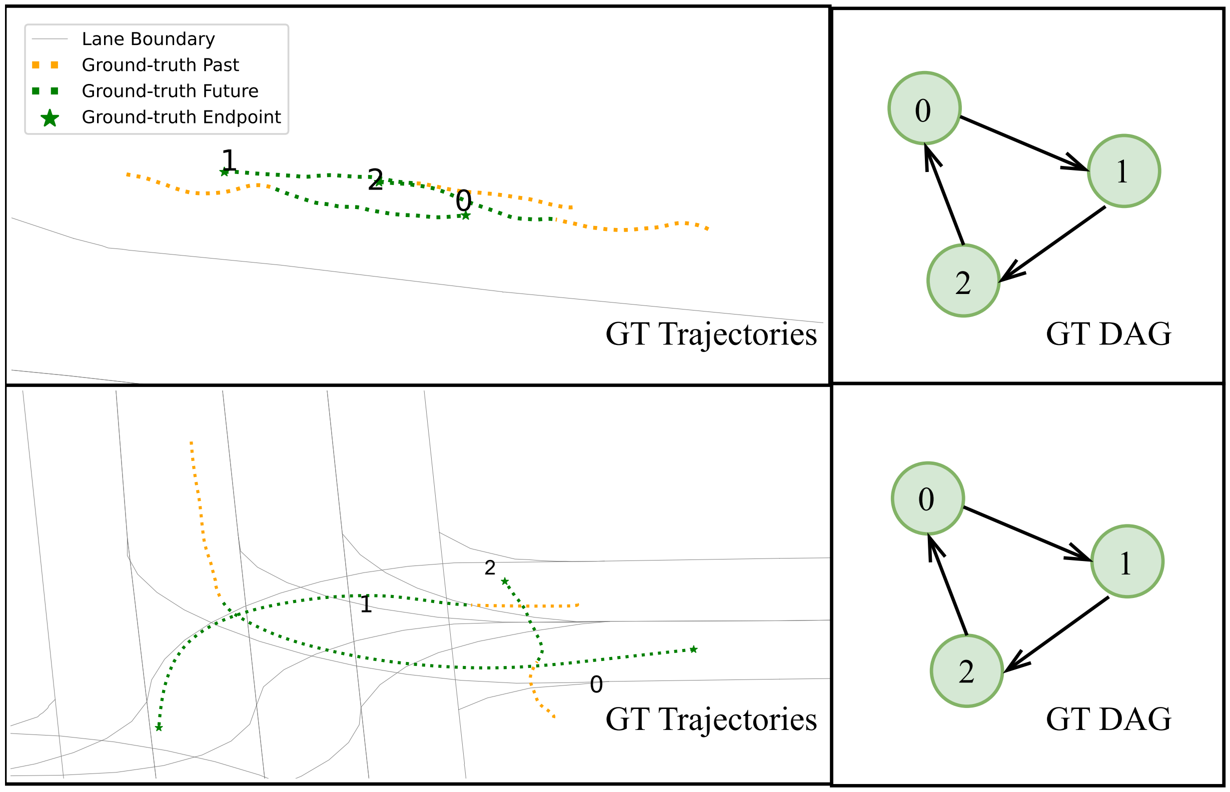


Figure 4.2: Examples of cycles in the ground-truth interaction graphs in the Argoverse 2 dataset.

4.3.3 Acyclicity Loss

FJMP requires a DAG to perform factorized decoding. We first note that there are only 50 cycles in the ground-truth interaction graph across all 225,000 scenes in the Argoverse 2 dataset [58], with 48 cycles in the training set and 2 cycles in the validation set. The large majority of the cycles arise from crowds of pedestrians walking in opposite directions, as shown in the top panel of Figure 4.2. A couple of cycles arise from cyclic structures in the trajectories, as shown in the bottom panel of Figure 4.2.

To encourage the FJMP interaction graph predictor to predict DAGs, we include an optional differentiable acyclicity loss function to bias the predictor to predict DAGs, which reduces the computational overhead of the dagification procedure proposed in Section 4.3.2.

Concretely, we introduce the following differentiable acyclicity loss function:

$$\mathcal{L}_{\text{DAG}} = \lambda_{\text{DAG}}\Big(\text{tr}(e^{\hat{A}\odot\hat{A}}) - N\Big), \tag{4.9}$$

where $\odot$ denotes element-wise multiplication, e^M denotes the matrix exponential operator on matrix M, and $\hat{A} \in \mathbb{R}^{N\times N}$ is the predicted weighted adjacency matrix derived from the softmax probability scores output by the interaction graph predictor:

$$\hat{A}_{mn} = \hat{r}_{m,n}[1], \tag{4.10}$$

$$\hat{A}_{nm} = \hat{r}_{m,n}[2]. \tag{4.11}$$

We interpret the weighted adjacency matrix as an edge (m, n) existing between nodes m and n iff $\hat{A}_{mn} > 0$. It can be shown that $\mathcal{L}_{\text{DAG}} \geq 0$ and $\mathcal{L}_{\text{DAG}} = 0$ iff the graph induced by $\hat{A}$ is acyclic [65]. Moreover, it can be shown that the values of $\mathcal{L}_{\text{DAG}}$ quantify the "DAG-ness" of the graph, with lower values of $\mathcal{L}_{\text{DAG}}$ indicating graphs that are closer to a DAG [65]. We note that the DAG loss is not used in the default FJMP formulation, but we experiment with adding the DAG loss in the ablation study conducted in Section 5.6.4.

4.4 Factorized Joint Trajectory Decoder

Given the future-aware LaneGCN feature encodings $H = \{\mathbf{h}_n\}_{n\in[N]}$ and the directed acyclic interaction graph $\mathcal{G}$, we perform factorized joint prediction according to the unique partial ordering of $\mathcal{G}$. We parameterize the factorized joint trajectory decoder using an adapted directed acyclic graph neural network (DAGNN) [51]. The originally proposed DAGNN framework performs DAG-level classification tasks on top of the representations of the *leaf* nodes in the DAG, where node features are propagated to the leaf nodes sequentially according to the partial ordering of the DAG. Although we desire to process the agents according to the partial ordering of the interaction graph $\mathcal{G}$, we also aim to use the intermediate updated node features of the DAG to generate conditional future trajectory predictions, and thus we adapt the DAGNN design to fit this criterion. We explain first how to produce a factorized joint prediction using the proposed adapted DAGNN decoder, and then how the proposed decoder is extended to produce multiple joint futures.

The factorized decoder first processes the source node(s) $\mathcal{S}$ in parallel. For each source node $s \in \mathcal{S}$, we first decode a marginal future trajectory prediction:

$$\hat{\mathbf{y}}^s = \text{DECODE}(\mathbf{h}_s), \tag{4.12}$$

where DECODE is a residual block followed by a linear layer and $\hat{\mathbf{y}}^s \in \mathbb{R}^{2T_{\text{fut}}}$ is the sequence of predicted future trajectory coordinates. We then encode the predicted future trajectories of each source node $s \in \mathcal{S}$:

$$\mathbf{e}_s = \text{ENCODE}(\hat{\mathbf{y}}^s), \tag{4.13}$$

where ENCODE is a 3-layer MLP. For each $s \in \mathcal{S}$, the encoding of the predicted future $\mathbf{e}_s$ is then fed along the outgoing edges of s. Namely, after processing the source nodes $\mathcal{S}$, we update the features of the nodes that are next in the partial ordering of $\mathcal{G}$. For every such node n, we perform the following update:

$$\mathbf{h}_n \leftarrow \text{COMB}\left(\text{AGG}\left(\{\mathbf{e}_m + \mathbf{a}_{mn} | m \in \text{pa}_{\mathcal{G}}(n)\}\right), \mathbf{h}_n\right), \tag{4.14}$$

where AGG is a neural network that *aggregates* the node features from n's parents and COMB is a neural network that *combines* this aggregated information with n's features to update the feature representation of n with conditional context about the predicted future of n's parents. $\mathbf{a}_{mn} = f^{\text{dec}}_{\text{type}}([a_m, a_n])$ is the output of a 2-layer MLP $f^{\text{dec}}_{\text{type}}$ applied to the agent types. From here, we let $\mathbf{b}_{mn} := \mathbf{e}_m + \mathbf{a}_{mn}$. Similar to DAGNN [51] which uses additive attention, we parameterize AGG using graph attention [55]:

$$\mathbf{m}_n := \text{AGG}\left(\{\mathbf{b}_{mn} | m \in \text{pa}_{\mathcal{G}}(n)\}\right) = \sum_{m \in \text{pa}_{\mathcal{G}}(n)} \alpha_{mn} \mathbf{W}_1 \mathbf{b}_{mn}, \tag{4.15}$$

$$\alpha_{mn} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}_1 \mathbf{b}_{mn} || \mathbf{W}_2 \mathbf{h}_n]))}{\sum_{k \in \text{pa}_{\mathcal{G}}(n)} \exp(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}_1 \mathbf{b}_{kn} || \mathbf{W}_2 \mathbf{h}_n]))}. \tag{4.16}$$

COMB is parameterized by a GRU recurrent module:

$$\mathbf{h}_n \leftarrow \text{COMB}(\mathbf{m}_n, \mathbf{h}_n) = \text{GRU}(\mathbf{m}_n, \mathbf{h}_n). \tag{4.17}$$

As the aggregated message $\mathbf{m}_n$ provides conditional context for updating the representation of $\mathbf{h}_n$, $\mathbf{m}_n$ is treated as the input and $\mathbf{h}_n$ is treated as the hidden state. It is important to note that the roles of the input and hidden state are reversed in the original DAGNN design [51]. The updated representation $\mathbf{h}_n$ for node n is now imbued with conditional context about the predicted future of the parent(s) of n, which can now be fed into DECODE to produce conditional future predictions for agent n. We sequentially continue the process of encoding, aggregating, combining, and decoding according to the DAG's partial order until all nodes in the DAG have a future trajectory prediction. The future trajectory predictions of all nodes are then conglomerated to attain a factorized joint prediction. We note that all nodes in the same topological order index i can be processed concurrently, resulting in an effective "sequence" length for the FJMP decoding procedure equal to the length of the longest path in the DAG. Typically, the longest path in the DAG is < 5, which limits the overhead for sequential processing and makes it less computationally expensive for our application.

4.4.1 Multiple Futures

To extend the DAGNN factorized decoder to produce multiple factorized joint predictions, we simply process K copies of $H = \{\mathbf{h}_n\}_{n \in [N]}$ through the DAG in parallel. To ensure each copy of H generates a different set of futures, we concatenate a one-hot encoding of the modality with $\mathbf{h}_n$ along the feature dimension, for each $n \in [N]$, prior to it being fed into DECODE. This approach is similar to the multiple futures approach proposed in SceneTransformer [42], and we found it to work well in our application. We train the factorized joint predictor to produce diverse multiple futures by training with a winner-takes-all joint regression loss $\mathcal{L}_{\text{reg}}$. More details about the regression loss can be found in Section 4.5.

4.5 Training Details

We first train the interaction graph predictor separately using its own feature encoder weights. The interaction graph predictor is trained via gradient descent, where the loss function is defined by:

$$\mathcal{L}_1 = \mathcal{L}_{\text{int}} + \mathcal{L}_{\text{prop}}. \tag{4.18}$$

If we use the DAG acyclicity loss, the interaction graph predictor loss is defined by $\mathcal{L}_1 = \mathcal{L}_{\text{int}} + \mathcal{L}_{\text{prop}} + \mathcal{L}_{\text{DAG}}$; however, the DAG acyclicity loss is not used in the default FJMP architecture. We experiment with utilizing the DAG acyclicity loss as an ablation in Section 5.6.4. Next, we train the factorized joint decoder using its own feature encoder weights, where the interaction graphs $\mathcal{G}$ are generated with the trained interaction graph predictor. The factorized joint predictor is trained via gradient-descent, where the loss function is defined by:

$$\mathcal{L}_2 = \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{prop}}. \tag{4.19}$$

Similar to M2I [49], we employ teacher forcing of the influencer's future trajectories during training, which helps to learn the proper influencer-reactor dynamics. The joint regression loss $\mathcal{L}_{\text{reg}}(\{\hat{Y}_k\}_{k\in[K]}, Y) := \mathcal{L}_{\ell_1}(\{\hat{Y}_k\}_{k\in[K]}, Y)$ is a scene-level smooth ℓ_1 regression loss applied to the best modality of $K = 6$ joint modalities $\{\hat{Y}_k\}_{k\in[K]}$, where the best modality attains the minimum loss:

$$\mathcal{L}_{\ell_1}(\{\hat{Y}_k\}_{k\in[K]}, Y) = \min_{k\in[K]} \frac{1}{A \cdot T_{\text{fut}}} \sum_{a\in[A]} \sum_{t\in[T_{\text{fut}}]} \text{reg}(\hat{Y}^a_{t,k} - Y^a_t), \tag{4.20}$$

where Y denotes the ground-truth future trajectory coordinates of all A agents in the scene, $\text{reg}(\mathbf{x}) = \sum_i d(x_i)$, x_i is the i'th element of $\mathbf{x}$, and $d(x)$ is the smooth ℓ_1 loss defined by:

$$d(x) = \begin{cases} 0.5x^2, & \text{if } ||x||_1 \leq 1 \\ ||x||_1 - 0.5, & \text{otherwise.} \end{cases} \tag{4.21}$$

Similarly, the auxiliary decoder loss $\mathcal{L}_{\text{prop}}$ is a scene-level smooth ℓ_1 loss applied to the best of $K = 15$ joint proposals $\{\hat{Y}^{\text{prop}}_k\}_{k\in[K]}$:

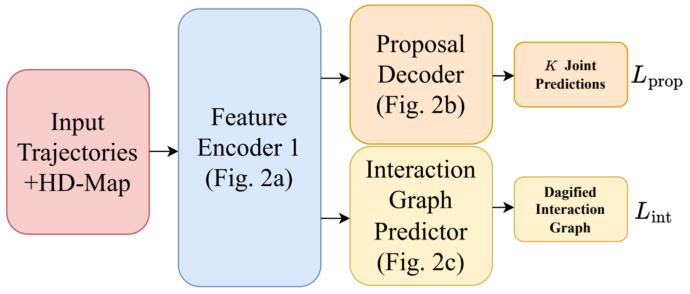
$$\mathcal{L}_{\text{prop}}(\{\hat{Y}^{\text{prop}}_k\}_{k\in[K]}, Y) := \mathcal{L}_{\ell_1}(\{\hat{Y}^{\text{prop}}_k\}_{k\in[K]}, Y). \tag{4.22}$$

We use the auxiliary proposal loss $\mathcal{L}_{\text{prop}}$ for training both the interaction graph predictor ($\mathcal{L}_1$ in Equation 4.18) and the factorized joint decoder ($\mathcal{L}_2$ in Equation 4.19) as both modules require explicit reasoning about interactions in the future trajectories, and thus *future-aware* agent features are beneficial for both modules.

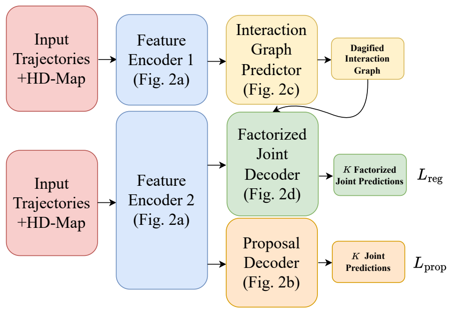
Figure 4.3 illustrates a high-level schematic of the FJMP architecture training stages at training time. We note that Feature Encoder 1 and Feature Encoder 2 consist of the same architecture as described in Section 4.2, but use separate weights.

4.6 Inference

Figure 4.4 illustrates a high-level schematic of the FJMP architecture and data flow at inference time. We note that at inference time the proposal decoders are removed.



(a) Interaction Graph Predictor training stage of FJMP.



(b) Factorized Joint Predictor training stage of FJMP.

Figure 4.3: High-level schematic of the training stages of FJMP.

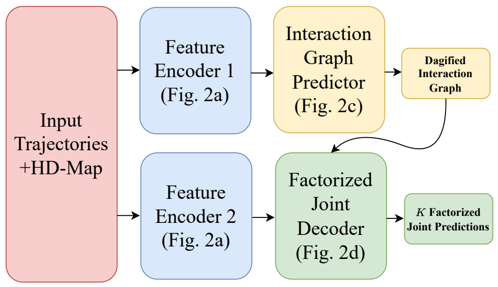


Figure 4.4: High-level schematic of the FJMP architecture at inference time.

Chapter 5

Experiments

In this chapter, we introduce the INTERACTION and Argoverse 2 motion prediction datasets that we use to evaluate FJMP. We also provide comprehensive implementation details of FJMP for each dataset. Furthermore, we provide an overview of the evaluation metrics that we use to quantify joint prediction performance as well as the performance of the FJMP interaction graph predictor. Next, we provide a concise overview of the methods we compare with FJMP. We conclude by conducting a detailed analysis of both quantitative and qualitative results across both datasets. Additionally, we present an ablation study that provides insight on the core components of FJMP responsible for its performance improvement.

5.1 Datasets

We evaluate FJMP on the INTERACTION v1.2 multi-agent dataset and the Argoverse 2 dataset, as both have multi-agent evaluation schemes for scenes with many interacting agents and require predicting joint futures for scenes with up to 40 and 56 agents, respectively. However, currently, only INTERACTION has a public benchmark for multi-agent joint prediction. Argoverse 2 contains *scored* and *focal* actors, which are high-quality tracks near the ego vehicle; and *unscored* actors, which are high-quality tracks more than 30 m from the ego vehicle. We evaluate FJMP on (i) only the scored and focal actors; and (ii) all scored, focal, and unscored actors.

INTERACTION requires predicting 3 seconds into the future given 1 second of past observations sampled at 10 Hz. INTERACTION contains 47,584 training scenes, 11,794 validation scenes, and 2,644 test scenes. A scene consists of a 4 s sequence of observations (1 s past, 3 s future) for each agent. INTERACTION contains pedestrians, bicyclists, and vehicles as context agents but only requires predicting vehicles in their multi-agent challenge. As bounding box length/width information is not provided for the pedestrian/cyclist labels, we set the length and width to a pre-defined value of 0.7m. We note that pedestrians and cyclists are not differentiated in the INTERACTION dataset.

Argoverse 2 requires predicting 6 seconds into the future given 5 seconds of past observations sampled at 10 Hz. Argoverse 2 contains 199,908 training scenes and 24,988 validation scenes. A scene consists of an 11 s sequence of observations (5 s past, 6 s future) for each agent. Argoverse 2 requires predicting trajectories for 5 agent types: vehicle, pedestrian, bicyclist, motorcyclist, and bus. As bounding box length/width information is not provided in the Argoverse 2 dataset, we use the following predefined length/width in meters for each agent type to construct the interaction labels (length/width): vehicle (4.0/2.0), pedestrian (0.7/0.7), bicyclist (2.0/0.7), motorcyclist (2.0/0.7), bus (12.5/2.5).

5.2 Implementation Details

Our models are trained on 4 NVIDIA Tesla V100 GPUs using the Adam optimizer [27]. The interaction graph predictor and factorized joint decoder are trained with the same hyperparameters. Our INTERACTION and Argoverse 2 models train in 10 and 15 hours, respectively. We now specify the implementation details for each dataset.

5.2.1 INTERACTION

For INTERACTION, we set the batch size to 64 and train for 50 epochs with a learning rate of 1e-3, step-decayed by a factor of 1/5 at epochs 40 and 48. The hidden dimension of FJMP is 128 except for the GRU history encoder, which has a hidden dimension of 256. The output of the GRU encoder is mapped to dimension 128 with a linear layer. We set $K = 6$ for the factorized decoder and $K = 15$ for the proposal decoders. For training the interaction graph predictor, we set $\gamma = 5$ and $\alpha = [1, 2, 4]$. We set $\epsilon_I = 2.5\,\text{s}$. During training, we center and rotate the scene on a random agent, as an input normalization step. During validation and test time, we center and rotate the scene on the agent closest to the centroid of the agents' current positions. We use $L = 2$ MapNet layers, $K_{\text{L2A}} = 2$ L2A layers, and $K_{\text{A2A}} = 2$ A2A layers. We use all agents in the scene for context that contains a ground-truth position at the present timestep. As centerline information is not provided in INTERACTION, for each lanelet we interpolate P evenly-spaced centerline points, where $P = \min\{10, \max\{L, R\}\}$ and L, R are the number of points on the lanelet's left and right boundaries, respectively; that is, we restrict long lanelets to have a maximum of 10 evenly-spaced centerline points. At validation time, we consider for evaluation all vehicles that contain a ground-truth position at both the present and final timesteps. We train our model on the train and validation set with the same training hyperparameters before evaluating FJMP on the INTERACTION test set.

5.2.2 Argoverse 2

The details in Section 5.2.1 apply to Argoverse 2 with the following exceptions. We set the batch size to 128 and train for 36 epochs with a learning rate of 1e-3, step-decayed by a factor of 1/10 at epoch 32. As bounding-box information is not provided with Argoverse 2, the collision checker used to construct interaction labels uses a predefined length/width for each agent type, as listed in Section 5.1. For training the interaction graph predictor, we set $\gamma = 5$ and $\alpha = [1, 4, 4]$. We set $\epsilon_I = 6\,\text{s}$ as interactions are comparatively more sparse in Argoverse 2. At validation time, we center on the ego vehicle. We increase the number of MapNet layers to 4 in Argoverse 2 to handle the larger amount of unique roadway. The L2A threshold is set to $10\,\text{m}$ as the centerline points are comparatively more dense in Argoverse 2 than in INTERACTION. We use all scored, unscored, and focal agents in the scene for context that contains a ground-truth position at the present timestep. In the *Scored* validation setting (see Section 5.2), we consider for evaluation all scored and focal agents with a ground-truth position at both the present and final timesteps. In the *All* validation setting (see Section 5.2), we consider for evaluation all scored, unscored, and focal agents with a ground-truth position at both the present and final timesteps.

5.3 Evaluation Metrics

We report the following joint prediction metrics: **minFDE** is the final displacement error (FDE) between the ground-truth and closest predicted future trajectory endpoint from the K joint predictions; **minADE** is the average displacement error (ADE) between the ground-truth and closest predicted future trajectory from the K joint predictions; **SMR** is the minimum proportion of agents whose predicted trajectories "miss" the ground-truth from the K joint predictions, where a miss is defined in Appendix A.1; and **SCR** is the proportion of modalities where two or more agents collide. The INTERACTION test set additionally reports two joint prediction metrics: **CrossCol** is the same as SCR but does not count ego collisions, and **CMR** is the same as SMR but only considers modalities without non-ego collisions. For all metrics, we evaluate $K = 6$.

These six joint prediction metrics do not necessarily capture the performance on the most interactive and challenging cases in the dataset, which is critically important for benchmarking and improving motion prediction systems. To address this limitation, we propose two new interactive metrics: (i) **iminFDE** first identifies the modality k with minimum FDE over all the agents in the scene and then computes the FDE of modality k only over agents that are *interactive*, which we heuristically define as agents with at least one incident edge in the ground-truth sparse interaction graph, where $\epsilon_I = 2.5\,\text{s}$. (ii) **iminADE** first identifies the modality k with minimum ADE over all the agents in the scene and then computes the ADE of modality k only over agents that are interactive. We found that many of the interactive cases in the datasets contain kinematically simple cases where agents exhibit simple leader-follower behaviour. To evaluate the *challenging* interactive cases, we further remove interactive agents in our evaluation that attain less than d meters in FDE with a constant velocity model. Details of the constant velocity model are in Appendix A.3. These metrics are denoted $\textbf{iminFDE}_\textbf{d}$ and $\textbf{iminADE}_\textbf{d}$, where we report $d = 3, 5$.

For the interaction graph predictor, we measure the edge accuracy for the 3 edge types: `no-interaction`, `m-influences-n`, `n-influences-m`, as well as the number of cycles predicted in the interaction graphs across all scenes in the validation set.

5.4 Methods under Comparison

We compare FJMP against the top-performing methods on the INTERACTION multi-agent test set leaderboard [17, 23, 19, 18]. FJMP is the only method on the leaderboard that performs factorized joint prediction. To systematically measure the improvement of FJMP over non-factorized approaches, we compare FJMP against a baseline called **Non-Factorized**, which computes K simultaneous joint futures from the feature representations output by the adapted LaneGCN feature encoder.

5.4.1 Non-Factorized Baseline

We explain the non-factorized baseline described in Section 5.4 in more detail. The non-factorized baseline uses the same feature encoder architecture as FJMP, but the factorized joint decoder is replaced with a DECODE module consisting of a residual block and linear layer for simultaneously decoding K joint future trajectory coordinates, where diverse futures are obtained by appending a one-hot encoding of the modality index to the agent feature representation before feeding it into DECODE, as is done in FJMP. The DECODE module is the same architecture as the DECODE module used in FJMP. The non-factorized baseline is trained with the scene-level winner-takes-all smooth ℓ_1 loss $\mathcal{L}_{\text{reg}}$ that is described in Section 4.5. The non-factorized baseline is trained with the same training hyperparameters as FJMP.

5.5 Results

5.5.1 Joint Prediction Results

Model	Venue	minADE	minFDE	SMR	CrossCol	CMR
THOMAS [17]	ICLR 2022	0.416	0.968	**0.179**	0.128	0.252
HDGT [23]	-	<u>0.303</u>	<u>0.958</u>	0.194	0.163	0.236
DenseTNT [19]	ICCV 2021	0.420	1.130	0.224	**0.000**	0.224
AutoBot [18]	ICLR 2022	0.312	1.015	0.193	0.043	0.207
HGT-Joint	-	0.307	1.056	0.186	0.016	0.190
Traj-MAE	-	0.307	0.966	<u>0.183</u>	0.021	<u>0.188</u>
FJMP (Ours)	-	**0.275**	**0.922**	0.185	<u>0.005</u>	**0.187**

Table 5.1: Joint prediction results on the INTERACTION multi-agent **test** set. Methods are sorted by the official ranking metric (CMR). For each metric, the best method is **bolded** and the second-best method is <u>underlined</u>. Lower is better for all metrics.

The joint prediction results for the top-performing methods on the INTERACTION multi-agent test set are shown in Table 5.1. FJMP performs the best on minFDE, minADE, and the official ranking metric CMR, while performing competitively on other metrics. Crucially, FJMP produces joint predictions that are both more accurate—as demonstrated by its superior performance on minADE and minFDE—and more scene-consistent—as demonstrated by its near-zero collision rate—than non-factorized approaches, which highlights the benefit of the proposed joint factorization.

Table 5.2 reports validation results on the INTERACTION and Argoverse 2 datasets, where we compare FJMP against the baseline method without joint factorization. For

Dataset	Actors Evaluated	Model	minFDE	minADE	SCR	SMR	iminFDE	iminADE	iminFDE$_3$	iminADE$_3$	iminFDE$_5$	iminADE$_5$
Interaction	-	Non-Factorized	0.643	0.199	0.004	0.088	0.688	0.210	0.784	0.240	0.854	0.261
		FJMP	0.630	0.194	0.003	0.084	0.672	0.206	0.758	0.232	0.826	0.252
		Δ	0.013	0.005	0.001	0.004	0.016	0.004	0.026	0.008	0.028	0.009
Argoverse 2	Scored	Non-Factorized	1.965	0.834	-	0.349	2.957	1.223	3.276	1.340	3.436	1.399
		FJMP	1.921	0.819	-	0.343	2.893	1.204	3.205	1.320	3.356	1.377
		Δ	0.044	0.015	-	0.006	0.064	0.019	0.071	0.020	0.080	0.022
	All	Non-Factorized	1.995	0.825	-	0.340	3.302	1.309	3.759	1.477	3.952	1.545
		FJMP	1.963	0.812	-	0.337	3.204	1.273	3.652	1.439	3.839	1.504
		Δ	0.032	0.013	-	0.003	0.098	0.036	0.107	0.038	0.113	0.041

Table 5.2: Non-Factorized Baseline vs. FJMP performance on joint metrics on the INTERACTION and Argoverse 2 validation sets. Lower is better for all metrics. We evaluate two settings on Argoverse 2: *Scored*, which include both scored and focal agents in Argoverse 2 dataset; and *All*, which includes all scored, focal, and unscored agents. Argoverse 2 lacks agent bounding box information, so SCR is not computed. Δ denotes the difference in performance between FJMP and the Non-Factorized baseline.

Argoverse 2, we have two evaluation schemes: (i) we evaluate the joint predictions of the scored and focal agents (Scored), and (ii) we evaluate the joint predictions of the scored, unscored, and focal agents (All) to demonstrate its scalability to scenes with a large number of agents. The results show that the proposed joint factorized predictor consistently provides an improvement in performance over the non-factorized baseline. We expect that FJMP improves the most over the baseline on the interactive cases in the dataset, as the proposed factorization directly enables conditioning the reactor predictions on the predicted futures of their influencers. Importantly, we note that for scenes with no predicted interactions, the factorization becomes a product of marginal predictions and thus FJMP reduces to the non-factorized prediction. As expected, the relative improvement of FJMP over the baseline is larger on the interactive and kinematically interesting cases, as demonstrated by a larger performance improvement on the interactive minFDE/minADE metrics. This indicates that the performance improvement from the joint factorization concentrates on the challenging interactive cases, while still producing accurate joint predictions for the full scene.

5.5.2 Interaction Graph Predictor Results

The edge type accuracies of the proposed interaction graph predictor on the INTERACTION and Argoverse 2 validation sets are reported in Table 5.3. Although we employed the focal loss, the `m-influences-n` and `n-influences-m` classes performed the worst in edge type accuracy as the class labels are very imbalanced, as shown in Appendix A.4.

Dataset	Edge Type	Edge Type Accuracy
INTERACTION	`no-interaction`	0.992
	`m-influences-n`	0.940
	`n-influences-m`	0.939
Argoverse 2	`no-interaction`	0.990
	`m-influences-n`	0.847
	`n-influences-m`	0.859

Table 5.3: Accuracy of each edge type on the INTERACTION and Argoverse 2 validation sets with the FJMP interaction graph predictor.

Model	Train IG	Inference IG	minFDE	minADE	iminFDE	iminADE
FJMP	Learned	Learned	1.963	0.812	3.204	1.273
FJMP	Learned	Ground-truth	1.947	0.807	3.165	1.265
FJMP	Ground-truth	Ground-truth	**1.888**	**0.789**	**2.986**	**1.220**
FJMP	Ground-truth	Learned	2.076	0.854	3.326	1.317

Table 5.4: FJMP with ground-truth vs learned interaction graphs at training and inference time on the Argoverse 2 validation set, All setting. For each metric, the best model is **bolded**. **Train IG** indicates the interaction graphs that are used during training, where **Learned** denotes the predicted interaction graphs from the interaction graph predictor and **Ground-truth** denotes the interaction graphs obtained from the labeling heuristic. The **Inference IG** column is interpreted similarly.

5.5.3 Ground-truth Interaction Graph Predictor Performance

Table 5.4 compares the performance of FJMP with three modified versions of FJMP, where either the training or inference (or both) interaction graphs are replaced with the ground-truth heuristic interaction graphs. The results in Table 5.4 indicate that the choice of interaction graph has a considerable effect on the performance of the factorized joint predictor, as indicated by an additional 4 cm improvement in iminFDE with the ground-truth interaction graph at inference time over the predicted interaction graph. Moreover, when the model is trained and evaluated with the ground-truth interaction graphs, we see a substantial increase in performance over FJMP with the learned interaction graphs. This indicates that further refinement of the interaction graph predictor may yield additional performance improvements with our FJMP design, which we leave to future work. We also see that training with the ground truth interaction graph and evaluating with the learned interaction graph performs the worst, as the network does not learn to reason about errors in the learned interaction graph.

Model	Focal Loss	`no-interaction` Acc.	`m-influences-n` Acc.	`n-influences-m` Acc.
FJMP	✓	0.990	**0.847**	**0.859**
FJMP	✗	**0.997**	0.807	0.815
Induced	-	0.996	0.830	0.837

Table 5.5: Ablation study that compares the effect of the DAG acyclicity loss on the Argoverse 2 dataset. **Num. Cycles** is the number of cycles predicted in the Argoverse 2 validation set.

5.5.4 Induced Interaction Graph Performance

Table 5.5 evaluates the interaction graph performance of an interaction graph predictor that generates predictions by using the same heuristic on the predicted joint trajectories as is used on the ground-truth future trajectories to generate the ground-truth interaction labels. Namely, we train a non-factorized baseline model that outputs $K = 1$ joint predictions and use these joint predictions to derive interaction labels following the heuristic labeling scheme proposed in Section 4.3.1. We call these interaction graph predictions the *induced* interaction graph predictions, as they are induced from the corresponding joint predictions.

Table 5.5 compares the induced interaction graph predictor against an interaction graph predictor trained with and without focal loss on the Argoverse 2 dataset. The induced interaction graph predictor performs similarly to the interaction graph predictor without focal loss, indicating that the focal loss biases the predictor to overpredict edges.

5.6 Ablation Studies

5.6.1 Sparse vs. Dense Interaction Graphs

Table 5.6 uses the INTERACTION dataset to ablate the design choice of representing the interaction graph sparsely with only the strongest pairwise interactions as edges in the graph. We compare FJMP against a variant of FJMP that uses a different labeling heuristic for the interaction graph, resulting in *denser* interaction graphs. Namely, FJMP (Dense) uses the M2I [49] heuristic: for each pair of agents, an interaction is defined to exist if any pair of future trajectory coordinates in the future trajectory horizon is within a threshold Euclidean distance of each other, where the threshold is taken to be the sum of the lengths of the two agents. The influencer-reactor relationship is determined by who reaches the

Model	minFDE	minADE	SMR	Prop. Edges	Inf. Time (s)
Non-Factorized	0.643	0.199	0.088	-	**0.010**
FJMP (Dense)	**0.623**	**0.193**	**0.081**	0.180	0.062
FJMP	<u>0.626</u>	**0.193**	<u>0.083</u>	0.045	<u>0.038</u>

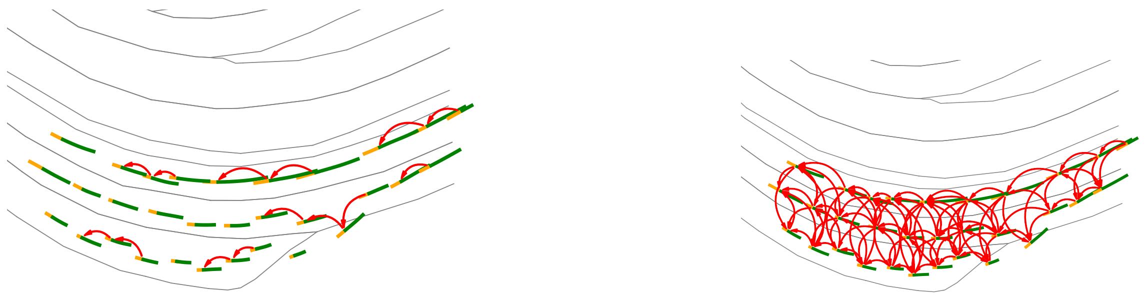
Table 5.6: Comparison of sparse vs. dense interaction graphs on the INTERACTION validation set. The FJMP model is trained and evaluated using the ground-truth sparse interaction graphs, and FJMP (Dense) is trained and evaluated using dense ground-truth interaction graphs attained via the M2I [49] labeling heuristic. **Prop. Edges** measures the proportion of agent pairs connected in the ground-truth training interaction graphs. **Inf. Time** is the inference time per validation scene on 1 NVIDIA Tesla V100 GPU.

conflict point first. We found that the M2I heuristic often adds several unnecessary edges, especially in congested scenes—as exemplified in Figure 5.1b. We train and evaluate the FJMP models in Table 5.6 using the ground-truth interaction graphs to precisely compare the different labeling heuristics. The results show that the dense (M2I) interaction graph improves very slightly over the sparse interaction graph; however, we retain most of the improvement over the non-factorized baseline with the sparse interaction graph, which indicates that modeling only the strongest interactions is sufficient to see most of the improvement with joint factorization. Moreover, the sparse interactions contain 75% fewer edges than the dense interaction graph, which accelerates inference by nearly 2x.

Figure 5.1 illustrates the ground-truth interaction graph of a congested scene according to the FJMP and M2I heuristics, respectively. We observe that the M2I heuristic adds several superfluous edges, which would lead to unnecessary additional computation for the factorized decoder.

5.6.2 Proposal Decoder and Teacher Forcing Ablation Study

Table 5.7 conducts an ablation study on Argoverse 2 where we analyze the effect of using the auxiliary proposal decoder and teacher forcing of the influencer's future trajectories during training. The results indicate that both the auxiliary proposal decoder and teacher forcing is critical for allowing the model to reason appropriately about the influencer-reactor future dynamics. Notably, without the proposal decoder (rows 2 and 3 in Table 5.7), FJMP performs similarly to the non-factorized baseline, which we hypothesize is because the LaneGCN-encoded features do not contain the necessary future context to reason appropriately about the future interactions. Teacher forcing also provides an additional



(a) Interaction graph generated with FJMP labeling heuristic.

(b) Interaction graph generated with M2I labeling heuristic.

Figure 5.1: Comparison of FJMP and M2I labeling heuristics on a congested scene from the INTERACTION dataset. The ground-truth pasts are indicated in yellow and the ground-truth futures are indicated in green. Lane boundaries are depicted as grey lines. Each red arrow points from an influencer agent to its corresponding reactor agent. We note that two agents at the bottom-right of the scene are on the shoulder of the lane.

performance benefit by removing the spurious noise in the predicted influencer trajectories, so that the model better learns the proper influencer-reactor dynamics during training. In Table 5.8, we repeat the FJMP ablation study conducted in Table 5.7 on the INTERACTION dataset. The results are consistent with Argoverse 2, showing that both the proposal decoder and teacher forcing are critical for performance.

5.6.3 Non-Factorized Baseline Ablation Study

In Table 5.9, we perform an ablation study on the various components of the non-factorized baseline model on the INTERACTION dataset. First, we ablate using a one-hot encoding for multiple futures (*One-hot Encoding*) compared with using separate decoder weights for each joint future modality (*Separate Weights*), as is done in LaneGCN [34]. The one-hot encoding method significantly improves performance; this is because when using separate weights, the winner-takes-all training process quickly converges to one future joint modality, and thus the other decoders' weights never receive gradients for updating their weights. As a result, the collision rate (SCR) significantly improves when using the one-hot encoding method. Next, we ablate using the default hyperparameter configuration for LaneGCN compared with the FJMP hyperparameter configuration. Namely, LaneGCN trains for 36 epochs with a batch size of 128, with the learning rate decreasing by a factor of 10 at epoch 32. FJMP trains for 50 epochs with a batch size of 64, with the learning rate decreasing by

Model	Prop?	TF?	minFDE	minADE	iminFDE	iminADE
Non-Factorized	✗	✗	1.995	0.825	3.302	1.309
FJMP	✗	✗	2.004	0.829	3.274	1.304
FJMP	✗	✓	2.001	0.827	3.300	1.312
FJMP	✓	✗	1.987	0.820	3.255	1.293
FJMP	✓	✓	**1.963**	**0.812**	**3.204**	**1.273**

Table 5.7: Ablation study of FJMP on Argoverse 2 validation set, All setting. **Prop?** denotes whether we include the proposal decoder during training. **TF?** denotes whether we teacher-force the influencer trajectories during training.

Model	Prop?	TF?	minFDE	minADE	iminFDE	iminADE
Non-Factorized	✗	✗	0.643	0.199	0.688	0.210
FJMP	✗	✗	0.647	0.200	0.690	0.212
FJMP	✗	✓	0.644	0.200	0.688	0.212
FJMP	✓	✗	0.636	0.197	0.677	0.208
FJMP	✓	✓	**0.630**	**0.194**	**0.671**	**0.206**

Table 5.8: Ablation study of FJMP on the INTERACTION validation set. **Prop?** denotes whether we include the proposal decoder during training. **TF?** denotes whether we teacher-force the influencer trajectories during training.

a factor of 5 at epochs 40 and 48. The FJMP hyperparameter configuration significantly improves performance over the LaneGCN hyperparameter configuration. Finally, we ablate using the modified LaneGCN feature encoder (*FJMP*) consisting of a GRU for processing agent trajectories instead of LaneGCN's proposed ActorNet module, 2 MapNet layers instead of 4, and the A2L and L2L blocks removed. These modifications yield further improvements in validation performance.

5.6.4 DAG Acyclicity Loss

Table 5.10 compares FJMP against a variant of FJMP where the interaction graph predictor is trained with the DAG acyclicity loss defined in Section 4.3.3 on the Argoverse 2 dataset. We use a DAG acyclicity loss coefficient $\lambda_{\text{DAG}} = 0.1$. Recall that there are 2 cycles in the ground-truth interaction graphs in the Argoverse 2 validation set. We can see that the DAG acyclicity loss reduces the number of predicted cycles from 114 to 6, which indicates that the DAG acyclicity loss helps in preventing FJMP from overpredicting cy-

Model	Multiple Futures Method	Hyperparameter Configuration	Feature Encoder	minFDE	minADE	SMR	SCR
LaneGCN [34]	Separate Weights	LaneGCN	LaneGCN	0.935	0.300	0.223	0.233
-	One-hot Encoding	LaneGCN	LaneGCN	0.807	0.264	0.142	0.010
-	One-hot Encoding	FJMP	LaneGCN	0.713	0.227	0.113	0.006
Non-Factorized Baseline	One-hot Encoding	FJMP	FJMP	**0.643**	**0.199**	**0.088**	**0.004**

Table 5.9: Ablation study of the Non-Factorized Baseline model on the INTERACTION validation set. **Multiple Futures Method** denotes the method used to attain multiple joint futures. **Hyperparameter Configuration** denotes the hyperparameter settings for batch size, learning rate/step, and the number of training epochs. **Feature Encoder** denotes whether we use the LaneGCN feature encoder (LaneGCN) or the simplified LaneGCN feature encoder with fewer components (FJMP).

Model	DAG Loss	λ_{DAG}	Num. Cycles	`no-interaction` Acc.	`m-influences-n` Acc.	`n-influences-m` Acc.
FJMP	✗	-	114	**0.990**	0.847	**0.859**
FJMP	✓	0.1	**6**	**0.990**	**0.849**	0.853

Table 5.10: Ablation study that compares the effect of the DAG acyclicity loss on the Argoverse 2 dataset. **Num. Cycles** is the number of cycles predicted in the Argoverse 2 validation set.

cles. Moreover, the edge-type accuracies are almost identical to those in FJMP. Table 5.11 additionally compares FJMP against the variant with the DAG acyclicity loss on the downstream joint prediction performance on the Argoverse 2 dataset. The results show that the DAG loss variant performs only marginally worse in joint prediction performance, indicating that the DAG loss is an effective way to reduce the number of predicted cycles without considerably impacting the performance of the joint predictor. We note that the DAG loss was a recently added design to FJMP, and thus is not included in the default FJMP architecture.

Model	DAG Loss	λ_{DAG}	minFDE	minADE	iminFDE	iminADE
FJMP	✗	-	**1.963**	**0.812**	**3.204**	**1.273**
FJMP	✓	0.1	1.967	0.814	3.234	1.289

Table 5.11: Ablation study that compares the effect of the DAG acyclicity loss on the downstream joint prediction performance on the Argoverse 2 dataset.

5.7 Qualitative Evaluation

5.7.1 INTERACTION

Figure 5.2 shows qualitative results of FJMP on various scenes in the INTERACTION dataset, with all $K = 6$ scene-level modalities visualized. We emphasize FJMP's ability to produce accurate and scene-consistent predictions for scenes with a large number of interacting agents.

5.7.2 Argoverse 2

In this section, we show qualitative results on scenes in the Argoverse 2 validation set where we show side-by-side comparisons between FJMP and the Non-Factorized Baseline. In Figure 5.3 and Figure 5.4, for each row, the left panel shows the non-factorized baseline predictions, the middle panel shows FJMP predictions, and the right panel shows the predicted DAG. We visualize only the best scene-level modality to avoid clutter. In Figure 5.3, we show examples where FJMP reasons properly in scenes with interactive pass-yield behaviours. In contrast, the non-factorized baseline incorrectly predicts conservative behaviour where the yielding vehicle avoids the passing vehicle's trajectory. In Figure 5.4, we show qualitative examples where FJMP correctly identifies chains of leader-follower interactions, which in turn leads to more accurate leader-follower predictions than the non-factorized baseline.

5.7.3 Failure Cases

In Figure 5.5, we illustrate two failure cases of the FJMP model on the Argoverse 2 dataset. In both cases, an erroneous influencer future prediction negatively biases the downstream reactor prediction.

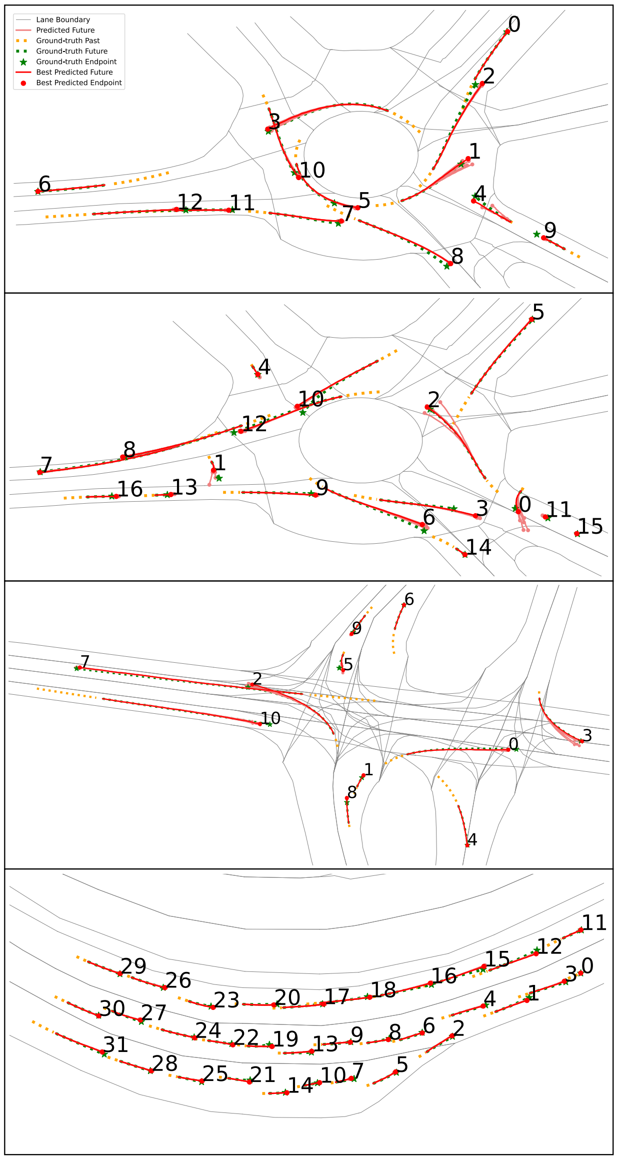


Figure 5.2: Qualitative examples of FJMP on agent-dense scenes in the INTERACTION dataset.

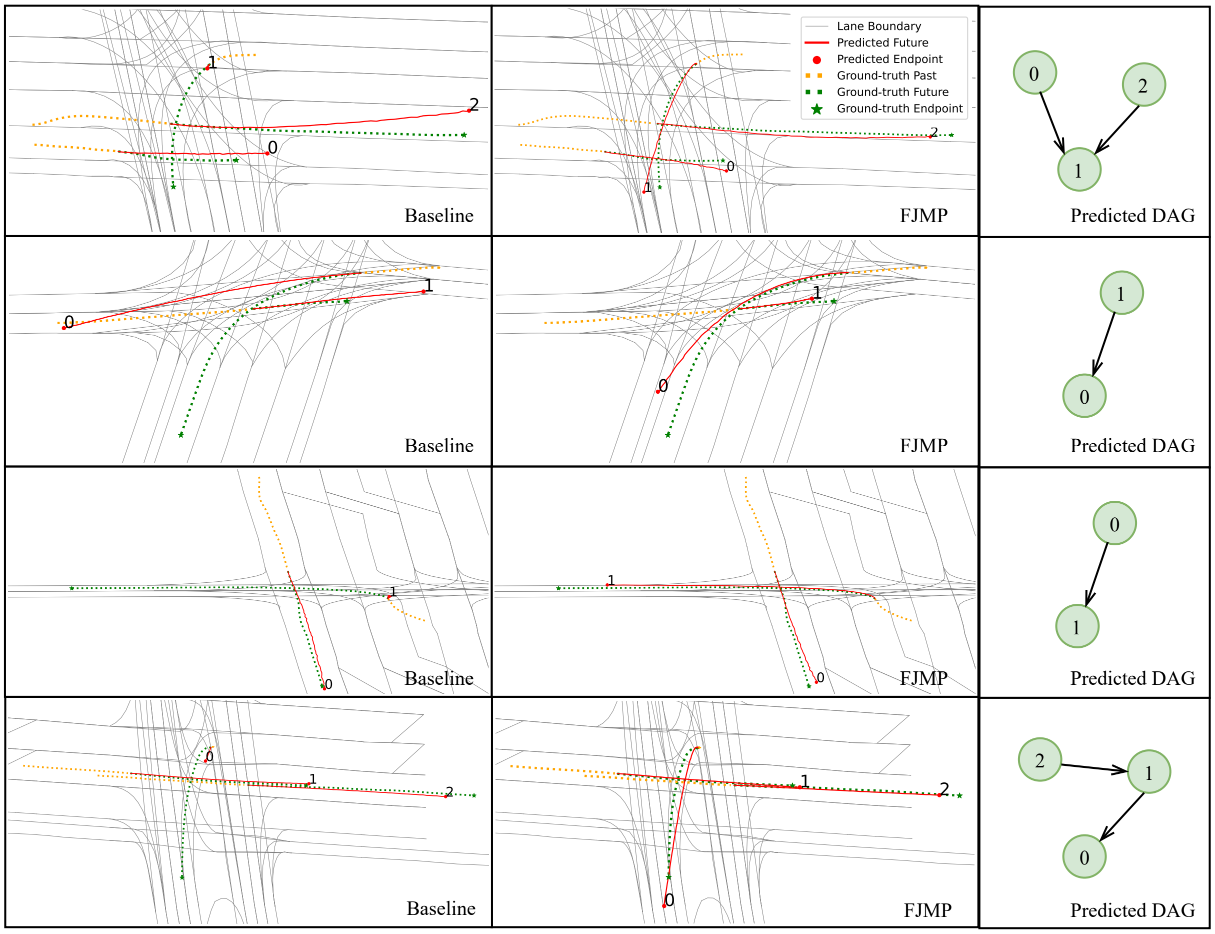


Figure 5.3: Qualitative examples of left-turn interactive scenes in the Argoverse 2 validation set. All predicted DAGs match the ground-truth DAG. In all scenes, FJMP correctly identifies the passing vehicle as the influencer and the left-turning vehicle as the reactor. The Non-Factorized baseline consistently predicts overly conservative behaviour that avoids the influencer trajectory. In contrast, FJMP consistently captures the proper left-turn behaviour.

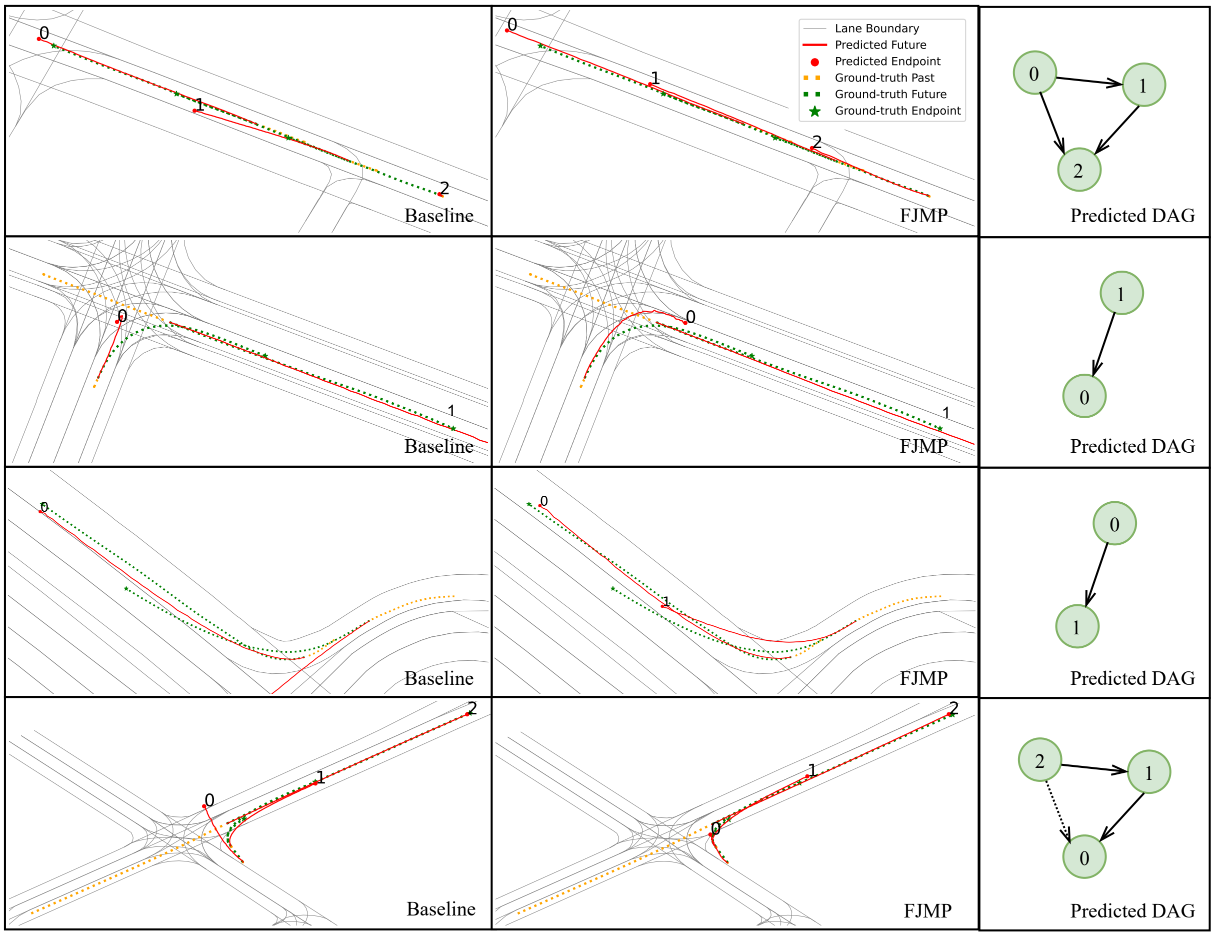


Figure 5.4: Qualitative examples of leader-follower interactive scenes in the Argoverse 2 validation set. Predicted DAGs are shown on the right, where true positive edges are indicated in solid black and true negative edges are shown in dotted black. In all of the above scenes, FJMP correctly predicts chains of influencer-reactor relationships. In the first row, the non-factorized baseline predicts conservative behaviour for the trailing vehicle. In contrast, FJMP predicts proper leader-follower behaviour for the trailing vehicle (leaf node in the DAG). In the second and third rows, the right-turn mode of the trailing vehicle is missed by the non-factorized baseline, whereas FJMP correctly identifies the right-turn mode due to correctly identifying the leader-follower interaction. In the last row, the non-factorized baseline predicts scene-incompliant behaviour for the trailing vehicle whereas FJMP predicts proper leader-follower dynamics reflecting the predicted DAG.

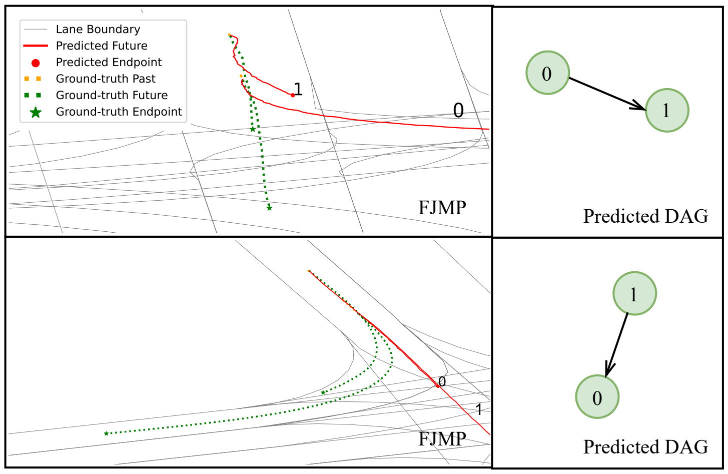


Figure 5.5: Qualitative examples of failure cases of the FJMP model. All predicted DAGs match the ground truth. In both rows, the interaction graph is correctly predicted; however, the influencer trajectory is erroneously predicted, which negatively biases the reactor's prediction to follow the influencer.

Chapter 6

Conclusion

6.1 Conclusion

In this thesis, we propose FJMP, a factorized joint motion prediction framework for multi-agent interactive driving scenarios. FJMP models the future scene interaction dynamics as a sparse directed acyclic interaction graph, which enables efficient factorized joint prediction. We demonstrate clear performance improvements with our factorized design on the Argoverse 2 and INTERACTION datasets and perform state-of-the-art on the challenging multi-agent INTERACTION benchmark.

6.2 Limitations and Future Work

The proposed framework adopts a heuristic labeling scheme to determine the ground-truth interaction graph. We observe a performance-efficiency tradeoff with a denser interaction graph; however, there may exist better heuristics for classifying future interactions that retain most of the sparsity of the interaction graph without trading off performance. Furthermore, long chains of leader-follower behaviour in congested traffic may require costly sequential processing with our method. Finding mechanisms to prune the interaction graph to best trade-off performance and efficiency is a direction we plan to explore in future work.

One critical limitation of FJMP is that the joint predictions often lack multimodality and diversity. This issue stems from the fact that while the joint future trajectory distribution is inherently multimodal, FJMP only predicts a single interaction DAG. In future work, we would like to explore predicting multiple interaction DAGs for conditioning multiple predicted joint futures. We expect that introducing diversity in the set of predicted interaction DAGs will lead to improved diversity and multimodality in the predicted joint trajectories.

Another interesting future direction is to assess FJMP's ability to perform interventional predictions at inference time by intervening on the predicted interaction graph by modifying the edges in the predicted interaction DAG and observing the effect in the downstream factorized predictions. These interventional queries may be useful for downstream planning to inform the AV system of possible "what-if" scenarios at runtime.

Finally, an interesting direction for future work is extending FJMP for use in AV planning and simulation. Our experiments have demonstrated that factorization enhances the prediction of interactive agents. We believe that a factorized design could prove valuable in generating safe and interaction-aware planned trajectories in closed-loop AV systems, especially in challenging interactive scenarios like merging and intersections.

References

[1] Yutong Ban, Xiao Li, Guy Rosman, Igor Gilitschenski, Ozanan R. Meireles, Sertac Karaman, and Daniela Rus. A deep concept graph network for interaction-aware trajectory prediction. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2022.

[2] Prarthana Bhattacharyya, Chengjie Huang, and Krzysztof Czarnecki. SSL-Lanes: Self-supervised learning for motion forecasting in autonomous driving. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2022.

[3] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[4] Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2020.

[5] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[6] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[7] Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Scept: Scene-consistent, policy-based trajectory predictions for planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[8] Alexander Cui, Sergio Casas, Abbas Sadat, Renjie Liao, and Raquel Urtasun. Lookout: Diverse multi-future prediction and planning for self-driving. In *Proceedings of the IEEE/CVF IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16087–16096, 2021.

[9] Alexander Cui, Sergio Casas, Kelvin Wong, Simon Suo, and Raquel Urtasun. Gorela: Go relative for viewpoint-invariant motion forecasting. *arXiv preprint arXiv:2211.02545*, 2022.

[10] Nachiket Deo, Eric M. Wolff, and Oscar Beijbom. Multimodal trajectory prediction conditioned on lane-graph traversals. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2021.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NACCL-HLT)*, 2019.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[13] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aurelien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset. In *Proceedings of the IEEE/CVF IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[14] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding HD maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[15] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. HOME: heatmap output for future motion estimation. In *Proceedings of the IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021.

[16] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. GOHOME: graph-oriented heatmap output for future motion estimation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2022.

[17] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. THOMAS: trajectory heatmap output with learned multi-agent sampling. In *International Conference on Learning Representations (ICLR)*, 2022.

[18] Roger Girgis, Florian Golemo, Felipe Codevilla, Martin Weiss, Jim Aldon D'Souza, Samira Ebrahimi Kahou, Felix Heide, and Christopher Pal. Latent variable sequential set transformers for joint multi-agent motion prediction. In *International Conference on Learning Representations (ICLR)*, 2022.

[19] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision, (ICCV)*, 2021.

[20] Zhiyu Huang, Haochen Liu, and Chen Lv. Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving. *arXiv preprint arXiv:2303.05760*, 2023.

[21] Zhiyu Huang, Xiaoyu Mo, and Chen Lv. Multi-modal motion prediction with transformer-based neural network for autonomous driving. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2022.

[22] Xiaosong Jia, Liting Sun, Hang Zhao, Masayoshi Tomizuka, and Wei Zhan. Multi-agent trajectory prediction by combining egocentric and allocentric views. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2021.

[23] Xiaosong Jia, Penghao Wu, Li Chen, Hongyang Li, Yu Liu, and Junchi Yan. HDGT: heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding. *arXiv preprint arXiv:2205.09753*, 2022.

[24] Donald B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM J. Comput.*, 4(1):77–84, 1975.

[25] Siddhesh Khandelwal, William Qi, Jagjeet Singh, Andrew Hartnett, and Deva Ramanan. What-if motion prediction for autonomous driving. *arXiv preprint arXiv:2008.10587*, 2020.

[26] Byeoungdo Kim, SeongHyeon Park, Seokhwan Lee, Elbek Khoshimjonov, Dongsuk Kum, Junsoo Kim, Jeong Soo Kim, and Jun Won Choi. Lapred: Lane-aware prediction of multi-modal future trajectories of dynamic agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[28] Sumit Kumar, Yiming Gu, Jerrick Hoang, Galen Clark Haynes, and Micol Marchetti-Bowick. Interaction-based trajectory prediction over a hybrid traffic graph. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[29] Yen-Ling Kuo, Xin Huang, Andrei Barbu, Stephen G. McGill, Boris Katz, John J. Leonard, and Guy Rosman. Trajectory prediction with linguistic representations. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2022.

[30] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[31] Donsuk Lee, Yiming Gu, Jerrick Hoang, and Micol Marchetti-Bowick. Joint interaction and trajectory prediction for autonomous driving using graph neural networks. *arXiv preprint arXiv:1912.07882*, 2019.

[32] Ding Li, Qichao Zhang, Shuai Lu, Yifeng Pan, and Dongbin Zhao. Conditional goal-oriented trajectory prediction for interacting vehicles with vectorized representation. *arXiv preprint arXiv:2210.15449*, 2022.

[33] Jiachen Li, Fan Yang, Hengbo Ma, Srikanth Malla, Masayoshi Tomizuka, and Chiho Choi. RAIN: reinforced hybrid attention inference network for motion forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision, (ICCV)*, 2021.

[34] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[35] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[36] Yongkang Liu, Xuewei Qi, Emrah Akin Sisbot, and Kentaro Oguchi. Multi-agent trajectory prediction with graph attention isomorphism neural network. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2022.

[37] Wenjie Luo, Cheol Park, Andre Cornman, Benjamin Sapp, and Dragomir Anguelov. JFP: joint future prediction with interactive multi-agent modeling for autonomous driving. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2022.

[38] Osama Makansi, Özgün Çiçek, Yassine Marrakchi, and Thomas Brox. On exposing the challenging long tail in future prediction of traffic actors. In *Proceedings of the IEEE/CVF IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[39] Xiaoyu Mo, Yang Xing, and Chen Lv. Recog: A deep learning framework with heterogeneous graph for interaction-aware trajectory prediction. *arXiv preprint arXiv:2012.05032*, 2020.

[40] Sriram Narayanan, Ramin Moslemi, Francesco Pittaluga, Buyu Liu, and Manmohan Chandraker. Divide-and-conquer for lane-aware diverse trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[41] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S. Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. *arXiv preprint arXiv:2207.05844*, 2022.

[42] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, David Weiss, Benjamin Sapp, Zhifeng Chen, and Jonathon Shlens. Scene transformer: A unified multi-task model for behavior prediction and planning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[43] Seong Hyeon Park, Gyubok Lee, Jimin Seo, Manoj Bhat, Minseok Kang, Jonathan Francis, Ashwin R. Jadhav, Paul Pu Liang, and Louis-Philippe Morency. Diverse

and admissible trajectory forecasting through multimodal context understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[44] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. PRECOG: prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE/CVF IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[45] Luke Rowe, Martin Ethier, Eli-Henry Dykhne, and Krzysztof Czarnecki. FJMP: factorized joint multi-agent motion prediction over learned directed acyclic interaction graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[46] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[47] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. MTR++: multi-agent motion prediction with symmetric scene modeling and guided intention querying. *arXiv preprint arXiv:2306.17770*, 2023.

[48] Haoran Song, Di Luan, Wenchao Ding, Michael Yu Wang, and Qifeng Chen. Learning to predict vehicle trajectories with model-based planning. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2021.

[49] Qiao Sun, Xin Huang, Junru Gu, Brian C. Williams, and Hang Zhao. M2I: from factored marginal trajectory prediction to interactive prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[50] Bohan Tang, Yiqi Zhong, Ulrich Neumann, Gang Wang, Siheng Chen, and Ya Zhang. Collaborative uncertainty in multi-agent trajectory forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[51] Veronika Thost and Jie Chen. Directed acyclic graph neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[52] Ekaterina I. Tolstaya, Reza Mahjourian, Carlton Downey, Balakrishnan Varadarajan, Benjamin Sapp, and Dragomir Anguelov. Identifying driver interactions via conditional behavior prediction. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2021.

[53] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S. Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi-Pang Lam, Dragomir Anguelov, and Benjamin Sapp. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2022.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[55] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[56] Jingke Wang, Tengju Ye, Ziqing Gu, and Junbo Chen. LTP: lane-based trajectory prediction for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[57] Wenshuo Wang, Letian Wang, Chengyuan Zhang, Changliu Liu, and Lijun Sun. Social interactions for autonomous driving: A review and perspectives. *Foundations and Trends in Robotics*, 10(3-4):198–376, 2022.

[58] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1*, 2021.

[59] David Wu and Yunnan Wu. Air^2 for interaction prediction. *arXiv preprint arXiv:2111.08184*, 2021.

[60] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[61] Maosheng Ye, Tongyi Cao, and Qifeng Chen. TPCN: temporal point cloud networks for motion forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[62] Maosheng Ye, Jiamiao Xu, Xunnong Xu, Tongyi Cao, and Qifeng Chen. DCMS: motion forecasting with dual consistency and multi-pseudo-target supervision. *arXiv preprint arXiv:2204.05859*, 2022.

[63] Wenyuan Zeng, Ming Liang, Renjie Liao, and Raquel Urtasun. Lanercnn: Distributed representations for graph-centric motion forecasting. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[64] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kümmerle, Hendrik Königshof, Christoph Stiller, Arnaud de La Fortelle, and Masayoshi Tomizuka. INTERACTION dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*, 2019.

[65] Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. Dags with NO TEARS: continuous optimization for structure learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[66] Zikang Zhou, Zihao Wen, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Qcnext: A next-generation framework for joint multi-agent trajectory prediction. *arXiv preprint arXiv:2306.10508*, 2023.

[67] Zikang Zhou, Luyao Ye, Jianping Wang, Kui Wu, and Kejie Lu. Hivt: Hierarchical vector transformer for multi-agent motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[68] Dekai Zhu, Guangyao Zhai, Yan Di, Fabian Manhardt, Hendrik Berkemeyer, Tuan Tran, Nassir Navab, Federico Tombari, and Benjamin Busam. IPCC-TP: utilizing incremental pearson correlation coefficient for joint multi-agent trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

APPENDICES

Appendix A

Additional Evaluation Details

A.1 Miss Rate

For both Argoverse 2 and INTERACTION, we use the definition of a miss used in the INTERACTION dataset: a prediction is considered a "miss" if the longitudinal or latitudinal distance between the prediction and ground-truth endpoint is larger than their corresponding thresholds, where the latitudinal threshold is $\epsilon_{\text{lat}} := 1\,\text{m}$ and the longitudinal threshold is:

$$\epsilon_{\text{long}} := \begin{cases} 1, & \text{if } v \leq 1.4\,\text{m/s} \\ 1 + \frac{v-1.4}{11-1.4}, & \text{if } 1.4\,\text{m/s} \leq v \leq 11\,\text{m/s} \\ 2, & \text{otherwise,} \end{cases} \tag{A.1}$$

where v is the ground-truth velocity at the final timestep. We note that Argoverse 2 officially defines a miss as a prediction whose endpoint is more than 2 m from the ground-truth endpoint; however, we report all miss rate numbers in Section 5.2 using the miss rate definition in INTERACTION as it is a more robust measure of miss rate that takes into account the agent's velocity. For completeness, we report miss rate numbers for Argoverse 2 using the Argoverse 2 definition of a miss in Table A.1.

A.2 Collision Checker

To construct the interaction labels as described in Section 4.3.1, a collision checker is used to identify collisions between all pairs of timesteps in the future trajectories. We use the

Actors Evaluated	Model	$\text{SMR}_{\text{Argoverse2}}$
Scored	Non-Factorized	0.264
	FJMP	0.259
	Δ	0.005
All	Non-Factorized	0.259
	FJMP	0.257
	Δ	0.002

Table A.1: Non-Factorized Baseline vs. FJMP performance on Argoverse 2 SMR metric on the Argoverse 2 validation set. Δ denotes the difference in performance between FJMP and the Non-Factorized baseline.

collision checker provided with the INTERACTION dataset. At each timestep, the collision checker defines each agent by a list of circles, and two agents are defined as colliding if the Euclidean distance between any two circles' origins of the given two agents is lower than the following threshold:

$$\epsilon_C := \frac{w_i + w_j}{\sqrt{3.8}}, \tag{A.2}$$

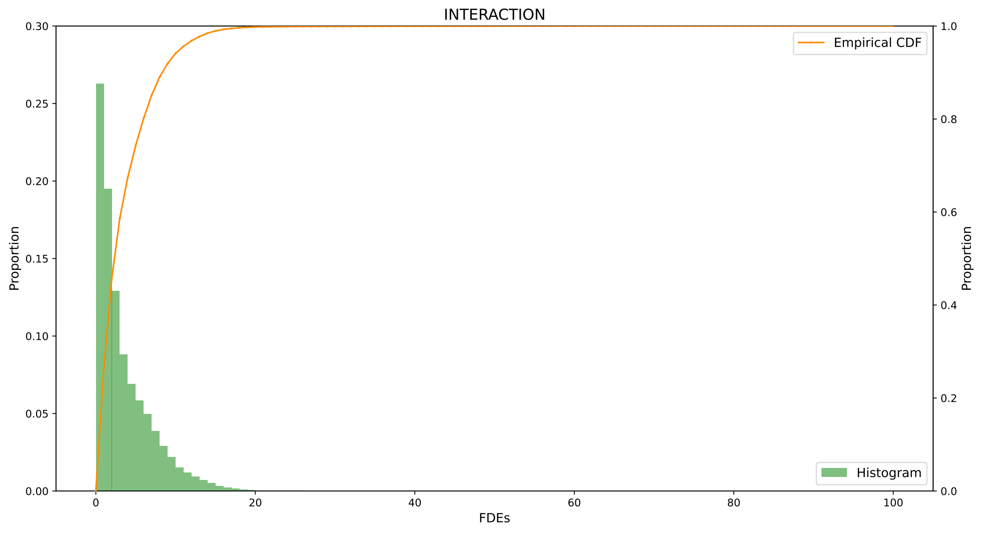
where w_i, w_j are the widths of agents i, j.

A.3 Constant Velocity Model

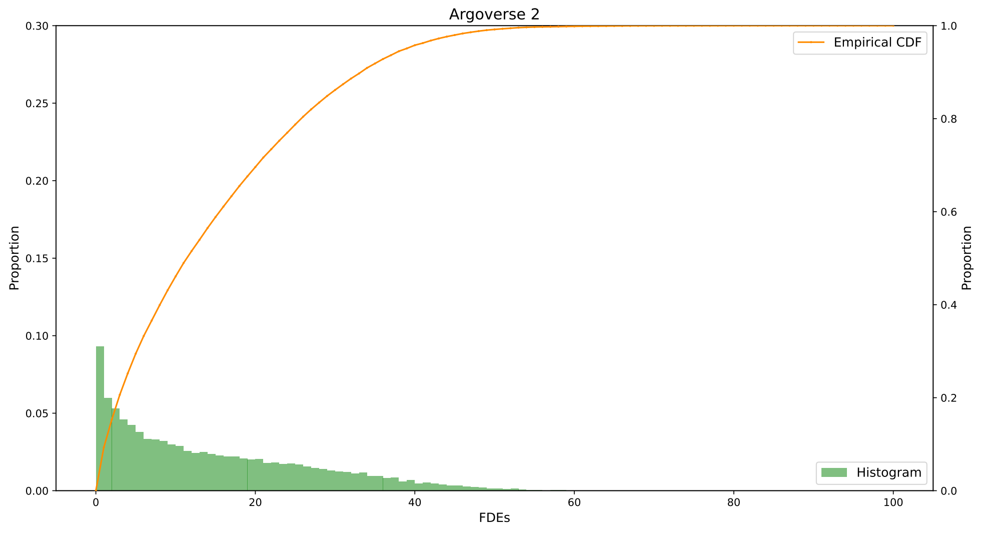
In Section 5.3, we identify the kinematically complex interactive agents in the datasets by filtering for agents that attain at least d m in FDE with a constant velocity model. An interactive agent is defined as an agent with at least one incident edge in the ground-truth interaction graph, where $\epsilon_I = 2.5\,\text{s}$, as is explained in Section 5.3. In this section, we describe the constant velocity model in more detail. The constant velocity model computes the average velocity over the observed timesteps and unrolls a future trajectory using the calculated constant velocity. Namely, the average velocity is calculated as:

$$\mathbf{v}_{\text{avg}} = \frac{1}{T_{\text{obs}}} \sum_{t \in [T_{\text{obs}}]} \mathbf{v}_t, \tag{A.3}$$

where $\mathbf{v}_t$ is the ground-truth velocity at timestep t. Using the constant velocity model, we calculate the agent-level FDE of all interactive agents in the INTERACTION and Argoverse 2 validation sets, respectively, where the FDE distributions are plotted in Figure A.1. We observe that a large proportion of the interactive agents have low FDE with a



(a)



(b)

Figure A.1: Histogram of FDEs on interacting agents in (a) the INTERACTION dataset, and (b) the Argoverse 2 dataset. The left y-axis corresponds to the histogram and the right y-axis corresponds to the empirical cumulative distribution function (CDF).

Dataset	d	Count
INTERACTION (112994)	0	50967
	3	21077
	5	13069
Argoverse 2 (248719)	0	37065
	3	29421
	5	26140

Table A.2: Number of *interactive* agents in the INTERACTION and Argoverse 2 datasets that attain at least d m in FDE with a constant velocity model. In parentheses, we include the total number of evaluated agents (interactive + non-interactive) in the respective validation sets.

Dataset	Edge Type	Edge Type Proportion
INTERACTION	`no-interaction`	0.955
	`m-influences-n`	0.037
	`n-influences-m`	0.008
Argoverse 2	`no-interaction`	0.973
	`m-influences-n`	0.015
	`n-influences-m`	0.013

Table A.3: Edge type proportions in the INTERACTION and Argoverse 2 training set interaction graphs with the FJMP labeling heuristic.

constant velocity model, especially in the INTERACTION dataset. By filtering out these kinematically simple agents, as is done in Section 5.5, we can assess the model's joint prediction performance on agents that are both interactive and kinematically complex. In Table A.2, we report the number of interactive agents in the INTERACTION and Argoverse 2 validation sets that attain at least d m in FDE, for $d = 0, 3, 5$. We note that $d = 0$ corresponds to the number of interactive agents in the respective validation sets.

A.4 Ground-truth Interaction Graph Statistics

Table A.3 reports the proportion of `no-interaction`, `m-influences-n`, and `n-influences-m` edges in the INTERACTION and Argoverse 2 training sets. Due to the severe class imbalance, we employ a focal loss when training the interaction graph predictor, as explained in Section 4.3.1.