Multi-Agent Hybrid Prediction in Autonomous Driving

by

Tiffany Yee Kay Yau

B.A.Sc. in Engineering Science, University of Toronto, 2022

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2025

Authored by:	Tiffany Yee Kay Yau Department of Aeronautics and Astronautics January 24, 2025
Certified by:	Brian C. Williams Professor of Aeronautics and Astronautics, Thesis Supervisor
Accepted by:	Jonathan P. How R.C Maclaurin Professor of Aeronautics and Astronautics Chair, Graduate Program Committee

Multi-Agent Hybrid Prediction in Autonomous Driving

by

Tiffany Yee Kay Yau

Submitted to the Department of Aeronautics and Astronautics
on January 24, 2025 in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

ABSTRACT

In autonomous driving, the hybrid task of predicting both high-level actions and low-level trajectories of human behaviour is fundamental to safe downstream decision-making. Much of the existing work in behaviour prediction tackle this problem without sufficiently modelling agent-agent interactions, limiting their ability to capture the full range of possible joint outcomes. Another key challenge in multi-agent prediction is the intractable prediction space that grows exponentially in the number of agents and duration of the prediction horizon. As a result, scalability is a major challenge. This thesis presents two approaches to address these challenges in multi-agent hybrid prediction.

In our first approach, we model interactions and address scalability by learning to factor the joint prediction distribution. We observe that agents do not interact with all other agents in the scene, but rather, there are groups that strongly interact. Therefore, we group agents and represent the high-level interaction outcomes of groups with discrete variables. We additionally assume that inter-group interactions are sparse and can be sufficiently represented with a directed acyclic graph. These assumptions enable us to factor the distribution into a product of factors, effectively reducing the prediction space, and providing an order in which to easily sample discrete values. We evaluate the performance of this method on a large-scale autonomous driving dataset and show that it exceeds prior methods in coverage of possible interaction outcomes by 24% to 48% on various multi-agent validation data splits, while maintaining state-of-the-art prediction error.

Our second approach represents agents in a traffic scene as a set of concurrent hybrid models and assumes a collision avoidance model of interactions, rather than learning the model from data like the first approach. Our method begins enumeration based on a simpler collision-agnostic prior distribution. Based on our factored representation, we determine the next best assignment to the prior. We extract bounding conflicts to correct the prior and increasingly reduce the error between the distribution used by enumeration and our collision-aware posterior distribution. Our experiments show that enumeration using A* with bounding conflicts (A*BC) is faster than A* and is therefore better at addressing scalability. In terms of prediction metrics, we find that our collision-aware posterior performs worse than the collision-agnostic prior and suggest future directions for improvement.

Thesis supervisor: Brian C. Williams
Title: Professor of Aeronautics and Astronautics

Acknowledgments

The past two and a half years have been an incredible journey, and completing this thesis would not have been possible without the people who have supported me along the way.

First and foremost, I want to express my sincerest gratitude to my advisor, Professor Brian Williams. Brian's guidance, encouragement, and insightful feedback have been instrumental in shaping this thesis. He taught me how to present my research as one coherent and intuitive story. He has always taken the time to explain concepts to me and give detailed feedback on my writing. Brian is also one of the most hospitable people I have ever met. Many unforgettable MERS memories were made at the Thanksgiving and barbecue dinners at his house. In addition, Brian goes above and beyond his role as a research advisor to genuinely care for the well-being of his students, and for this I am truly grateful.

It has been an incredible privilege to be mentored and advised by Dr. Guy Rosman. I really appreciate his patient guidance, deep insights, and thoughtful and constructive feedback. From him, I have learned what it means to be a methodical, detail-oriented, and rigorous researcher. I will always remember how he has challenged me to perform research with high standards. Moreover, I am very grateful for his time and availability, even on his days off. A mentor of his brilliance and dedication is truly rare to find. Guy's extensive knowledge about so many different fields has been an inspiration to me and has made an immeasurable impact on this thesis.

I would like to thank my fellow MERS members – past and present – who helped me along the way. To Cyrus, Marlyse, Sungkweon, and Yuening – thank you all for your wise advice as senior students. Cyrus helped me get up to speed on my work early on and stuck around even after graduating to give advice and bridge the transition. Marlyse mentored me and met with me frequently. She was always available to discuss ideas and to be a sounding board. Her contagious positivity has kept me sane through tough times. To Amy, Annabel, Ingrid, Shashank - thank you for the caffeine and sugar breaks, collection of furry stuffed animals, shared dark humour, and lessons on how to distinguish between Canadian and American accents. To all MERS members – Allegra, Amy, Aneesh, Annabel, Anoop, Cameron, Cecelia, Dominique, Ingrid, Jake, Lucian, Marlyse, Meng, Morgan, Shashank, Viraj, Weiqiao, Yuening – thank you for giving me two and a half memorable years.

I would not be here without the tremendous amount of encouragement and help from my MIT Graduate Christian Fellowship (GCF) family. They were a shoulder to lean on and an outlet from stress and uncertainty. Their support uplifted me when I felt utterly weak and helpless during the difficult times. The happy moments and time that we spent cooking, eating, watching movies, and working together, are all memories that I cherish dearly. GCF, I thank God each time I think of you.

Most of all, I would like to express my deepest gratitude to my parents, Eddy and Eva, and, my brother and sister-in-law, Nathan and Claudia, in Toronto. I am thankful for their phone calls, visits to Boston, and the time we spent together doing puzzles. I would not have made it to this point without my family's unceasing love and support.

Special thanks to Toyota Research Institute for generously sponsoring this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In various applications of human-robot interaction, predicting the behaviours of humans is important to enable robots to act safely. For instance, in service robotics, it is important to predict the behaviour of surrounding humans so that the robot can interact appropriately and navigate safely. In autonomous driving, behaviour prediction is an essential task that informs the autonomous car of possible future motions of surrounding agents (e.g., other vehicles, pedestrians, and cyclists). This allows the autonomous car to safely navigate its dynamic environment. This thesis focuses on multi-agent behaviour prediction in autonomous driving.

There are several crucial considerations when predicting behaviour. This thesis aims to address two key challenges.

1. **Scalability**. The task of behaviour prediction becomes increasingly challenging as the number of agents increases. This is because the space of possible futures grows exponentially and becomes intractable. As a result, an increasing number of joint prediction samples and time are required to accurately approximate the joint distribution. We address this challenge of scalability by factoring the joint distribution and effectively reducing the prediction space.

2. **Interaction Coverage**. In complex applications like driving, agents are highly interactive. Each individual's future motion is strongly impacted by their surrounding agents. In order to capture the possible trajectories that agents may take and better inform downstream planning, it is important to model the interactions and cover a diverse set of interaction outcomes in joint predictions.

Further details on how we address these challenges are provided in Section 1.2.

1.1 Behaviour Prediction as a Hybrid Problem

Human behaviour consists of hybrid discrete and continuous structure, where discrete factors are high-level actions and low-level motion are the continuous states. This hybrid structure has been explored in video activity recognition, where low-level human pose estimation and high-level activity recognition are closely linked [1]. Beyond activity recognition, hybrid representations have been used for human activity forecasting from video [2], where low-level trajectory and high-level action are jointly modelled in various applications including predicting household activities [3] and sports-related behaviours [4]. Understanding the breakdown of discrete factors informs us of the underlying structure of behaviours and provides us with a way to accurately model and predict it. For example, in sports, when part of a team is executing a certain play, the trajectories that the players take will correspond to that play. Being able to understand the breakdown (i.e., who is involved in the play) and predict the high-level behaviour (i.e., the play) helps to predict low-level motion (i.e., the trajectories).

The prediction problem in autonomous driving has also been viewed as a hybrid discrete-continuous problem, where the goal is to predict both the high-level discrete behaviours and low-level continuous states. Discrete behaviours could represent interaction modes between multiple agents (such as explicit yielding and passing or leading and following patterns [5]–[7] or implicit latent variables [8]–[10]) or individual maneuvers [11], [12], trajectory goal points [13]–[19], anchor trajectories that correspond to modes of the future distribution [20], [21], embeddings that are latent representations of anchors [22], or lanes to follow [23], [24]. The low-level states of agents represent their continuous-valued trajectories in a global coordinate frame. This thesis follows this line of reasoning and addresses behaviour prediction as a hybrid problem. We leverage several tools from the approximate inference and model-based reasoning communities [25]–[30] and combine them with the expressive power of deep neural networks to accomplish hybrid prediction.

1.2 Approach

1.2.1 Learned Factorization and Sampling for Multi-Agent Prediction

In our first approach, we learn to factor the joint distribution using neural networks. High-level interaction outcomes are represented by discrete variables and individual motion by continuous variables. We learn to factor the joint distribution over discrete variables, and assume that continuous motion of agents are independent when conditioned on the discrete variables. For this method, we take inspiration from factorization in Bayesian inference and model-based reasoning [25]–[27], [31]–[33] and apply them to a neural network-based predictor.

We learn to group closely interacting agents together and represent each group as a factor. Each group has a high-level interaction outcome that corresponds to one discrete variable. Examples of different high-level interaction outcomes could be "A yields to B", "B follows A", or the order in which agents pass through a shared region on the road (e.g., "pedestrian B crosses the street between vehicles A and C" or "pedestrian B crosses after vehicles A and C pass"). We assume that interactions across groups are sparse (i.e., not every group will interact with all other groups) and therefore sufficiently represented by a directed acyclic graph (DAG). We learn to output the DAG, where nodes represent groups and directed edges represent interactions where the source node is the group that influences the discrete interaction outcome of the target node.

The DAG structure factors the joint distribution over discrete variables in a way that enables us to sample discrete variables in order, where predecessors are sampled first and

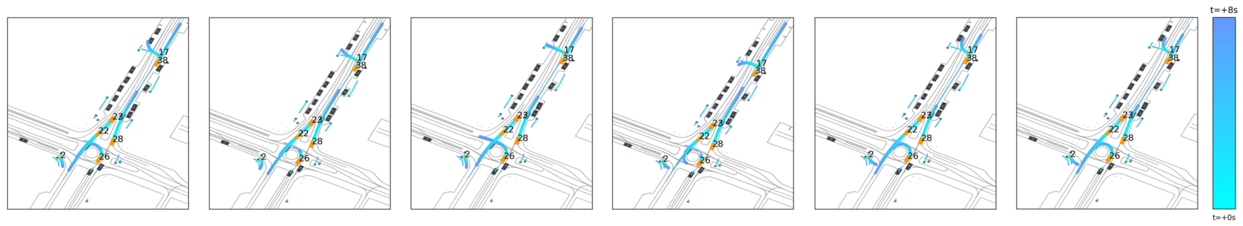


Figure 1.1: An example of joint predictions made by our learned factorization and sampling-based method.

successors are conditioned on the predecessor samples. The sampled values of the discrete variables, along with the groups, are inputted to a neural network decoder to form the final joint trajectory sets.

Our experimental results show that this method improves coverage of interaction outcomes by 24% to 48% on validation datasets of various interaction complexity. We also demonstrate an improvement in sample efficiency by over 33% compared to prior work, highlighting strong scalability. Examples of joint prediction samples from this method are shown in Figure 1.1. We discuss this method in more detail in Chapter 4.

1.2.2 Collision-Aware Best-First Enumeration for Multi-Agent Prediction

Our second approach assumes a model of traffic scenes that represents agents as concurrent hybrid models. Discrete variables represent individual high-level maneuvers and continuous variables represent the continuous states conditioned on the discrete mode. We assume agents exhibit collision avoidance behaviour during interactions and discrete modes are independent given collision-free future behaviour. With this approach, the assumed hybrid model and interaction model are explicitly defined beforehand, rather than learned from data. Additionally, we explore approximate inference using best-first enumeration for scaling to multi-agent prediction, as opposed to sampling, to capture the maximum joint probability mass with a limited number of samples.

We aim to approximate a joint collision-aware posterior distribution using best-first enumeration to assign values to discrete mode variables. We begin enumeration by assigning the best value to a factor according to a simpler collision-agnostic prior distribution. We then leverage our factored, concurrent representation to extend the partial assignment with the best assignment of a second factor. We continue this process of assigning values to factors until we reach a full assignment.

When we reach a full assignment, we compute the joint collision-aware posterior and compare this probability to the collision-agnostic prior. We extract bounding conflicts [34], which we use to correct our estimation of the posterior. As the search process continues, the bounding conflicts are used to re-prioritize regions of the search space and avoid expanding unlikely assignment until later. At each search node corresponding to a full assignment, we extract bounding conflicts to increasingly reduce the error between the prior and the posterior.

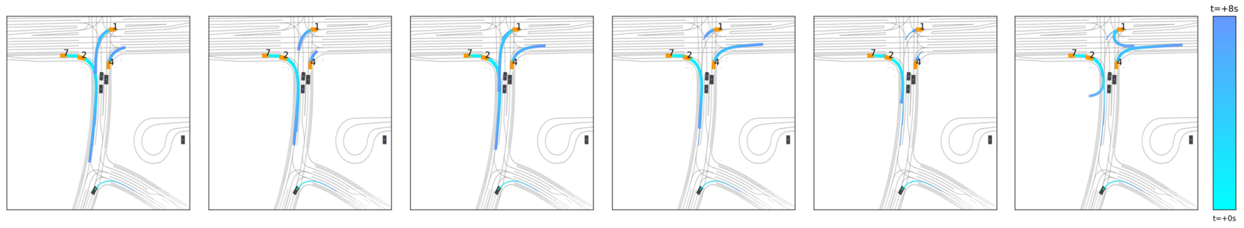


Figure 1.2: An example of joint predictions made by our collision avoidance and enumeration-based method.

Our experiments show that best-first enumeration using A* with bounding conflicts (A*BC) [34] is faster than using A*. An example of an output from this method is shown in Figure 1.2. Chapter 5 discusses further details of this method.

1.3 Thesis Outline

The remainder of this thesis is structured as follows. Chapter 2 will formally describe the multi-agent prediction problem. Chapter 3 describes at a high level the two approaches in this thesis. Chapter 4 presents the details and experimental results of our learned factorization and sampling method. We then discuss in Chapter 5 our collision-aware best-first enumeration method. We compare the prediction performance and scalability of these two methods in Chapter 6. This thesis concludes with Chapter 7, which presents a summary of contributions and future work.

Chapter 2

Problem Statement

Here, we present the problem statement for multi-agent prediction. Future behaviour of agents is commonly approximated with a set of weighted joint trajectory samples to capture their probabilistic and multi-modal nature [12], [35]. Due to the computational cost of risk assessment for each prediction sample, real-time systems can only viably process process a small set of trajectory samples. As a result, it is desirable for the samples to represent a diverse set of possible futures and provide good coverage of possible outcomes.

2.1 Inputs

The multi-agent trajectory prediction problem takes as input the following:

1. **Past observations**, $O = \{\mathbf{o}_{t,n} : -T_p \leq t \leq 0, 1 \leq n \leq N\}$, which consist of the observed trajectories of N agents of interest over a past time horizon T_p

2. **Scene context**, C, which consist of environmental information of each agent's surroundings, such as neighbouring objects and map coordinates or polylines.

3. **A single-agent trajectory predictor**, P, that takes O and C and outputs a set of K weighted continuous-state trajectories for each agent in the scene. These weighted trajectory sets represent the multi-modal *marginal* future distribution $p(\mathbf{x}_{1:T_f,i}|O,C)$ for each agent $i = 1, ..., N$.

4. **A hybrid model**, H, that represents a multi-agent traffic scenario with a set of continuous and discrete variables and a model of interactions. Continuous variables represent the continuous motion of agents in the scene. Discrete variables can represent different behaviour modes (e.g., individual maneuvers or interaction patterns) and vary depending on the methodology.

An example of the inputs of the multi-agent trajectory prediction problem is shown in Figure 2.1. In this example, the inputs are:

1. State variables, which are the observed past trajectories of each agent. These are represented by the dotted lines.

2. Scene context, including lane lines and road boundaries, represented by dashed and solid lines, respectively.

3. A single-agent trajectory predictor illustrated by the solid black rectangle, which uses the past trajectories and scene context to produce a set of marginal future trajectories for each agent. In this thesis, we leverage a method called MTR [15], which is described in Section 3.5.

4. There is no hybrid model for this example.

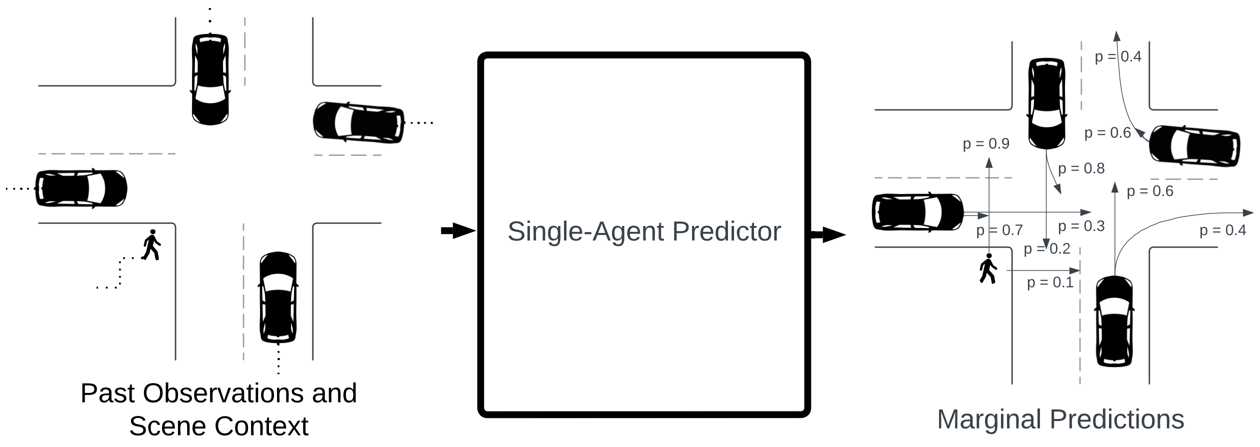


Figure 2.1: An example of the inputs to the multi-agent prediction problem. The past observed states of the agents are denoted by the dotted lines. Solid lines represent the road boundaries and dashed lines represent lanes. Arrows represent marginal predictions, and associated numbers are their corresponding weights.

2.2 Outputs

The goal of multi-agent prediction is to estimate the *joint* probability distribution of future trajectories of all agents in the scene, denoted by

$$p(\{\mathbf{x}_{1:T_f,i}\}_{i=1}^{N}|O, C, P, H) \tag{2.1}$$

Similar to single-agent prediction, this distribution is inherently multi-modal, as joint behaviour can unfold into multiple high-level outcomes. For example, high-level outcomes can represent interaction outcomes (e.g., whether two interacting agents result in "A yields to agent B" or "B yields to A") or individual maneuvers (e.g., A turns right while B goes straight). This distribution does not have a closed-form expression. Therefore, the output of this problem is a weighted set of K *joint* trajectory samples $S = \{(\mathbf{x}^{(k)}, w^{(k)})\}_{k=1}^{K}$. Each joint trajectory sample $\mathbf{x}^{(k)} = \{\mathbf{x}_{1:T_f,i}^{(k)}\}_{i=1}^{N}$ consists of one predicted trajectory for each agent of interest for all time steps in the future horizon T_f. An example of the outputs of the multi-agent trajectory prediction problem is shown in Figure 2.2. The outputs consist of joint trajectory predictions and associated weights, which represent prediction likelihoods.

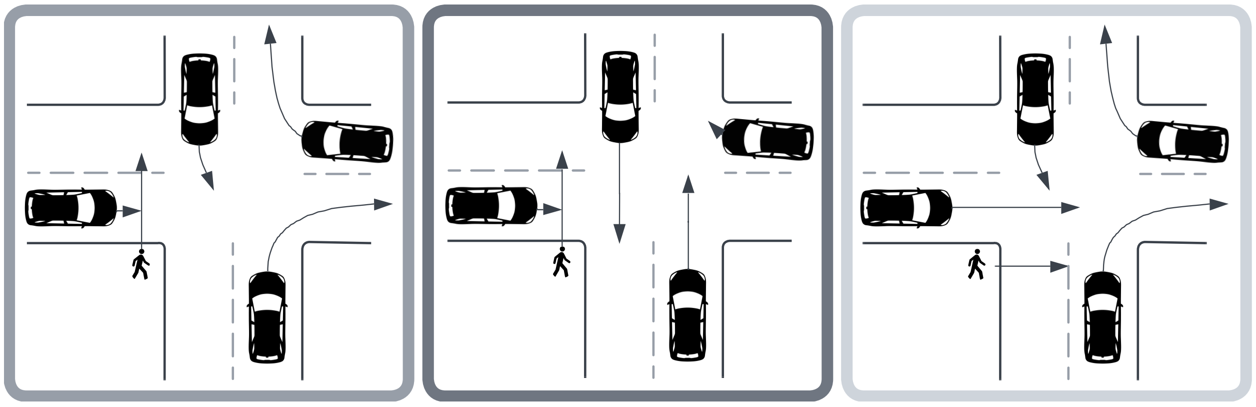


Figure 2.2: An example of the outputs of a multi-agent prediction method. Each square represents a different joint prediction sample. Agents' futures are denoted by solid arrows. The weight of each joint prediction is represented by the shade of the square border.

2.3 The Multi-Agent Prediction Problem

We define the multi-agent prediction problem as follows:

Definition 2.1 (Multi-Agent Prediction Problem)**.** The multi-agent prediction problem is to learn a function $f(O, C, P, H)$ that maps the inputs – past observations O, context C, single-agent predictor P, and hybrid model H – to a set of joint prediction samples S that minimizes the cost $\mathcal{J}(S, \{\mathbf{x}^{gt}_{1:T_f,i}\}_{i=1}^{N})$ between the joint prediction samples and the ground truth future trajectories $\{\mathbf{x}^{gt}_{1:T_f,i}\}_{i=1}^{N}$.

In this thesis, we use a combination of neural networks and model-based reasoning to approximate the mapping. We first train the single-agent predictor P on a dataset with a set of training scenarios containing O, C, and $\{\mathbf{x}^{gt}_{1:T_f,i}\}_{i=1}^{N}$. O and C are inputs to P during training, and P outputs weighted marginal trajectories for each agent $\{S_i\}_{i=1}^{N}$ where $S_i = \{(\mathbf{x}_i^{(k)}, w_i^{(k)})\}_{k=1}^{K}$. Using $\{\mathbf{x}^{gt}_{1:T_f,i}\}_{i=1}^{N}$, we train P to minimize a single-agent prediction cost that optimizes accuracy. Specifically, we leverage MTR [15] as our single-agent predictor P due to its strong performance on single-agent prediction. Further details on the neural network architecture and training cost are provided in Section 3.5.

2.4 Modelling Interactions for Multi-Agent Prediction

In multi-agent prediction, various interactions occur between agents in a scenario. These interactions must be considered in order to model the traffic scene in a way that supports accurate prediction. In this thesis, we explore two different ways to model interactions, and incorporate this into our hybrid model input H.

In Chapter 4, we address hybrid modelling as a deep learning problem by learning to obtain the hybrid model H for each traffic scenario. To do this, we train a neural network to identify interacting agents in a scene. We represent the high-level interaction outcome of each interacting group using a discrete variable. We additionally learn to uncover the dependencies across discrete variables in order to factor the joint discrete distribution. Intuitively, the discrete variable of one group is dependent on the discrete variable of a second group if the interaction outcome of the first group depends on that of the second group. Inspired by prior work in DAG approximation in belief propagation [31]–[33], we assume that interactions between groups can be approximated by a DAG, and use this to factor the discrete distribution in a way that enables efficient inference. In addition to the discrete variables, continuous variables represent each individual's continuous motion. Using the learned hybrid model H, along with O, C, and P, we train a neural network for the mapping $f(O, C, P, H)$ from inputs to joint prediction samples S.

In Chapter 5, we explore multi-agent prediction based on the idea that agents avoid collision. As a result, when agents interact, the future trajectory rollouts are unlikely to involve collisions. Using interaction labels provided in the training dataset, we define a proxy for collision likelihood based on joint trajectories. We compute the time-to-collision between all pairs of interacting agents and construct a kernel density estimate of the time-to-collision. Given a joint prediction candidate, we compute the time-to-collision with the trajectories and derived velocities and use the estimated probability density as a proxy for the likelihood of the joint prediction. The idea is that less likely and lower times-to-collision correspond with unlikely joint maneuvers. In addition to collision avoidance as an assumed model, we model the traffic scene with a set of concurrent hybrid agent models, where agents' individual maneuvers are represented by discrete variables and the corresponding continuous motions are represented by continuous variables. The concurrent hybrid agent models and collision avoidance model are used as input H to the multi-agent prediction problem. In this method, apart from learning the kernel density estimate of time-to-collision, the factored hybrid model (i.e., conditional independence of concurrent behaviours) is fully assumed beforehand instead of learned. Together with other input variables O, C, and P, we obtain the joint prediction samples S from the mapping $f(O, C, P, H)$.

Our two approaches solve variants of the same multi-agent prediction problem, as presented in Section 2.3. However, they differ in the input hybrid model H. In Chapter 4, we learn to identify with a neural network the hybrid model for each traffic scene. The discrete variables represent interaction outcomes of groups of agents. On the other hand, our approach in Chapter 5 does not learn the hybrid model and instead assumes a set of conditionally independent concurrent hybrid agent models given collision avoidance behaviour, where the discrete variables represent maneuvers of individual agents. Our two approaches also differ in inference technique, with Chapter 4 performing inference using sampling and Chapter 5 using best-first enumeration.

Chapter 3

Approach

3.1 Behaviour Prediction as a Hybrid Problem

In this thesis, we view the multi-agent driving scenario as a hybrid discrete-continuous system and reason qualitatively about high-level discrete behaviours of traffic agents using inspiration from hybrid methods in the model-based reasoning community [27]–[29]. Prior works in prediction have represented high-level qualitative states in a traffic scene as a finite set of single-agent maneuvers [11], [12], trajectory goal points [13]–[19], anchor trajectories [20], [21], embeddings [22], or lanes to follow [23], [24]. In addition, discrete variables can represent interactive behaviours [5]–[7]. In this thesis, we present two methods – one that primarily focuses on discrete modes as high-level multi-agent interactive behaviours and one that leverages single-agent modes, specifically goal-based maneuvers, to enumerate joint trajectories from marginal trajectories.

Addressing prediction as a hybrid problem allows us to draw on insights from hybrid inference [27]–[29] to support accurate and scalable prediction. This thesis explores two complementary ways of doing so. In the first method, we take inspiration from factored inference [25], [26] and hybrid sampling [27] to enable better coverage of discrete modes with fewer samples. Discrete modes represent high-level interaction outcomes of multiple agents. We provide an overview in Section 3.3. In the second method, we leverage an efficient best-first enumeration technique [34] to form joint trajectory predictions directly using marginal trajectories from the output of a single-agent predictor. In this method, marginal trajectories correspond to single-agent intention points (which represent trajectory endpoints of distinct maneuvers), and the selection of a particular trajectory represents the discrete choice. We provide an overview of this method in Section 3.4.

Both methods that we present are general frameworks that can be applied on top of different single-agent prediction methods. Various types of neural networks have been introduced to encode past observations and context and generate predictions, including RNNs [35]–[37], CNNs [20], [37]–[39], GNNs [40], [41], and Transformers [15], [42]. In this thesis, we leverage the Transformer-based method in [15], known as MTR, as our backbone network due to its strong performance. We outline the necessary details about MTR in Section 3.5.

3.2 Overview

In this thesis, our approaches to the multi-agent prediction problem defined in Chapter 2 consist of the general algorithm presented in Algorithm 1 after all neural network-based components of P and H have been trained. Specifics on training are provided where relevant in Section 3.5 and Chapter 4.

Algorithm 1 Our General Algorithm for Multi-Agent Prediction

```
1: function MULTI-AGENT-PREDICTION($O, C, P, H$)
2:     $S_{marg} \leftarrow \{\}$
3:     for $i = 1, ...N$ do
4:         $S_i \leftarrow P(O, C)$
5:         $S_{marg} \leftarrow S_{marg} \cup \{S_i\}$
6:     end for
7:     $S \leftarrow$ SELECT-JOINT-TRAJECTORIES($S_{marg}$)
8:     return $S$
9: end function
```

Generally, our two approaches involve leveraging a trained single-agent predictor P to obtain marginal trajectory predictions S_i for all agents $i = 1, ..., N$ (lines 3 to 6). In practice, this loop is parallelized as the agent dimension is batched and the neural network forward pass is done on a GPU, but we illustrate it as a loop here for clarity. Then, the set of marginal trajectory samples for each agent $S_{marg} = \{S_i\}_{i=1}^{N}$ is used to form the joint trajectory samples S (line 7). The function `Select-Joint-Trajectories` differs between our two methods, with one method performing the selection based on discrete samples that are fed to a neural network (Section 3.3) and the second method using best-first enumeration (Section 3.4).

3.3 Learned Factorization and Sampling for Multi-Agent Prediction

In the first method that we present in this thesis, we address both scalability and interaction coverage. We handle hybrid prediction by factoring the joint distribution to effectively reduce the prediction space. This allows us to cover the prediction space more easily using fewer prediction samples [26], and therefore scale better to joint prediction for larger numbers of agents. In multi-agent prediction, it is important for joint prediction samples to cover diverse interaction outcomes so that the samples reflect the full range of possible interactive future behaviours. Therefore, in our method, we focus on achieving diversity of high-level interaction outcomes, and present a way to factor the joint discrete distribution over these outcomes.

We observe that agents do not interact with every other agent in the scene, but only a few. Within smaller groups, agents can closely interact with each other. We view these tightly coupled behaviours as a group whose high-level interaction outcome can be represented by a discrete variable. Examples of high-level interaction outcomes are: "agent A yields to B" or "agent B follows A". It is also possible for the interaction mode of one group to affect another. We observe that inter-group dependencies are likely sparse and can be sufficiently approximated as a directed acyclic graph (DAG), where vertices represent groups and predecessors are groups that impact successors groups' interaction outcomes. Therefore, our method is also designed to capture these dependencies among discrete variables and exploit the directed acyclic structure for simple and efficient sampling.

In our method, a neural network first groups agents that are closely interacting. This allows us to determine the agents whose interaction outcome corresponds to a discrete variable, and agents can be assigned to more than one group. Second, a neural network identifies the underlying DAG structure among these variables. The nodes of the graph are the discrete variables representing interacting groups, and the edges are interactions with source nodes as those that are inferred first and influence the outcomes of target nodes. We assume that interactions across groups are weak and an acyclic representation is sufficient to represent them. Based on the DAG, we can obtain an order in which to easily sample the variables so that predecessors are sampled before successors. This allows us to condition on predecessors according to the DAG.

Using the order, we sample discrete values representing group interaction outcomes, conditioning on the DAG predecessors for each variable. At each discrete variable, the conditional distribution from which we sample is output by a neural network that takes in the predecessors' sampled values. Finally, conditioned each agent's corresponding discrete variables and groups, past observations, and context, we use a neural network to select joint trajectories. Of note, we assume that knowledge of groups' high-level interaction outcomes provided by the discrete variables is sufficient to reason about joint future behaviour, and therefore agents' future trajectories are independent when conditioned on the discrete variables. We repeat the ordered discrete variable sampling and joint trajectory selection K times to obtain K joint continuous-state trajectory prediction samples with potentially different high-level interaction outcomes. Figure 3.1 shows an overview of this method. Detailed methodology, experiments, and results are presented in Chapter 4.

This method is shown in Algorithm 2, with extensions from Algorithm 1 highlighted in blue. In this algorithm, our method operates on the latent space of the single-agent predictor P, so we show P into two parts – P_{enc} and P_{dec} which represent the encoder and decoder of the single-agent predictor, MTR [15], and correspond to the descriptions in Sections 3.5.3 and 3.5.4, respectively.

3.4 Collision-Aware Best-First Enumeration for Multi-Agent Prediction

The second method that we present focuses on addressing the scalability challenge of multi-agent prediction. We model traffic scenarios as a set of concurrent hybrid agent models. We assume conditional independence given collision-free behaviour to factor the joint distribution, where a factor corresponds to a single agent. We observe that agents tend to behave in a way that avoids collision [36], [43]–[45]. We therefore focus on this attribute in our modelling of interactions. We use a best-first enumeration algorithm, A* with bound-

Algorithm 2 Learned Factorization for Multi-Agent Prediction

```
 1: function LEARNED-FACTORIZATION-MULTI-AGENT-PREDICTION($O, C, P, H$)
 2:     $S_{marg} \leftarrow \{\}$
 3:     $F \leftarrow [\,], Z \leftarrow \{\}$
 4:     for $i = 1, \ldots N$ do
 5:         $F_i \leftarrow P_{enc}(O, C)$
 6:         $F \leftarrow append(F, F_i)$
 7:         $S_i \leftarrow P_{dec}(F_i)$
 8:         $S_{marg} \leftarrow S_{marg} \cup \{S_i\}$
 9:     end for
10:     $F_G \leftarrow$ GROUP-AGENTS($F$)
11:     $A \leftarrow$ DETERMINE-DAG-STRUCTURE($F_G$)
12:     $order \leftarrow$ ORDER-DISCRETE-VARIABLES($A$)
13:     for $k = 1, \ldots, K$ do
14:         $\mathbf{z}^{(k)} \leftarrow$ SAMPLE-DISCRETE-VARIABLES($F_G, order$)
15:         $Z \leftarrow Z \cup \{\mathbf{z}^{(k)}\}$
16:     end for
17:     $S \leftarrow$ SELECT-JOINT-TRAJECTORIES($S_{marg}$, $F, F_G, Z$)
18:     return $S$
19: end function
```

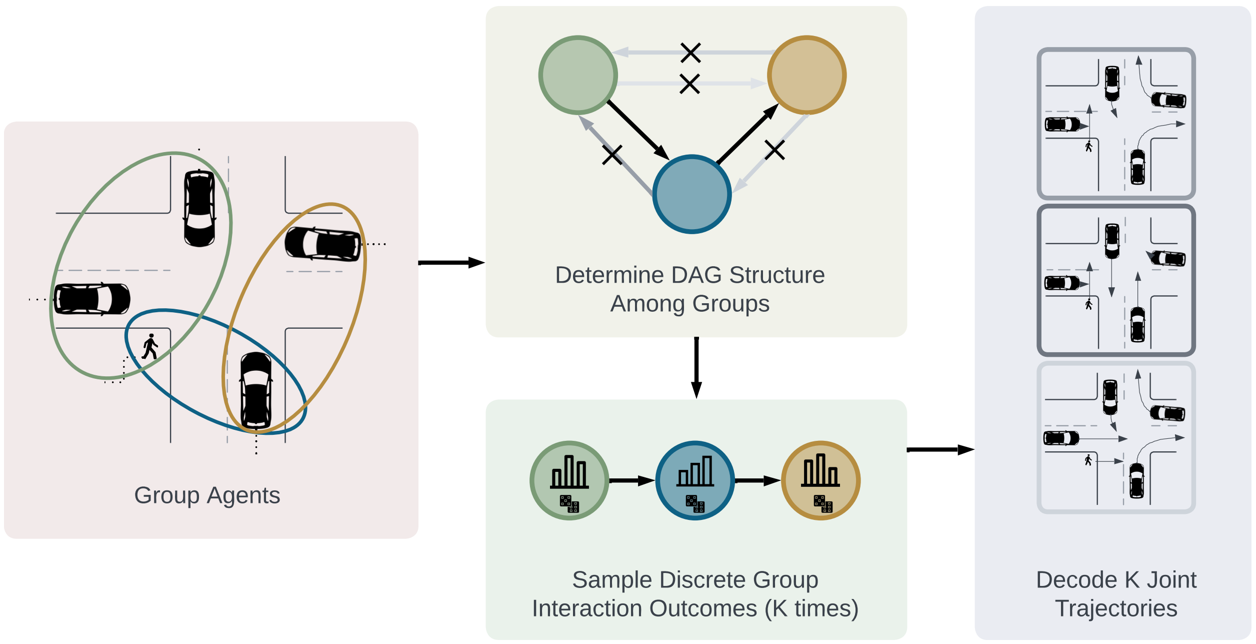

Group Agents
Determine DAG Structure Among Groups
Sample Discrete Group Interaction Outcomes (K times)
Decode K Joint Trajectories

Figure 3.1: Overview of joint prediction by learning to factor. We train a neural network to factor the joint distribution of discrete interaction outcomes by grouping agents (with each group representing a factor) and outputting the directed acyclic interaction graph between groups of agents. We then sample for each factor following the DAG topology and then decode joint predictions based on the samples.

ing conflicts (A*BC) [34], to assign values to discrete variables representing agents' future behaviour modes.

This method generates a best assignment for one factor based on a prior collision-agnostic distribution, where the assignment represents a discrete high-level maneuver corresponding to a continuous trajectory and the factor corresponds to a single agent. Subsequent assignments to other factors extend upon assignments to previous factors, until a full assignment (i.e., maneuvers to every agent) is reached. Upon reaching a full assignment, compute the posterior collision-aware probability. We learn bounding conflicts as a correction step to increasingly reduce the error between the collision-agnostic and collision-aware distributions.

To compute collision likelihood given assigned trajectories, we compute the time-to-collision of every pair of agents in the scene. We estimate the time-to-collision likelihood using a kernel density estimate on the training dataset. We base the probability of collision on the time-to-collision likelihood. Intuitively, if two agents come close to each other with a small time-to-collision, and the time-to-collision has a low likelihood according to the kernel density estimate, it reflects a higher probability of collision. We use the collision likelihood to compute bounding conflicts associated with sets of assigned trajectories. This way, we can keep track of the error between the collision-agnostic prior and collision-aware posterior to better order search nodes during enumeration.

In this thesis, we use [15] as the single-agent predictor that provides the prior collision-agnostic distribution. Our method can be applied to any single agent predictor that outputs multiple weighted trajectory predictions per agent. An overview of this method is shown in Figure 3.2 and deviations from Algorithm 1 are highlighted in blue in Algorithm 3. We apply additional trajectory selection to the outputs of the neural network P, as done by [15]. This post-processing step, which we denote as P_{NMS}, is known as non-maximum suppression and helps narrow down model outputs to fewer prediction samples, described in Section 3.5.5. Further details and experimental results are presented in Chapter 5.

Algorithm 3 Best-First Enumeration for Multi-Agent Prediction

```
1:  function Best-First-Enumeration-Multi-Agent-Prediction($O, C, P, H$)
2:      $S_{marg} \leftarrow \{\}$
3:      for $i = 1, ...N$ do
4:          $S_i \leftarrow P(O, C)$
5:          $S_i \leftarrow P_{NMS}(S_i)$
6:          $S_{marg} \leftarrow S_{marg} \cup \{S_i\}$
7:      end for
8:      $S \leftarrow$ Select-Joint-Trajectories($S_{marg}$, $H$)
9:      return $S$
10: end function
```

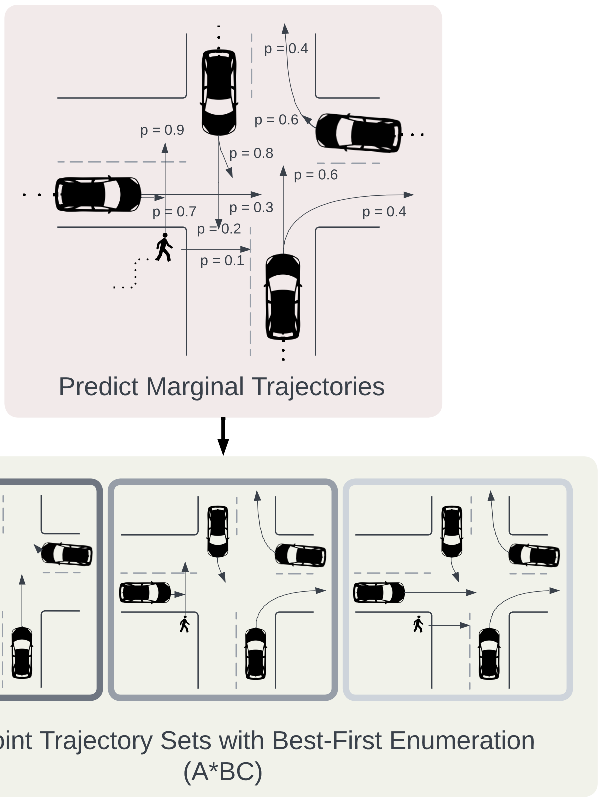

p = 0.4
p = 0.6
p = 0.9
p = 0.8
p = 0.6
p = 0.7
p = 0.3
p = 0.4
p = 0.2
p = 0.1
Predict Marginal Trajectories
Form Joint Trajectory Sets with Best-First Enumeration (A*BC)

Figure 3.2: Overview of joint prediction by combining marginal predictions through best-first enumeration. We use the marginal predictions output by the single-agent predictor and enumerate joint trajectories in best-first order for a collision-aware posterior distribution. We begin enumeration using the prior collision-agnostic distribution and use A*BC to increasingly reduce the error between the prior and the posterior.

3.5 Preliminaries: Single-Agent Prediction Using Motion TRansformer (MTR)

In this section, we introduce the single-agent predictor that we use in this thesis, MTR [15]. In reference to the problem statement in Chapter 2, MTR is the model that we use as the input P to the multi-agent prediction problem. We describe the architecture and training details in the remainder of this section.

3.5.1 Inputs and Outputs

MTR [15] is a Transformer-based [46] method for trajectory prediction. MTR takes as input the observations O and scene context C and outputs a set of weighted trajectories per agent of interest. That is, for the i^{th} agent of interest, MTR outputs $S_i = \{(\mathbf{x}^{(k)}_{1:T_f,i}, w^{(k)}_i)\}_{k=1}^{K}$, where $w^{(k)}_i = p(\mathbf{x}_{1:T_f,i} = \mathbf{x}^{(k)}_{1:T_f,i} | O, C)$ and $\mathbf{x}_{1:T_f,i}$ are the predicted continuous states at each timestep in the prediction horizon T_f for agent i.

The authors also propose a simple way to convert these *marginal* predictions into *joint* predictions based on a factorization of the joint distribution that assumes conditional independence given only the inputs O and C. MTR's factorization of the joint distribution is as follows

$$p(\{\mathbf{x}_{1:T_f,i}\}_{i=1}^{N} | O, C) = \prod_{i=1}^{N} p(\mathbf{x}_{1:T_f,i} | O, C) \tag{3.1}$$

The products of the marginal probabilities are used in an exhaustive search to find the top K predictions from K^N possible combinations of N agents of interest. However, it should be noted that MTR is designed primarily to achieve good single-agent prediction performance, and evaluates multi-agent prediction metrics only on two-agent scenarios. We refer readers to [15] for further details on the results.

3.5.2 Architecture Overview

An overview of the MTR architecture is shown in Figure 3.3. We summarize the key details of the MTR encoder and decoder necessary for this thesis in Sections 3.5.3 and 3.5.4, respectively. For more in-depth details, we refer readers to [15].

3.5.3 The Encoder

The encoder of MTR takes as input the past trajectories of agents in the scene O and the map of the environment C, where each trajectory and each map element is represented as a polyline. Both sets of polylines are encoded by PointNet-like encoders [47] to obtain agent and map tokens for each agent and each map polyline, respectively.

The agent and map tokens obtained from the polyline encoders are then further processed by a Transformer-based context encoder. The tokens are updated via local self-attention,

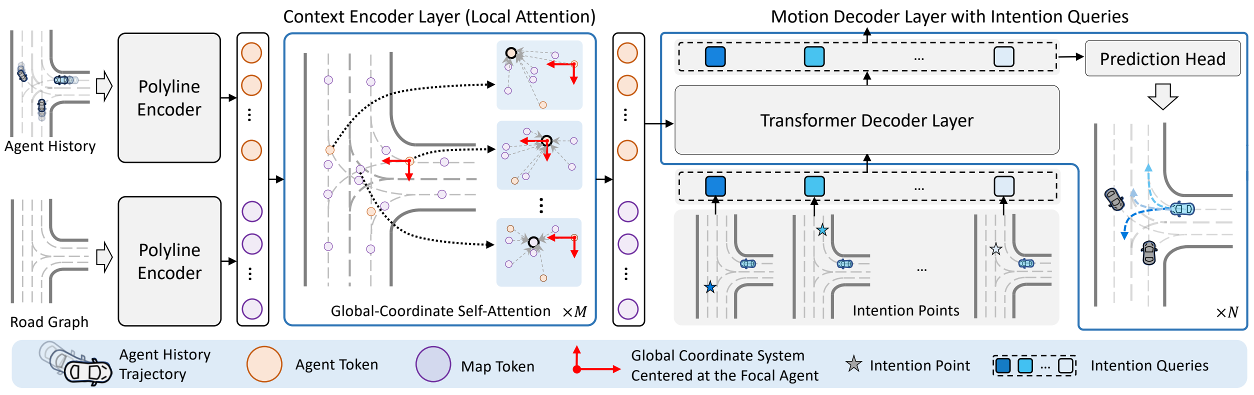

Context Encoder Layer (Local Attention)
Motion Decoder Layer with Intention Queries
Agent History
Polyline Encoder
Road Graph
Polyline Encoder
Global-Coordinate Self-Attention ×M
Transformer Decoder Layer
Intention Points
Prediction Head
×N
Agent History Trajectory
Agent Token
Map Token
Global Coordinate System Centered at the Focal Agent
Intention Point
Intention Queries

Figure 3.3: Overview of the MTR architecture. [16]

where each token attends only to their local neighbouring tokens. The k-nearest neighbour algorithm is used to determine the k polylines that are closest to each query polyline.

In addition to encoding past trajectory and map information, the encoder of MTR also includes future information. The agent tokens from the context encoder are used in an auxiliary prediction task. The goal of this task is to directly predict the future trajectories of the agents using a multi-layer perceptron (MLP). The predictions are re-encoded using the same polyline encoder as before. Another MLP is then used to fuse these future agent features with the original agent tokens, providing us with the final agent tokens that the decoder uses.

Each agent of interest has a different local coordinate frame. In order to handle this, the encoding process is completed N times, once for each agent of interest. Each time, the past observations O and scene context C are encoded in the coordinate frame centered on the agent of interest.

3.5.4 The Decoder

The decoder of MTR takes as input the agent and map tokens from the encoder and uses them for the Transformer keys and values. For each agent of interest, the decoder is conditioned on a set of intention points specific to the agent type (vehicles, cyclists, and pedestrians), representing trajectory end points. For each agent type, the corresponding intention points are determined by running the k-means clustering algorithm on the trajectory endpoints in the training set, with $k = 64$. At each future time step in the prediction horizon T_f, the decoder outputs the parameters of a Gaussian mixture model (GMM) with 64 components, forming 64 trajectories with each trajectory corresponding to an intention point. The predicted distribution $P_{t,i}(x, y)$ of the i^{th} agent's position $\mathbf{x}_{t,i} = (x, y)$ at time step t follows the formulation in [20], [22] and can be written as:

$$P_{t,i}(x, y) = \sum_{k=1}^{64} p_k \cdot \mathcal{N}(x - \mu_{x,k}, \sigma_{x,k}; y - \mu_{y,k}, \sigma_{y,k}; \rho) \tag{3.2}$$

where probability p and parameters (μ_x, μ_y, σ_x σ_y, and ρ) of each component are output by the decoder. We omit the time t and agent i for clarity.

3.5.5 Marginal Trajectory Selection

MTR outputs 64 trajectories corresponding to 64 different high-level maneuvers, where each maneuver corresponds to one of the intention points from Section 3.5.4. In reality, only a small number of trajectories can be processed due to the high computational cost of downstream tasks. As a result, MTR further selects samples to narrow down from the 64 trajectories to six marginal trajectories. For each agent, to generate the final set of six trajectories, MTR uses non-maximum suppression to select the top six distinct predictions from the 64 predicted trajectories (μ_x and μ_y) based on endpoint distances and a distance threshold of 2.5 meters.

3.5.6 Loss Function

The loss function that is minimized during training includes a term for optimizing the GMM parameters and a term for the auxiliary prediction task. At a given future time step for an agent, MTR maximizes the likelihood of the agent's ground truth position (x_{gt}, y_{gt}) by optimizing the predicted GMM parameters with the negative log-likelihood. This is written as

$$\mathcal{L}_{GMM} = -log f_h(x_{gt} - \mu_x, y_{gt} - \mu_y) - log(p_h) \tag{3.3}$$

where f_h is the probability density function of the selected GMM component for optimization and p_h is the corresponding mixture probability. MTR follows [20], [22] and uses a hard assignment strategy to select the GMM component using the distance between intention points and ground truth trajectory endpoints. For the outputs of the auxiliary trajectory regression MLP ($\mathcal{L}_{aux}$), MTR uses the $L1$ regression loss. The total loss used to train the MTR framework is

$$\mathcal{L} = \mathcal{L}_{GMM} + \mathcal{L}_{aux} \tag{3.4}$$

Chapter 4

Learned Factorization and Sampling for Multi-Agent Prediction

4.1 Introduction

An agent's future behaviour can be affected by the presence of others and the need to share space on the road. Therefore, in joint prediction, it is important to model these interactive behaviours and capture possible high-level interaction outcomes in the prediction samples. Additionally, due to the computational complexity of downstream risk assessment, only a small set of predictions can be processed onboard a vehicle. Thus, a joint predictor needs to achieve good coverage of possible high-level outcomes given a limited number of prediction samples. It is challenging, however, to cover the joint prediction space because it grows exponentially in the number of agents and future time steps. In this chapter, we present a prediction method inspired by approximate inference and hybrid estimation algorithms [25]–[27], [48] to address the large joint prediction space and improve sample efficiency, which ultimately allows us to provide diverse joint predictions.

Modelling and predicting human behaviour, can be viewed as a problem with hybrid structure. In the context of joint prediction, high-level interaction outcomes can be represented by discrete variables, while continuous variables correspond to agents' continuous motion [11], [12]. Different discrete interaction outcomes (i.e., outcomes that are qualitatively different) can be identified for specific groups of interacting agents. For instance, for a pedestrian interacting with two cars, distinct discrete outcomes can represent whether the pedestrian crosses before, between, or after the two cars. In this way, one discrete variable can correspond to multiple agents. In a traffic scene, there can be multiple concurrent interactions, each involving a subset of agents in the scene.

Here, we introduce a hybrid prediction method that models the structure of discrete interactions and continuous motion. We present a method that breaks down the traffic scene by identifying the discrete variables and factoring the joint discrete distribution. We represent groups of interacting agents with a discrete variable. We observe that interactions between groups are weak and therefore the joint distribution over the discrete variables can be factored. Inspired by directed acyclic graph (DAG) approximation approaches in belief approximation [31]–[33], we infer a DAG across groups, where edges represent inter-group interaction relationships and target nodes of the directed edges are groups whose interaction outcome is influenced by that of the source node. The DAG allows us to easily sample values corresponding to each factor in an order and condition on previous factors. The discrete outcomes are then used to determine individual agents' future trajectories. Factoring enables better sample efficiency and has been explored in multi-agent monitoring to reduce the state space, reduce sample variance, and maintain accuracy with fewer samples [26]. In this chapter, we explore the notion of neural network-based factorization to reduce the multi-agent prediction space and enable more efficient sampling of interaction outcomes.

The contributions of this chapter are as follows:

- We take inspiration from factorization in probabilistic inference and model-based reasoning and demonstrate how to apply them to neural network-based estimators.
- Our method accomplishes neural factorization and hybrid inference within the task of multi-agent prediction in autonomous driving.
- In our experiments, we probe the effectiveness of our model for improving high-level interaction coverage and joint prediction performance, and compare our method to state-of-the-art methods.

4.2 Preliminaries

4.2.1 Factored Inference

The discrete prediction space grows exponentially in the number of agents and prediction intervals. Several works tackle this problem of scalability by factoring the state distribution. For instance, the method in [25] approximates complex belief state distributions with compact products of factors by clustering strongly coupled state variables and assuming independence across weakly interacting components in a known dynamic Bayesian network. Combining particle filtering and [25], [26] introduces the notion of factored particles and performs particle sampling within each cluster of variables. Compared to particle filtering, [26] reduces sampling variance and therefore more accurately approximates beliefs in larger state spaces using fewer samples. It is also able to handle much larger systems than [25] via sampling rather than exact inference within each factor. In [27], factored particles is extended to hybrid systems. Inspired by this line of work, we leverage factoring to reduce the prediction space and achieve better coverage of outcomes given limited prediction samples. Unlike [25]–[27] which involve problems with known structure, the structure of traffic scenarios and dependencies among variables are not known beforehand, and change across different data samples. We therefore train a neural network to map inputs (past observations and context) of traffic scenarios to their corresponding factored graphical structure. The methods in [25]–[27] also do not represent relationships between factors. We draw inspiration from prior work in DAG approximation in belief propagation [31]–[33] and learn to determine a DAG structure between groups, where groups correspond to discrete factors.

4.2.2 Interactive Prediction

In our work, we learn to group interacting agents. A body of work explored how to cluster agents and capture group behaviours [49]–[52]. These methods grouped agents that are interacting, extracted group features, and leveraged the group features in conjunction with individual agent features to predict trajectories. Of these works, [49] and [50] group agents purely using rule-based clustering based on agent positions and proximity. Beyond rule-based grouping, [51] and [52] neurally inferred hypergraph structure and discrete interaction types of each group. Discrete variables were assumed conditionally independent given past observations. Inspired by these methods, we neurally determine which agents belong in a group while additionally leveraging rule-based filters to identify interacting agents and weakly supervise our model's group assignments to provide semantic meaning. Unlike existing methods, we additionally infer inter-group interaction relationships. We take advantage of the inter-group structure to evaluate discrete group outcomes in an order that enables us to condition on previous groups' outcomes, instead of assuming independence across group behaviours.

A number of joint trajectory prediction methods have learned relational structures among agents to model pairwise interactions [5], [7]–[9], [53]. Among these methods, [5], [7], [53] address the problem as conditional behaviour prediction by determining a directed acyclic structure over individual agents and predict in order according to the graph. The focus in [5] and [53] is on pairwise interacting scenarios, and [7] scales the idea to networks of pairwise interactions. Other methods [8], [9] learn to infer interaction graphs and corresponding discrete pairwise interaction types for each edge. We take inspiration from these methods and learn to determine a directed acyclic interaction graph, but over groups of agents rather than individual agents. This way, we can capture both tightly coupled multi-agent behaviours within groups and weaker interactions across groups.

4.3 Approach Overview

The prediction problem formulation follows Chapter 2, where our goal is to output a weighted set of K joint trajectory samples $S = \{(\mathbf{x}^{(k)}, w^{(k)})\}_{k=1}^{K}$. In this chapter, we address this as a hybrid discrete-continuous prediction problem, where the continuous variables $\{\mathbf{x}_{1:T_f,i}\}_{i=1}^{N}$ represent the future continuous trajectories of the N agents in the scene, as defined in Chapter 2. The discrete variables $\{\mathbf{z}_m\}_{m=1}^{M}$ represent high-level future interaction outcomes among groups of agents, which we assume are fixed over the prediction horizon. Although we focus on interactions, our framework also enables these variables to capture individual maneuvers as well. Formally, our method estimates the joint distribution over all variables, $p(\{\mathbf{x}_{1:T_f,i}\}_{i=1}^{N}, \mathbf{z}_1, ..., \mathbf{z}_M | O, C)$, where O and C are past observations and scene context, as defined in Chapter 2. By the chain rule of conditional probability, this can be factored as

$$p(\{\mathbf{x}_{1:T_f,i}\}_{i=1}^{N}, \mathbf{z}_1, ..., \mathbf{z}_M | O, C) = p(\{\mathbf{x}_{1:T_f,i}\}_{i=1}^{N} | \mathbf{z}_1, ..., \mathbf{z}_M, O, C) p(\mathbf{z}_1, ..., \mathbf{z}_M | O, C) \tag{4.1}$$

Our inference task in this chapter is to infer the distribution over continuous trajectories, specifically the first term of the right side of Equation 4.1. In our method, we make two assumptions that allow us to further factor Equation 4.1. First, we assume that a qualitative, high-level abstraction of joint future behaviour is sufficient to capture dependencies across agents' futures. That is, given an assignment to the discrete variables, we can assume that agents' future trajectories are conditionally independent. Therefore, we can factor the first term on the right side of Equation 4.1 as follows

$$p(\{\mathbf{x}_{1:T_f,i}\}_{i=1}^{N} | \{\mathbf{z}_i = z_i\}_{i=1}^{M}, O, C) = \prod_{i=1}^{N} p(\mathbf{x}_{1:T_f,i} | \{\mathbf{z}_i = z_i\}_{i=1}^{M}, O, C) \tag{4.2}$$

where z_i is the assignment to the i^{th} discrete variable. We additionally observe that across groups of agents, interactions are sparse. Therefore, dependencies among discrete variables $\{\mathbf{z}_m\}_{m=1}^{M}$ are sparse. This motivates our second assumption – that the dependencies among discrete variables form a directed acyclic graph (DAG), where the source node is the factor that influences the interaction outcome of the target node. This allows us to factor $p(\mathbf{z}_1, ..., \mathbf{z}_M | O, C)$ to obtain an order in which to conveniently sample the discrete variables, one at a time, and condition on previously sampled variables. Specifically, we have

$$p(\mathbf{z}_1, ..., \mathbf{z}_M | O, C) = \prod_{m=1}^{M} p(\mathbf{z}_m | parents(\mathbf{z}_m), O, C) \tag{4.3}$$

To summarize, we assume the following factorization of the joint distribution over hybrid variables.

$$\begin{aligned} &p(\{\mathbf{x}_{1:T_f,i}\}_{i=1}^{N}, \mathbf{z}_1, ..., \mathbf{z}_M | O, C) \\ &= \prod_{i=1}^{N} p(\mathbf{x}_{1:T_f,i} | \mathbf{z}_1, ..., \mathbf{z}_M, O, C) \prod_{m=1}^{M} p(\mathbf{z}_m | parents(\mathbf{z}_m), O, C) \end{aligned} \tag{4.4}$$

where $parents(\cdot)$ denotes the parents of a variable in the DAG.

A detailed diagram of our method is shown in Figure 4.1. In the remainder of this chapter, we discuss details of our approach and present experimental results. To uncover the interaction structure in traffic scenes and obtain the factorization, our method

1. learns to assign agents to each factor (Section 4.4.1)
2. learns to determine edges between factors (Section 4.4.3).

Using the obtained structure, we then make joint predictions by

1. sampling diverse discrete interaction outcomes (Section 4.5.1)
2. decoding trajectories conditioned on the interaction outcomes (Section 4.5.2).

4.4 Approach: Learning to Factor for Multi-Agent Prediction

This section describes how we obtain the factorization described by Equation 4.3.

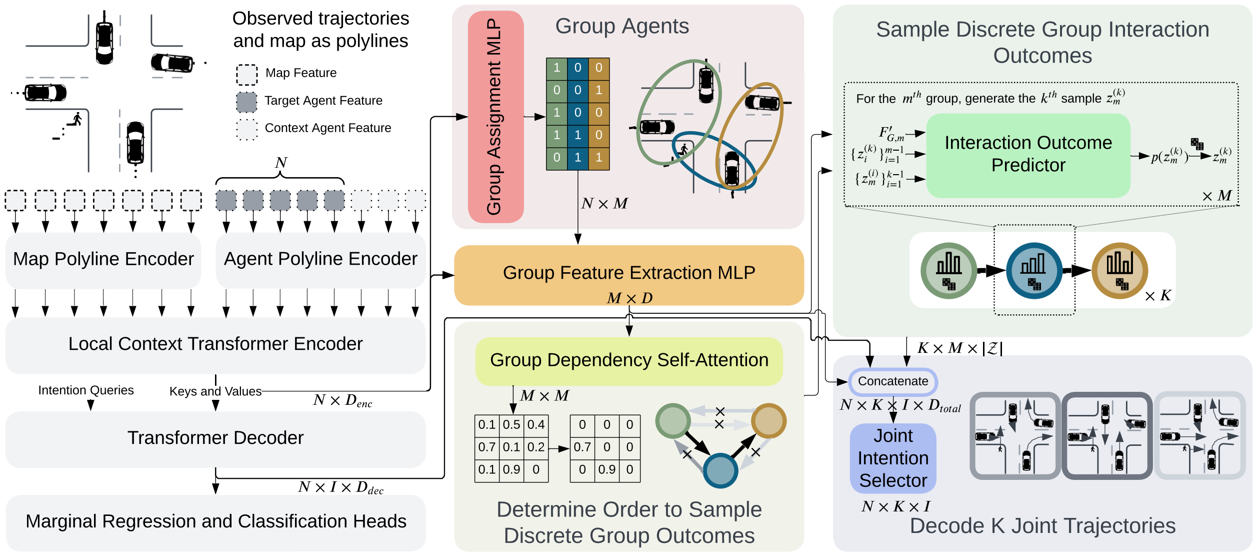

Observed trajectories and map as polylines
Map Feature
Target Agent Feature
Context Agent Feature
N
Map Polyline Encoder
Agent Polyline Encoder
Local Context Transformer Encoder
Intention Queries
Keys and Values
$N \times D_{enc}$
Transformer Decoder
$N \times I \times D_{dec}$
Marginal Regression and Classification Heads
Group Assignment MLP
Group Agents
$N \times M$
Group Feature Extraction MLP
$M \times D$
Group Dependency Self-Attention
$M \times M$
Determine Order to Sample Discrete Group Outcomes
Sample Discrete Group Interaction Outcomes
For the m^{th} group, generate the k^{th} sample $z_m^{(k)}$
Interaction Outcome Predictor
$\times M$
$\times K$
$K \times M \times |Z|$
Concatenate
$N \times K \times I \times D_{total}$
Joint Intention Selector
$N \times K \times I$
Decode K Joint Trajectories

Figure 4.1: Overview of the proposed method. We leverage MTR [15] as a backbone marginal trajectory predictor to obtain a weighted set of I marginal trajectories per agent. Our method takes the agent embeddings from the encoder to (1) group agents according to interactions, (2) extract group-level features (F_G), (3) factor the joint distribution over discrete variables representing the groups, and (4) sample each discrete variable in order according to the directed acyclic factorization. We do this K times to obtain K joint sample sets. (5) We take the agents' features from the decoder, their corresponding group features, and the discrete variables and pass them to the joint intention selector. For every agent, the selector outputs one group-aware distribution over the I possible selections for each $k = 1, ..., K$.

4.4.1 Grouping Interacting Agents

We begin with a driving scenario containing N agents that are interacting in M groups. The groups are not necessarily partitions – agents can be part of more than one group. In order to determine which agents belong in a group together, we aim to approximate with a neural network the probability that an agent i belongs to group m. Formally, the goal is to estimate the probability distributions for $i = 1, ..., N$ and $m = 1, ..., M$

$$p(G_{i,m}|O, C) \in [0, 1] \tag{4.5}$$

where $G_{i,m} \in \{0, 1\}$ represents the event that the i^{th} agent belongs to the m^{th} group. The observations O and context C are as defined in Chapter 2. We accomplish this using a multi-layer perceptron (MLP) that takes as input the features from the backbone encoder and outputs a vector of size M per agent representing group assignment probabilities for M groups. Our MLP consists of three layers, with sizes of (256, 256, M). Note that the maximum number of groups M must be selected, and our framework can use fewer than M groups depending on the scenario.

We sample from the resulting distribution to obtain a final N by M binary incidence matrix G representing the assignment of agents to groups. During training, we use the Gumbel-softmax [54] trick to sample in a differentiable manner. Note that groups are not partitions, so agents can belong to more than one group. In addition, it is possible for agents to not belong to any group.

We softly constrain the group assignments using our loss terms to encourage certain desirable qualities in the groups. First, we note that interactions between agents can be identified by filtering for specific behaviours in the data. These filters can consider information including shared space based on the paths of multiple agents, changes in speed in the presence of other agents, among others. We leverage this insight and filter for interactions that involve yielding, following, or general intersecting between agents' paths. Details on the filtering criteria for each interaction type are presented in Appendix A. We use these filters during training to encourage semantic meaning in the groups outputted by our network.

We introduce a loss term that encourages agents to be in the same group if they are filtered to be interacting. Specifically, during training, we use the ground truth future trajectories to construct a ground truth interaction graph of all agents. We encourage neighbouring pairs of agents in this graph (i.e., agents that directly interact) to be placed in the same group using the following

$$\mathcal{L}_{sem.\,dense} = 2 - \max_{m}(G_{i,m} + G_{j,m}) \tag{4.6}$$

Intuitively, this term encourages a more dense incidence matrix G based on ground truth interactions.

We note that for smaller group sizes, there is a lower number of possible interaction outcomes. As a result, smaller groups require fewer samples to achieve good coverage of interaction outcomes. Therefore, to encourage better sample efficiency, we prefer to have more small groups over fewer large groups when possible. With this in mind, in addition to encouraging semantically meaningful groups by densifying the groups, the inverse can be done to discourage overly dense groups. Groups consisting of agents that are not neighbours in the ground truth interaction graph can be penalized with the following cost

$$\mathcal{L}_{sem.\,sparse} = \sum_{m=1}^{M} \sum_{(i,j)\in group\, m} (d_{i,j} - 1)(G_{i,m} + G_{j,m} - 1) \tag{4.7}$$

where $d_{i,j}$ is the shortest distance between agents i and j in the interaction graph. If the agents are not connected, the distance is set to N. Intuitively, if agents are not adjacent (i.e., $d_{i,j} - 1 > 0$), this cost term would penalize groups that contain both agents i and j.

We additionally consider two edge cases using margin costs. First, to avoid grouping all agents into one group – and thereby defeating the purpose of factoring, we softly constrain group sizes such that groups with more agents are penalized according to

$$\mathcal{L}_{small\, groups} = \sum_{m=1}^{M} \max(N_m - margin_{sparse}, 0) \tag{4.8}$$

where N_m represents the number of agents assigned to group m and the sparsity margin we set is $margin_{sparse} = 3$. With this margin, $\mathcal{L}_{sparse\, groups}$ contributes to the overall objective when $N_m \geq 4$. In terms of the binary incidence matrix G, this loss term is used when the sum of columns exceed $margin_{sparse}$.

The second edge case is when agents are not assigned to any group. When this occurs, the agent does not take part in subsequent feature extraction (Section 4.4.2) and factorization and discrete outcome sampling (Section 4.4.3) steps. This effectively results in predictions equivalent to [15]. This is reasonable for agents that do not interact with others. However, to leverage our method for potential advantages in coverage, we encourage agents to be assigned to at least one group. The corresponding cost term is computed as

$$\mathcal{L}_{min\, groups} = \sum_{i=1}^{N} - \min(M_i - margin_{min\, groups}, 0) \tag{4.9}$$

where M_i is the number of groups to which agent i is assigned and $margin_{min\, groups} = 1$ to encourage at least one group per agent. In terms of the binary incidence matrix G, this loss term is used when the sum of rows is below $margin_{min\, groups}$. For the agents that do not interact with others, they can still be represented by being the only agent assigned to a group.

Overall, the loss function for grouping interacting agents and associated coefficients, selected empirically, are as follows

$$\mathcal{L}_{groups} = \mathcal{L}_{sem.\,dense} + 0.5\mathcal{L}_{sem.\,sparse} + 0.5\mathcal{L}_{small\, groups} + \mathcal{L}_{min\, groups} \tag{4.10}$$

4.4.2 Extracting Group Features

According to the group assignments, we extract a feature vector representing each group, which fuses the encoder features of the agents that belong to the group. These group-level features are used for subsequent group-level operations, including determining the conditioning structure in the factorization and sampling from the factors to determine high-level discrete outcomes, as per Section 4.4.3.

We process the agent features from the encoder through a three-layer MLP, with layer sizes of (256, 256, 256), and obtain a feature vector per agent. To form feature vectors per group, we take the element-wise weighted sum of feature vectors of the agents that belong to each group. Feature vectors are weighted according to the group assignment probabilities denoted by Equation 4.5.

4.4.3 Determining Inter-Group Dependencies and Sampling Order

The next step is to determine inter-group dependencies. We represent this with a $M \times M$ adjacency matrix representing relationships between groups. The latent variables $\mathbf{z} = \{\mathbf{z}_m\}_{m=1}^{M}$ represent the high-level interaction outcomes of the groups. To infer the sparse relations between groups, we assume that dependencies between groups have a directed acyclic structure. As a result, the joint distribution of latent variables representing the groups can be factored as

$$p(\mathbf{z}_1, ..., \mathbf{z}_M | O, C) = \prod_{m=1}^{M} p(\mathbf{z}_m | parents(\mathbf{z}_m), O, C) \tag{4.11}$$

Based on the factorization, we obtain an order in which to evaluate the latent variables $\{\mathbf{z}_m\}_{m=1}^{M}$ that allows us to condition on the other latent variables as appropriate, according to the factorization.

We use self-attention to determine the adjacency matrix. In our implementation, we use three three-layer MLPs, all with layer sizes of (256, 256, 256), to map the group features to keys, values, and queries. We use two attention heads and take the average of the two for the final attention matrix. We refer readers to [46] for details on self-attention. To train the network, we draw inspiration from prior work in DAG structure learning [55]–[57]. We adopt the equation in [57] as a term in our loss function to softly constrain the attention matrix to have a DAG structure, as follows:

$$\mathcal{L}_{DAG} = tr[(I + \eta A \circ A)^M] - M \tag{4.12}$$

where η is a constant that we set to 1.

Finally, we follow the post-processing procedure presented in [56] to obtain the DAG adjacency matrix. This process consists of pruning the weakest connections until the resulting matrix forms a DAG. This effectively assumes that the weakest inter-group interactions are negligible. Based on the DAG, we order the variables $\{\mathbf{z}_m\}_{m=1}^{M}$ by topologically sorting with Kahn's algorithm.

4.5 Approach: Sampling-Based Factored Joint Prediction

This section describes how we obtain joint hybrid predictions described by Equation 4.4, given the factorization in Equation 4.3.

4.5.1 Sampling Diverse Interaction Outcomes

We aim to obtain K joint discrete samples that cover diverse interaction outcomes. To do this, we first sample the discrete latent variables according to the DAG order, conditioning subsequent samples on the previous ones, until we have sampled values for all discrete variables. This is follows Equation 4.3. We repeat the ordered sampling process K times to obtain K joint samples of the discrete variables. For each set of $\{\mathbf{z}_m^{(k)}\}_{m=1}^M$ for $k = 1, ..., K$, we additionally condition on previously sampled sets of discrete variables, as done in [11].

Specifically, for the k^{th} joint sample of the m^{th} group, we feed the updated group features from self-attention, encoded features of the parent latent variables, and encoded features of the previous $k-1$ joint samples to a three-layer MLP with layer sizes (768, 768, $|\mathcal{Z}|$). We represent samples with one-hot vectors with a dimension of $|\mathcal{Z}|$. The sampled vectors of parent latent variables along with the corresponding groups' features are encoded using a three-layer MLP with layer sizes (256, 256, 256) followed by the element-wise weighted sum to aggregate features from multiple parent groups. The weights come from the adjacency matrix. The previous $k-1$ joint samples are encoded with a two-layer MLP with sizes (256, 256) and aggregated with max pooling. For the first group and first joint sample, feature vectors are initialized with zeros. During training, we use the Gumbel-softmax trick to sample in a differentiable manner. After obtaining K discrete samples for each group, we associate them back to the relevant agents. For agents that belong to more than one group, we sum the discrete vectors corresponding to its groups.

The aim of our method is to achieve good coverage of discrete interaction outcomes. We use two loss terms together to encourage this. The first term encourages high-level semantic interaction meaning to be captured by the discrete values. For two sets of discrete samples, if the interaction outcome within a group is different, according to the decoded continuous trajectories (Section 4.5.2), the corresponding discrete variable should be different. For K joint trajectory samples, can be summarized as

$$\begin{aligned}&\mathcal{L}_{distinct}\\ &= \sum_{m=1}^{M}\sum_{u=1}^{K-1}\sum_{v=u+1}^{K} \texttt{isDistinct}\left(\{\mathbf{x}_{1:T_f,i}^{(u)}\}_{i=1}^{N_m}, \{\mathbf{x}_{1:T_f,i}^{(v)}\}_{i=1}^{N_m}\right) * \|\mathbf{z}_m^{(u)} - \mathbf{z}_m^{(v)}\|_2 \end{aligned} \tag{4.13}$$

To determine whether two interaction outcomes are different, we consider the predicted continuous trajectories of the agents in the group and construct an interaction graph based on yielding, following, and intersecting behaviours. Two agents in the group are connected by a directed edge according to the filtered interaction type. This forms the interaction graph. If the interaction graphs of different samples are different, we consider the two joint predictions to be semantically distinct. This corresponds to `isDistinct` in Equation 4.13.

The second term we use for discrete interaction coverage is the entropy of the probability distribution for each discrete variable. We aim to maximize the entropy to encourage sampling diverse discrete values. That is

$$\mathcal{L}_{coverage} = \sum_{m=1}^{M} \sum_{d=1}^{|\mathcal{Z}|} p(z_{m,d}) \log p(z_{m,d}) \tag{4.14}$$

where $|\mathcal{Z}|$ is the domain size of the discrete variables.

We empirically set the coefficient of the distinctness term to be twice that of the coverage term. To summarize, we use the following loss to encourage diverse interaction outcomes

$$\mathcal{L}_{interaction} = 2\mathcal{L}_{distinct} + \mathcal{L}_{coverage} \tag{4.15}$$

4.5.2 Predicting Joint Trajectory Distributions

Finally, to obtain the joint continuous trajectories, we condition on the discrete variables to produce a distribution over future trajectory candidates and corresponding intention points. With the intention point-conditioned agent features from the Transformer decoder ($N \times I \times D_{dec}$), group features and sampled discrete outcomes that are associated back to the relevant agents ($N \times D_{enc}$ and $K \times N \times |\mathcal{Z}|$, respectively), and a one-hot sample index indicating k ($K \times D_{index}$), we repeat tensor dimensions as required and concatenate them to form a $N \times K \times I \times D_{total}$ feature tensor, where $D_{total} = D_{dec} + D_{enc} + |\mathcal{Z}| + D_{index}$. We feed these features to our joint intention selector.

Our selector consists of two parts. The first maps the feature dimension per agent *per intention point* to 1. This is a two-layer MLP with layer sizes (256, 1). After reshaping the resulting tensor, we obtain a vector of dimension I per agent. Then, we pass all features through a two-layer MLP with layer sizes (I, I). After applying softmax, we obtain the categorical distribution over the I intention points and corresponding trajectory candidates from the backbone method.

For supervision, we use a min-of-K joint cross-entropy loss, which first sums the cross-entropy over all N agents and takes the cross-entropy corresponding to the most accurate joint trajectory set out of the K samples. We use the joint min-of-K to encourage joint diversity among the sampled distributions [58]. We use the intention point closest to the ground truth trajectory endpoint as the positive intention point for computing the cross-entropy. Note that we do not use NMS for sample selection per agent. Rather, to obtain K joint samples, we sample $\mathbf{z}$ K times. To summarize,

$$\mathcal{L}_{JIS} = \min_{k=1,\ldots,K} \sum_{i=1}^{N} CrossEntropy(I_i^{(gt)}, JIS_i(O, C, G, \mathbf{z}^{(k)})) \tag{4.16}$$

The intention points in MTR are obtained by running k-means on single-agent trajectory endpoints from the training data, without any other information about the trajectories. In our case, however, the MTR intention points may not be optimally located when the goal is to achieve good interaction coverage. Therefore, when training the joint intention selector, we turn the intention points into trainable parameters and also estimate gradients for them, updating them at each training iteration. We add an additional term to guide the intention point training. This term minimizes the distance from the selected intention point to the ground truth trajectory endpoint for the closest corresponding minimum intention point. With $I_i^{(k)}$ as the selected intention point corresponding to the k^{th} joint sample for the i^{th} agent, this loss is

$$\mathcal{L}_{kmeans} = \sum_{i=1}^{N} \min_{k=1,\ldots,K} \|\mathbf{x}_{T_f,i} - I_i^{(k)}\|_2 \tag{4.17}$$

4.6 Experiments

In this section, we evaluate the effectiveness of our method at covering interaction modes and accurately predicting joint trajectories. We also evaluate the impact of each element of our factorization on the number of interaction modes captured.

4.6.1 Dataset Details

We train and validate our model on the Waymo Open Motion Dataset [59]. The dataset consists of $487k$ driving scenes in the training split, 44097 scenes in the regular validation split, and 43479 scenes in the interactive validation split. Each target agent has an observed history of up to one second. Ground truth future trajectories are provided for up to eight seconds in the future. All scenes are provided at $10Hz$.

Training and regular validation scenes consist of up to eight target agents, while interactive validation scenes consist of two closely interacting agents as target agents. Previous work such as [15], [16] and the Waymo benchmark focus on the two-agent interactive validation split to evaluate joint prediction performance and do not compare joint metrics on the regular validation set with more target agents. While we provide joint prediction results for the interactive validation split for reference, our analysis primarily focuses on joint prediction of complex subsets of the regular validation split, as this provides insights into interactive scenarios beyond the commonly examined two-agent scenarios.

To identify complex subsets of the regular validation split, we filter for yield interactions. Further details on our yield criteria are provided in Appendix A. We evaluate our method on highly complex *networked* yield scenarios containing at least $2 \leq N_{yield} \leq 5$ agents connected in a network of yield interactions. Two yield interactions are considered to be *networked* if one of the agents in the first yield is also involved in the second yield.

4.6.2 Training Details

Before training our method, we initialize the weights that correspond to the MTR backbone (grey boxes in Figure 4.1) to fully trained MTR weights. These weights are obtained by training MTR according to the hyperparameters in [15] and taking the best checkpoint with results as close as possible to [15]. The other weights in our model are randomly initialized. Then, with the weights of the MTR encoder frozen and all other weights trainable, we train our method end-to-end in two phases.

When training our method, our full loss function takes the following form

$$\mathcal{L} = \alpha \mathcal{L}_{MTR} + \beta \mathcal{L}_{groups} + \gamma \mathcal{L}_{DAG} + \delta \mathcal{L}_{interaction} + \epsilon \mathcal{L}_{JIS} + \zeta \mathcal{L}_{kmeans} \tag{4.18}$$

In the first phase, we train with $\alpha = 0.05$, $\beta = 1$, $\gamma = 1$, $\delta = 5$, $\epsilon = 0.0001$, and $\zeta = 0$. We are not yet training the intention points in the first phase. We train with these coefficients for two epochs. In the second phase, using a trainable copy of MTR's intention points, we jointly optimize them with the other model parameters. We modify the coefficients for this stage of training and have $\delta = 10$, $\epsilon = 0.00001$, and $\zeta = 1$. α, β, and γ remain the same as the first phase. We train our model for four epochs in the second phase. In both phases, we use a learning rate of 0.0001 and train on two GPUs with a batch size of 20.

We sample $K = 6$ times during training and evaluation, following the Waymo [59] prediction benchmark. We set the maximum number of groups per scene to be $M = 8$ as there are up to eight target agents per scene. We set the latent variable $\mathbf{z}$ domain size to be $|\mathcal{Z}| = 10$, sample index dimension to $D_{index} = 10$, and follow [15] for $D_{enc} = 256$, $D_{dec} = 512$, and $I = 64$.

4.6.3 Interaction Mode Coverage and Joint Prediction Error

We evaluate our method's ability to cover diverse interaction modes and accurately predict joint trajectories. For interaction mode coverage, we compute the number of different interaction modes captured by the K joint prediction samples in a scene, and average across all scenes in the data split. We denote this with N_{modes}. To determine whether a joint prediction sample is a new mode, we construct an interaction graph among agents in the scene based on the predicted joint trajectories. If two agents are in a yield or follow interaction, or have paths that intersect, we connect them in the interaction graph. Two joint prediction samples are considered to reflect different interaction modes if their corresponding interaction graphs are different. Details on how we filter for yielding, following, or intersecting trajectories are provided in Appendix A. For joint prediction, we compare performance on standard metrics – mean average precision (mAP), minimum average displacement error (minADE), minimum final displacement error (minFDE), and miss rate, as defined in [59].

We evaluate our method on several data splits. We filter for subsets of the Waymo validation data containing networked yield interactions involving at least $2 \leq N_{yield} \leq 5$ agents. Scenes from these data subsets contain *at least* N_{yield} interacting agents and up to eight target agents. Not all target agents are connected in the yield network, but at least N_{yield} (and up to 8) of them are. Additionally, we include results on the full Waymo interactive validation split for reference, as this is the widely accepted data split in which to evaluate joint prediction. We compare with MTR [15], our backbone marginal trajectory predictor, which obtains joint predictions assuming independence and factoring the joint distribution into the product of marginals. MTR takes the top K out of K^N possible combinations of marginal trajectories that have the highest product of marginal prediction scores. The results are presented in Table 4.1.

From the results on the first four data splits in Table 4.1, we can see that our method performs best in interaction mode coverage (N_{modes}) across all networked yield data splits. The coverage improvement that we obtain over MTR is 0.466 (23.7%), 0.770 (35.4%), 1.079

Data Split	Method	mAP (↑)	minADE (↓)	minFDE (↓)	MissRate (↓)	N_{modes} (↑)
$N_{yield} = 2$	MTR	0.0514	1.1943	2.7445	0.7979	1.965
	Ours	0.0251	1.1939	2.7660	0.8181	**2.431**
$N_{yield} = 3$	MTR	0.0561	1.2580	2.9460	0.8768	2.178
	Ours	0.0243	1.2419	2.8993	0.8810	**2.948**
$N_{yield} = 4$	MTR	0.1023	1.3246	3.0153	0.9291	2.371
	Ours	0.0600	1.2969	3.0276	0.9267	**3.450**
$N_{yield} = 5$	MTR	0.0010	1.3753	3.1082	0.9619	2.625
	Ours	0.0009	1.3782	2.7828	0.9517	**3.888**
Interactive Validation	MTR (reproduced)	0.2066	0.9375	2.1140	0.4349	**1.725**
	Ours	0.1538	1.0619	2.4595	0.5209	1.485
Validation (full, $N \geq 1$)	MTR	0.1982	1.1248	2.5320	0.6960	1.425
	Ours	0.1264	1.1450	2.6176	0.7266	**1.663**
Validation (subset, $N \geq 3$)	MTR	0.0670	1.1810	2.7682	0.7910	1.596
	Ours	0.0513	1.1701	2.7309	0.8065	**1.926**

Table 4.1: Interaction mode coverage and joint prediction metrics on various validation data splits. (↓ indicates lower is better and ↑ indicates higher is better.)

(45.5%), and 1.263 (48.1%) for $2 \leq N_{yield} \leq 5$, respectively. This is plotted as mode coverage ratios of our method compared to MTR in Figure 4.2. Notably, as interaction complexity increases (i.e., as N_{yield} increases), we obtain an increasing margin of improvement in N_{modes} over MTR, while maintaining comparable minADE, minFDE, and miss rate. Figure 4.3 illustrates interaction coverage along with corresponding minADE for MTR and our method. As yield complexity increases, coverage improvement grows and minADE remains comparable with MTR. This shows that our method is better at scaling and predicting diverse interaction outcomes in complex, interactive multi-agent traffic scenes without compromising prediction error.

Overall, our performance on prediction errors (minADE, minFDE, and miss rate) are comparable with MTR. However, we observe that our method consistently performs worse on mAP. Among these metrics, mAP is the only one that considers the prediction weights, while the others measure errors based only on the predictions' distances to ground truth. mAP uses the weights as a relative ranking of prediction samples and measures the area under the precision-recall curve averaged over various semantic buckets, where each bucket represents a high-level ground truth maneuver [59]. Our lower mAP, despite comparable errors, suggests that the gap is ultimately caused by an inferior ranking of joint predictions. We leave the improvement of the prediction weights and mAP as future work and exclude it in the remainder of our analyses.

We note that MTR achieves a N_{modes} that is 0.240 (16.1%) better than our method on the two-agent interactive validation data split. MTR's ability to capture multiple interaction modes can be attributed to its use of NMS in selecting marginal predictions. NMS effectively brute-forces diversity for single-agent prediction. Then, when selecting the top K out of K^2 options, although MTR does not guarantee *joint semantic* diversity, the selection process ensures every joint prediction is composed of a different set of marginal trajectories. On

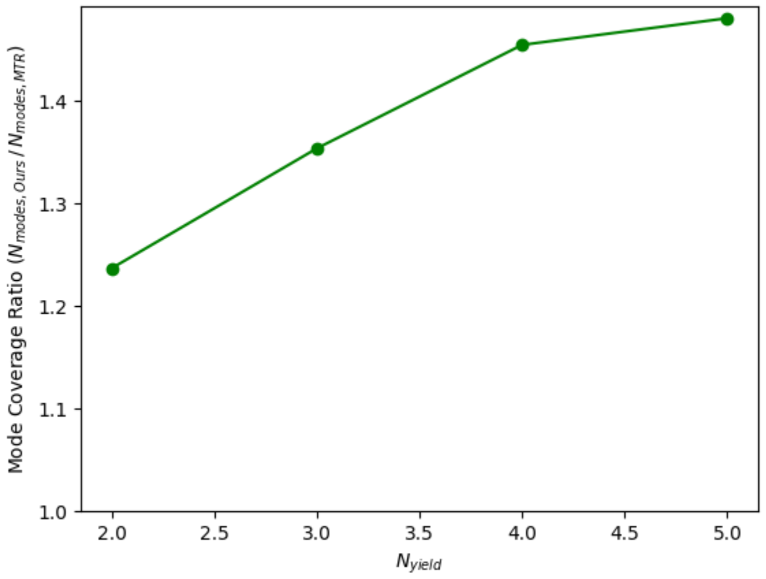


Figure 4.2: Mode coverage ratio of our method to MTR at varying interaction complexities.

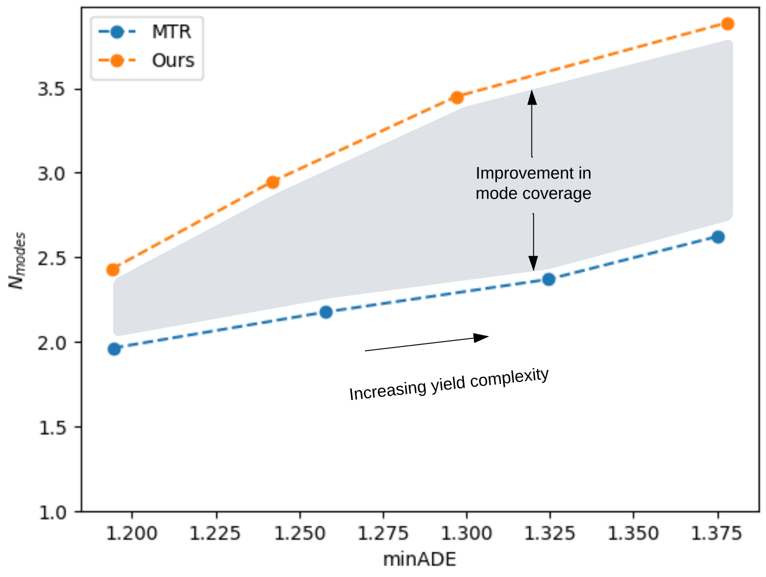


Figure 4.3: A comparison of MTR and our method in terms of N_{modes} covered versus minADE for $2 \leq N_{yield} \leq 5$.

the other hand, our method does not use NMS and relies solely on latent variable $\mathbf{z}$ for diversity. It is therefore possible for different $\mathbf{z}$'s to result in the same set of trajectory selections. When evaluating traffic scenes with strictly two agents, NMS allows MTR to still achieve good semantic diversity without reasoning about joint behaviour. However, this approach fails when more agents are involved, such as in complex networked yield scenarios and those with more than two agents. The same line of reasoning also explains why our method performs worse on prediction metrics on the interactive split, but is comparable to MTR on other data splits.

In the last two data splits of Table 4.1, we additionally show performance on the full standard validation dataset, which contains one to eight target agents, along with a subset that excludes scenarios with only one or two agents. From these two data splits, we show that even without further filtering for the presence of complex interactions, our method achieves better mode coverage than MTR once we include scenes with more than two agents. Additionally, when focusing on the scenarios with more agents ($N \geq 3$), our method achieves improved mode coverage and prediction errors relative to MTR. This emphasizes our method's ability to cover diverse interaction outcomes in multi-agent scenarios.

4.6.4 Sample Efficiency for Interaction Mode Coverage

In this section, we compare the sample efficiency of our method compared with MTR by looking at the number of modes covered N_{modes} when varying the number of joint prediction samples K. Sample efficiency is important because a limited number of prediction samples can be processed in real-time onboard an autonomous vehicle due to the high computational complexity of downstream tasks. Therefore, it is desirable for our method to cover more interaction modes even when having fewer prediction samples.

In Figure 4.4, we show the performance of our method compares with MTR when we vary the number of joint prediction samples for $1 \leq K \leq 6$. Since two-agent scenarios have been well studied by past methods, we focus here on the validation subset with $N_{yield} = 3$ to highlight performance on scenarios involving at least three agents. We find that the performance of our method is better than MTR at every value of K. (At $K = 1$, all methods are guaranteed to have $N_{modes} = 1$.) This highlights the fact that our method achieves superior semantic diversity, even when fewer joint prediction samples are afforded. In fact, with $K = 4$ samples from our method, we already exceed the N_{modes} achieved by $K = 6$ samples from MTR. This reflect an improvement in sample efficiency of over 33% in terms of interaction mode coverage.

4.6.5 Ablation Studies

Effect of Factorization on Mode Coverage

In this section, we present an ablation study to show how each step in our factorization method contributes to interaction mode coverage. As with Section 4.6.4, we present results for the validation subset with $N_{yield} = 3$ to focus on scenarios with three or more interacting agents. We investigate the impact of the three main elements of our method – the $\mathbf{z}$ variable for obtaining different interaction modes, the grouping of agents for factoring and effectively

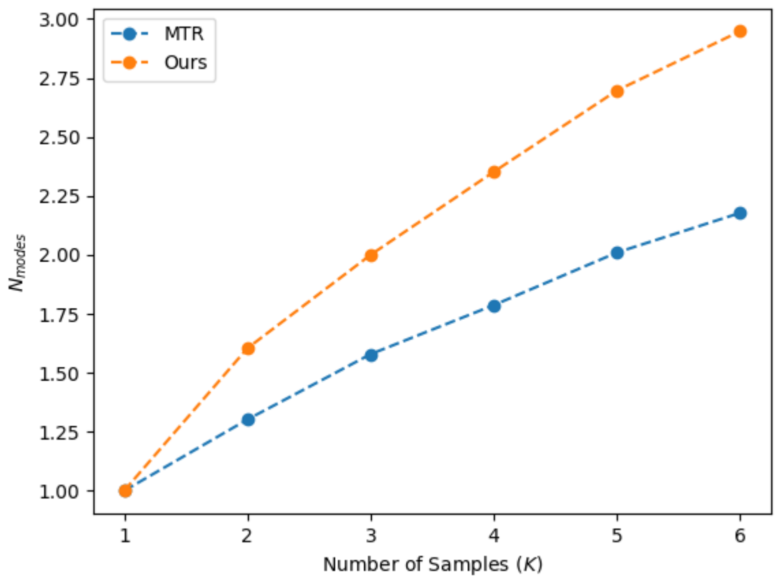


Figure 4.4: The number of interaction modes N_{modes} covered by MTR versus our method at various numbers of joint prediction samples K for the $N_{yield} = 3$ validation data split.

reducing the prediction space, and the ordering of the groups for capturing interactions across groups by conditioning and sampling $\mathbf{z}$ values.

The results of this ablation study are presented in Table 4.2. We observe that our full method performs the best across most metrics. Additionally, when incrementally including components of our method, we observe performance improvements, particularly in N_{modes}, as expected. The improvement from sampling $\mathbf{z}$ shows that varying the latent variable improves interaction mode coverage, as expected. The results in N_{modes} from grouping agents show that reducing the sample space by decomposing the joint prediction distribution is effective at improving interaction coverage. Ordering variables for conditioning and sampling $\mathbf{z}$ provides further improvement, reflecting its effectiveness in modelling influences that groups have on each other.

Sample $\mathbf{z}$	Group	Order	minADE (↓)	minFDE (↓)	MissRate (↓)	N_{modes} (↑)
✗	✗	✗	1.2471	2.9111	0.8856	2.810
✓	✗	✗	1.2444	2.8985	0.8827	2.841
✓	✓	✗	1.2430	2.8955	0.8817	2.913
✓	✓	✓	1.2419	2.8993	0.8810	**2.948**

Table 4.2: Interaction mode coverage and joint prediction metrics for various versions of our factoring approach on the $N_{yield} = 3$ validation split.

Learned Intention Points

We also examined the impact of learning the intention points after initializing them to MTR's points from k-means. We observed that the learned points were shifted towards regions with a higher density of eight-second trajectory endpoints, and the intention points farther from the agent's position at $t = 0$, $(0, 0)$, shifted more, as shown in Figure 4.5. The k-means algorithm spreads out k points to cover the ground truth trajectory endpoints as well as possible, even those that are far away from others. This is not necessarily the best option, especially if it reduces the ability to choose the right intentions. By making these points trainable, we can get points that better help us optimize coverage in areas where there are more ground truth endpoints.

We verified this by comparing metrics on the yield validation split with three or more interacting agents. By learning the points, we saw an improvement in most metrics including mode coverage, shown in Table 4.3. This makes sense because when the intention points are closer to more likely endpoints, they are better able to produce accurate trajectories. This allows the model to select more diverse points and cover different interaction modes without an accuracy trade-off. Consequently, learnable intention points improves performance on most metrics, for both mode coverage and prediction performance.

	minADE (↓)	minFDE (↓)	MissRate (↓)	N_{modes} (↑)
k-means	**1.2396**	2.9160	0.8843	2.923
Learned	1.2419	**2.8993**	**0.8810**	**2.948**

Table 4.3: Interaction mode coverage and joint prediction metrics when using k-means intention points versus learnable intention points on the $N_{yield} = 3$ validation split.

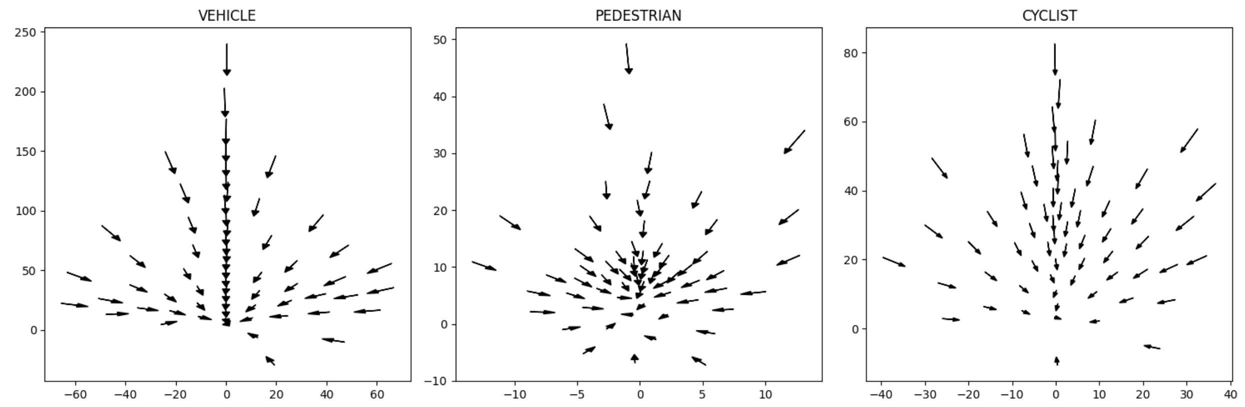


Figure 4.5: Intention points from k-means as in MTR versus learned intention points in our trained model. The tails of the arrows represent the original k-means intention points and the heads represent the corresponding learned points.

4.6.6 Qualitative Results

In Figure 4.6, we show a few examples of joint prediction samples output by our method. Each row in the image array corresponds to one traffic scenario and each image corresponds to one joint prediction sample. Our method outputs different interaction outcomes for tightly interacting groups of agents. The different outcomes include agents crossing shared space on the road in different orders and at various timings. In the first row, agent 8 is predicted to either U-turn or turn left after agents 2 and 6 cross the intersection. Our predictions account for different durations for which agent 8 may wait before executing its maneuver. In the second row, agents 22, 23, and 26 show different roundabout interaction outcomes corresponding to different speeds that agent 26 could take to execute its maneuver in the shared space. In the third row, the samples show agents 0, 5, and 6 crossing each other's paths with different times. In the fourth row, our outputs capture different possibilities for agents 1, 14, 20, and 124, including whether 20 follows 14 in a right turn or continues straight, whether 1 goes straight or yields and turns left after 20 passes, and whether 124 crosses the street in front of 1 in either situation.

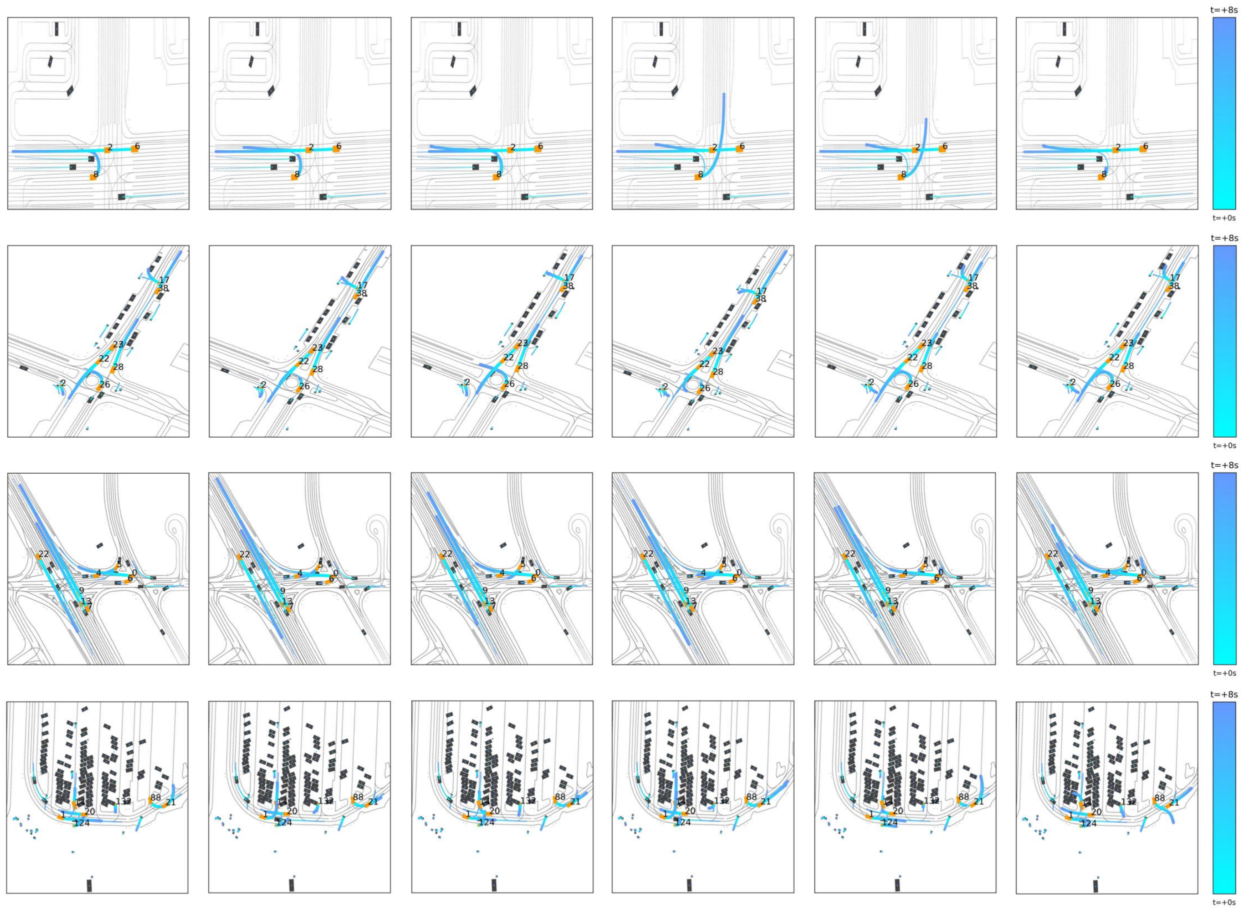


Figure 4.6: Examples of joint prediction samples output by our method. Each row corresponds to a traffic scenario and each image corresponds to one joint prediction sample. Target agents are in orange and other surrounding agents are in black. Thick blue lines represent predicted trajectories and thin blue lines are ground truth.

4.7 Conclusion

In this chapter, we presented a method to perform learned factorization of a hybrid system, and demonstrated how to leverage the factorization to improve interaction mode coverage for joint trajectory prediction. In our experiments, we showed that our method exceeded prior work in interaction mode coverage by 24% to 48% on validation splits of varying complexity while maintaining state-of-the-art performance in prediction error. Additionally, when increasing scenario complexity and size, our method demonstrated increasing margins of improvement. We also showed an improvement in sample efficiency by 33%. In our ablation studies, we showed that our discrete latent variable, grouping of agents, and ordering of groups for sampling discrete values all contributed to diversity of interaction outcomes. We demonstrated that learning the intention points helped improve mode coverage and prediction errors. For future work, it would be useful to explore how to improve the joint prediction weights in order to improve weighted measures of performance, such as mAP.

Chapter 5

Collision-Aware Best-First Enumeration for Multi-Agent Prediction

5.1 Introduction

Joint prediction of multiple traffic agents is challenging because the prediction space grows exponentially with the number of agents in a scene. In order to accurately approximate the joint prediction distribution, existing sampling-based methods designed for single-agent prediction [11], [60], [61] would require many samples due to the combinatorics of many agents acting concurrently. However, due to the computational complexity of processing predictions, only a small set of predictions can be processed for downstream risk assessment [62]. To cover the largest possible probability mass of the joint future distribution within the allocated number of samples, we leverage a best-first enumeration algorithm to form the joint trajectory outputs.

In this chapter, we model traffic scenarios as concurrent hybrid systems. The hybrid transition function factors into a discrete transition function and a continuous transition function, which are conditionally independent. The discrete transition function factors into a set of conditionally independent transition functions, one per agent. We use MTR [15] to describe the dynamics. We model interactions by assuming agents aim to avoid collisions with each other [36], [43]–[45]. We assume agents' discrete transitions are conditionally independent given collision avoidance behaviour, and use this factorization to enumerate assignments to discrete variables in best-first order using A*BC.

We begin enumerating based on the collision-agnostic prior distribution provided by MTR. We then extract bounding conflicts in a correction step to improve the distribution used by enumeration. This effectively reduces the error between the collision-agnostic prior and collision-aware posterior by correcting the prior using collision probability. These bounding conflicts enable us to prioritize search nodes that are more likely under the collision avoidance-based model of interaction, therefore leading to better scalability in the combinatorial challenge of multi-agent prediction. Section 5.4 provides further detail. Figure 5.1 illustrates an overview of our method.

The contributions of this chapter are as follows:

- We present a method that leverages A*BC to enumerate joint collision-aware predictions based on marginal collision-agnostic predictions. Our method combines hybrid models with the expressive power of deep neural networks to predict joint trajectories.

- We show that A*BC reduces the number of search nodes expanded and wall time required compared to A*, and is therefore better at addressing scalability.

- We compare joint prediction performance of the collision-agnostic and collision-aware predictions and find that our method performs worse than the collision-agnostic baseline. We present further analyses on specific subsets of data to provide insights on the performance gap and point to directions for future improvement.

In Section 5.2, we discuss how we model traffic scenarios for this approach. Then, Section 5.4 presents how we leverage this model and enumerate to form joint trajectory sets from multi-modal single-agent predictions. Experiments follow in Section 5.5.

5.2 Approach Overview

The multi-agent prediction problem follows the formulation presented in Chapter 2. We take as input the observed past trajectories O, scene context C, a single-agent predictor P, and a hybrid model of the traffic scene H (discussed in Section 5.3), and outputs a set of joint trajectory samples $S = \{(\mathbf{x}^{(k)}, w^{(k)})\}_{k=1}^{K}$. Our outputs are multi-modal, with continuous dynamics conditioned on discrete states (i.e., modes). In this chapter, we make two assumptions. First, we model traffic scenes with concurrent single-agent hybrid models, as described in Sections 5.3.1 and 5.3.2. Second, we assume that traffic agents interact in ways that avoid collision [36], [43]–[45]. This assumption is further explored and used in Sections 5.3.3 and 5.4.

5.3 Approach: Hybrid System Modelling and Inference

5.3.1 Agent-Level Model

We model the set of agents (cars, cyclists, and pedestrians) in a driving scenario as a set of concurrent hybrid discrete-continuous models. We represent multi-agent driving scenarios at two levels – the agent level and the scene level. An agent-level hybrid model describes a single agent. Formally, this is a tuple $\langle \mathbf{s}, \mathbf{w}, F, \mathbf{s}_0, \mathcal{Z} \rangle$, where:

- $\mathbf{s} = \mathbf{x} \cup \mathbf{z}$ denotes the hybrid state, where $\mathbf{z} \in \mathcal{Z}$ represents the discrete state variables with finite domain $\mathcal{Z}$ and $\mathbf{x} \in \mathbb{R}^{n_x}$ represents the continuous states.

- $\mathbf{w}$ denotes the input variables – the scene context C and past observations O.

- $F : \mathcal{Z} \to \mathcal{F}$ denotes the continuous dynamics function for each discrete mode as a set of learned discrete-time difference equations over the state and input variables.

- $\mathbf{s}_0$ represents the initial distribution of the hybrid state, at time $t = 0$.

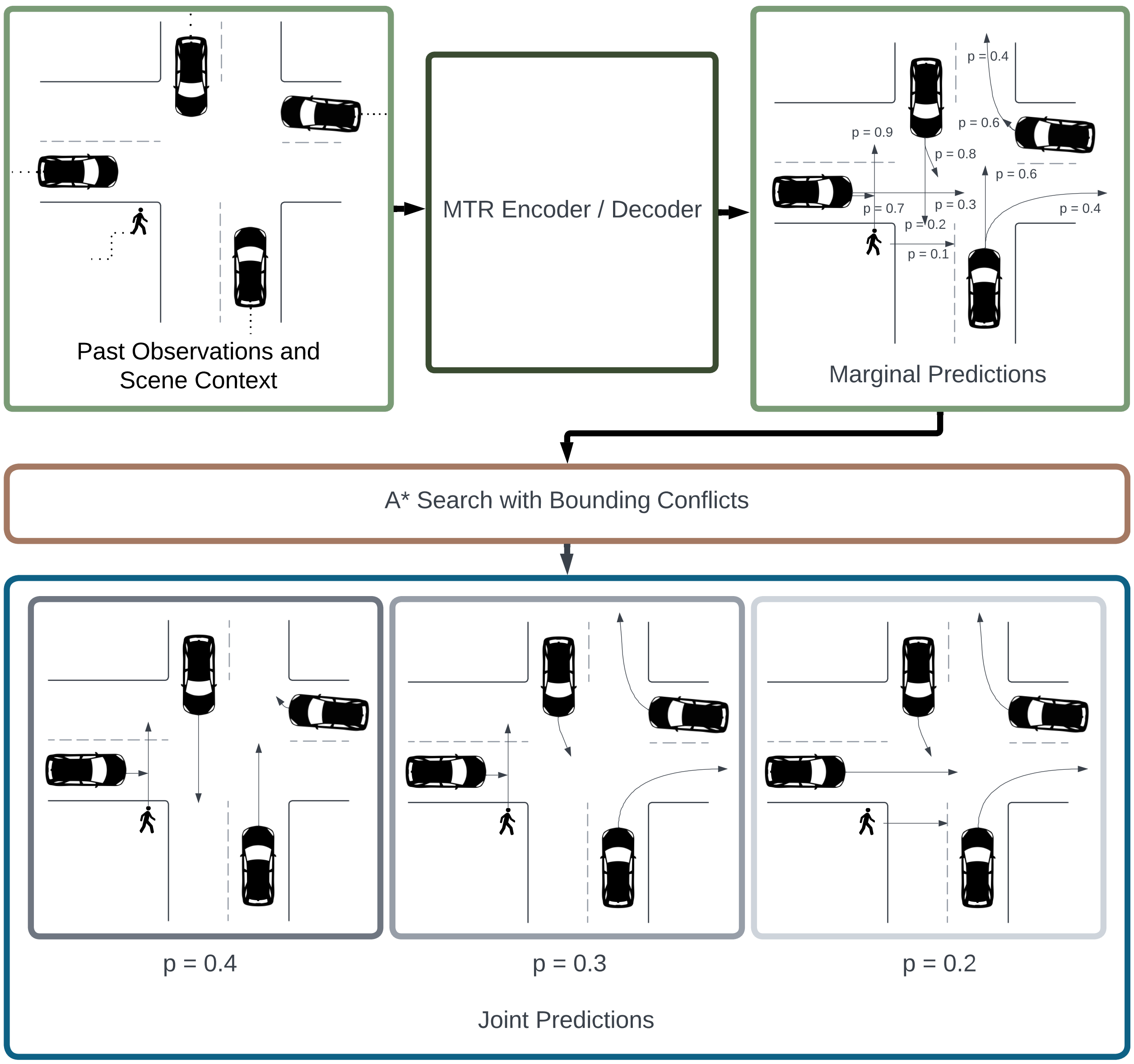


Figure 5.1: An overview of our approach. We leverage a single-agent trajectory predictor, MTR [15], to obtain weighted sets of marginal trajectories. Then, we use a best-first enumeration algorithm, A*BC [34] to obtain the top joint predictions from marginal predictions.

Intuitively, the discrete states represent high-level maneuvers and being in a mode corresponds to the agent taking the maneuver. The continuous transition function $F(z) = p(\mathbf{x}_{t:t+h-1}|O, C, \mathbf{z}_t = z)$ describes the continuous motion taken by the agent while performing that maneuver. The dependencies of the hybrid state variables are shown in Figure 5.2.

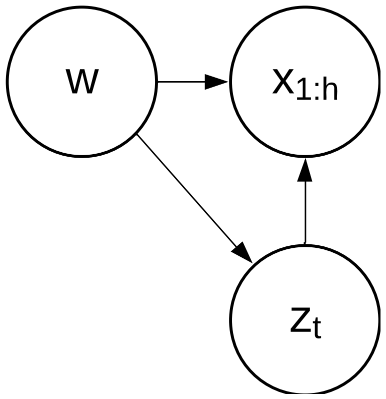


Figure 5.2: Graphical model of a hybrid model representing a traffic agent at $t = 0$. x represents the continuous variable, z represents the discrete variable, and h represents the length of the prediction horizon. Arrows represent dependencies.

5.3.2 Scene-Level Model

At the scene level, the hybrid models representing the agents in the scene are concurrently evolving. Instead of joining agent models into a single model with an exponentially larger state space, we keep the factored representation and exploit this structure for joint prediction via enumeration. Our scene-level model is described by a tuple $\langle \mathbf{s}, \mathbf{w}, F, \mathbf{s}_0, \mathcal{Z} \rangle$, where:

- $\mathbf{s} = \mathbf{x} \cup \mathbf{z}$, with $\mathbf{x} \triangleq \mathbf{x}_1 \cup ... \cup \mathbf{x}_N$ and $\mathbf{z} \triangleq \mathbf{z}_1 \cup ... \cup \mathbf{z}_N$
- $\mathbf{w} \triangleq \mathbf{w}_1 \cup ... \cup \mathbf{w}_N$
- $F(\mathbf{z}) \triangleq F_1(\mathbf{z}_1) \cup ... \cup F_N(\mathbf{z}_N)$
- $\mathbf{s}_0 \triangleq \mathbf{s}_{0,1} \times ... \times \mathbf{s}_{0,N}$
- $\mathcal{Z} \triangleq \mathcal{Z}_1 \times ... \times \mathcal{Z}_N$

We assume that the mode assignments (i.e., agents' maneuvers) are independent of each other when conditioned on the input variables. That is, $\{\mathbf{z}_{t,i}\}_{i=1}^{N}$ are conditionally independent given $\mathbf{w}$. In Figure 5.3, we show an example of a multi-agent prediction problem that can be modelled as a set of hybrid models.

In this chapter, we approximate continuous dynamics using a deep neural network trained for single-agent prediction on a large dataset [59]. Specifically, our method leverages MTR [15] as the continuous dynamics function (details in Section 3.5). We focus on scaling the single-agent prediction problem to multi-agent and perform enumeration for one prediction period h. We present the details of our method in Section 5.4.

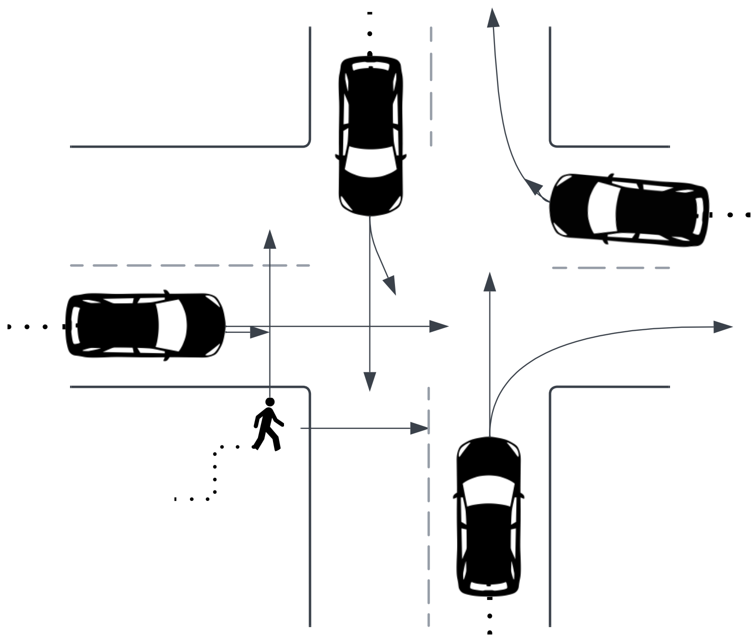


Figure 5.3: An illustration of a driving scene that can be modelled using set of concurrent hybrid models. We assume that discrete modes do not change over one prediction period. Dotted lines represent agents' past continuous observations, solid arrows represent trajectory prediction samples. Solid and dashed lines represent road boundaries and lane lines, respectively.

5.3.3 Collision-Aware Inference

The single-agent predictor, MTR, provides a set of weighted trajectories per agent $S_i = \{(\mathbf{x}^{(k)}_{1:T_f,i}, w^{(k)}_i)\}^K_{k=1}$ representing the multi-modal marginal distribution $p(\mathbf{x}_{1:T_f,i}|O, C)$. MTR forms joint predictions in a collision-agnostic manner by assuming independence of future trajectories given past observations O and context C. MTR's joint collision-agnostic distribution is therefore factored as the product of marginal distributions, with the probabilities $p(\mathbf{x}_{1:T_f,i}|O, C)$ corresponding to the trajectory weights w_i, as follows

$$p(\{\mathbf{x}_{1:T_f,i}\}^N_{i=1}|O, C) = \prod_{i=1}^{N} p(\mathbf{x}_{1:T_f,i}|O, C) \tag{5.1}$$

In this chapter, we treat the assignment of an agent to one of the K marginal trajectories as a discrete mode assignment, with the discrete variables representing the selection of the trajectory. Therefore, we can write the *joint collision-agnostic prior distribution* over discrete variables as

$$p(\{\mathbf{z}_i\}^N_{i=1}|O, C) = \prod_{i=1}^{N} p(\mathbf{z}_i|O, C) \tag{5.2}$$

where the probabilities $p(\mathbf{z}_i|O, C)$ are the weights w_i output by MTR.

Denoting the event of no collision in the prediction horizon $\mathcal{X}$ as a proposition that is true, the *joint collision-aware posterior distribution* that we aim to approximate is

$$p(\{\mathbf{z}_i\}_{i=1}^N|\mathcal{X}, O, C) \tag{5.3}$$

This distribution can be broken down into two parts according to Bayes' rule, as follows

$$p(\{\mathbf{z}_i\}_{i=1}^N|\mathcal{X}, O, C) \propto p(\mathcal{X}|\{\mathbf{z}_i\}_{i=1}^N, O, C)p(\{\mathbf{z}_i\}_{i=1}^N|O, C) \tag{5.4}$$

The first term on the right side represents the collision-free likelihood of the joint motion and the second term represents the joint collision-agnostic prior from Equation 5.2. We assume that collision-free probability is independent of past observations O and context C when given discrete maneuvers $\{\mathbf{z}_i\}_{i=1}^N$, so we can simplify the first term on the right side of Equation 5.4. We can also substitute Equation 5.2 for the second term. Therefore, our collision-aware distribution has the following factorization

$$p(\{\mathbf{z}_i\}_{i=1}^N|\mathcal{X}, O, C) \propto p(\mathcal{X}|\{\mathbf{z}_i\}_{i=1}^N) \prod_{i=1}^{N} p(\mathbf{z}_i|O, C) \tag{5.5}$$

We solve this inference problem using best-first enumeration to output joint collision-aware predictions that approximate Equation 5.5. This equation therefore forms the basis of our enumeration search cost, which we will describe in Section 5.4.1. In Section 5.3.4, we first discuss how we compute the collision-free likelihood $p(\mathcal{X}|\{\mathbf{z}_i\}_{i=1}^N)$. As previously mentioned, the second term on the right side of Equation 5.5 is the collision-agnostic prior that is obtained directly from MTR's output.

5.3.4 Computing Collision-Free Likelihood

In this section, we describe how to compute collision-free likelihood. First, we observe that there will be no collisions in the prediction horizon $\mathcal{X}$ if there are no collisions between any two agents in the scene. Letting $\mathcal{X}_{i,j}$ denote the event of no collision between agents i and j this is

$$\mathcal{X} = \bigwedge_{i=1,j=i+1}^{N-1,N} \mathcal{X}_{i,j} \tag{5.6}$$

We make two assumptions that allow us to factor and simplify $p(\mathcal{X}|\{\mathbf{z}_i\}_{i=1}^N)$. First, we assume that the events $\mathcal{X}_{i,j}$ are conditionally independent given the discrete assignments of future trajectories for all agents $\{\mathbf{z}_i\}_{i=1}^N$ (Equation 5.8). Second, we assume that $\mathcal{X}_{i,j}$ is only dependent on the future trajectories of agents i and j (Equation 5.9). With this, we can factor $p(\mathcal{X}|\{\mathbf{z}_i\}_{i=1}^N)$ as

$$p\left(\mathcal{X}|\{\mathbf{z}_i\}_{i=1}^N\right) = p\left(\bigwedge_{i=1,j=i+1}^{N-1,N} \mathcal{X}_{i,j}|\{\mathbf{z}_i\}_{i=1}^N\right) \tag{5.7}$$

$$= \prod_{i=1,j=i+1}^{N-1,N} p\left(\mathcal{X}_{i,j}|\{\mathbf{z}_i\}_{i=1}^N\right) \tag{5.8}$$

$$= \prod_{i=1,j=i+1}^{N-1,N} p\left(\mathcal{X}_{i,j}|\mathbf{z}_i, \mathbf{z}_j\right) \tag{5.9}$$

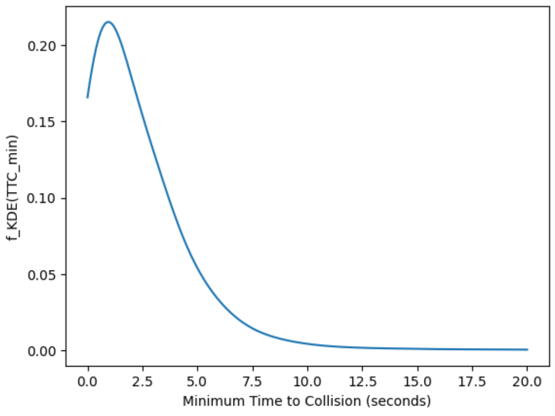
According to this factorization, we can compute the collision-free likelihood $p\left(\mathcal{X}|\{\mathbf{z}_i\}_{i=1}^N\right)$ by computing the probability of each pair of agents colliding given their discrete assignment, $p(\mathcal{X}_{i,j}|\mathbf{z}_i, \mathbf{z}_j)$, and taking the product.

In practice, it is not possible to compute $p\left(\mathcal{X}_{i,j}|\mathbf{z}_i, \mathbf{z}_j\right)$ because collisions are rare in reality and not present in naturalistic datasets. As a result, we need to compute a quantity that can be a proxy for $p\left(\mathcal{X}_{i,j}|\mathbf{z}_i, \mathbf{z}_j\right)$. In our method, we assume that risky driving behaviour leads to higher collision likelihoods and therefore higher costs. Risky behaviour can be captured by how close an agent comes to colliding with another, measured by the time-to-collision.

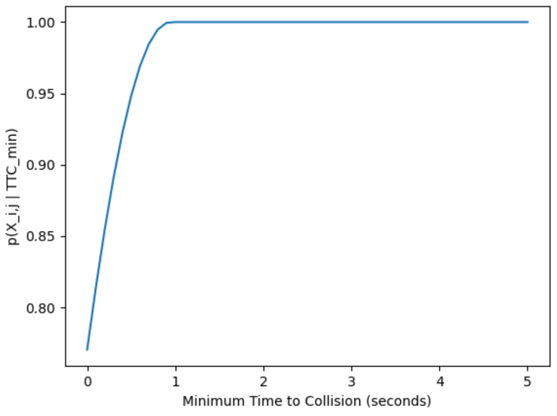
We therefore use the *minimum time-to-collision* (TTC_{min}) between two agents' trajectories ($\mathbf{z}_i$ and $\mathbf{z}_j$) to determine $p\left(\mathcal{X}_{i,j}|\mathbf{z}_i, \mathbf{z}_j\right)$. At each future time step, we compute TTC by dividing pairwise distance between agents by their relative velocities. We take the lowest TTC across all future time steps as TTC_{min}. To construct the probability density function of TTC_{min}, we compute the TTC_{min} over every pair of agents in an interactive subset of the training data and fitting a kernel density estimate of the TTC_{min} samples. The resulting kernel density estimate, which we denote as $f_{KDE}(TTC_{min})$, is shown in Figure 5.4a.

The probability density at higher TTC_{min} is low due to infrequent appearance in interactive data samples. In reality, however, higher TTC_{min} corresponds to lower probabilities of collision, or higher $p(\mathcal{X}_{i,j}|\mathbf{z}_i, \mathbf{z}_j)$. Thus, for all values in $TTC_{min} > \arg\max_{TTC_{min}} f_{KDE}(TTC_{min})$, we set $p(\mathcal{X}_{i,j}|\mathbf{z}_i, \mathbf{z}_j) = 1$. For all other values of TTC_{min}, we set $p(\mathcal{X}_{i,j}|\mathbf{z}_i, \mathbf{z}_j) = \eta f_{KDE}(TTC_{min})$, where η is a scaling factor to rescale the output of this function to range from 0 to 1. In summary, Equation 5.10 details how we compute $p(\mathcal{X}_{i,j}|\mathbf{z}_i, \mathbf{z}_j)$, and we show the corresponding plot in Figure 5.4b.

$$\begin{aligned} p(\mathcal{X}_{i,j}|\mathbf{z}_i, \mathbf{z}_j) &= \\ p(\mathcal{X}_{i,j}|TTC_{min}) &= \begin{cases} \eta f_{KDE}(TTC_{min}), & TTC_{min} \leq \arg\max_{TTC_{min}} f_{KDE}(TTC_{min}) \\ 1, & \text{otherwise} \end{cases} \end{aligned} \tag{5.10}$$



(a) Kernel density estimate of TTC_{min} in an interactive subset of the training data.



(b) Probability of no collision $\mathcal{X}_{i,j}$ given TTC_{min} for agents i and j.

Figure 5.4: Plots showing the probability density of TTC_{min} (left) and probability of no collision given TTC_{min} (right).

5.4 Approach: From Single- to Multi-Agent Prediction via Best-First Enumeration

In this section, we describe how we use best-first enumeration for scaling single-agent prediction to multi-agent prediction. We leverage a single-agent predictor, MTR [15], to obtain a set of marginal trajectories per agent, $S_i = \{(\mathbf{x}^{(k)}_{1:T_f,i}, w^{(k)}_i)\}^K_{k=1}$, with each of the K trajectories representing a different mode. We search through possible combinations of discrete mode assignments for the agents in a given scene to enumerate joint predictions in best-first order according to our collision-aware distribution. In this work, a mode assignment represents the discrete selection of a particular continuous trajectory. Therefore, a mode assignment $\mathbf{z}_i$ to agent i directly assigns the joint continuous states $\mathbf{x}_{1:T_f,i}$ without further computation.

Our method forms joint trajectory sets from marginal trajectories in a way that considers the compatibility of behaviours, rather than assuming futures are independent of each other. When considering driving scenarios, we take advantage of the intuitive observation that agents tend to interact in ways that avoid collision [36], [43]–[45], [63]. We incorporate this as an assumption into our search cost function by enumerating the collision-aware posterior distribution $p(\{\mathbf{z}_i\}^N_{i=1}|\mathcal{X}, O, C)$ rather than the collision-agnostic prior $p(\{\mathbf{z}_i\}^N_{i=1}|O, C)$. We begin by providing details on how we obtain the search cost in Section 5.4.1.

5.4.1 Collision-Aware Distribution as a Search Cost for Best-First Enumeration

First, we introduce our enumeration search cost. Following the notation defined in Section 5.3, let $\mathbf{z}_i$ denote the discrete choice of a particular marginal trajectory for agent i. MTR outputs K marginal trajectories per agent, so the domain size of $\mathbf{z}_i$ is K ($|\mathcal{Z}_i| = K$).

The cost function takes the form of $f(n) = g(n) + h(n)$ for search node n, where $g(n)$ is the incurred cost of a partial assignment to the discrete variables and $h(n)$ is an admissible heuristic cost for the remaining unassigned variables. Admissibility of the heuristic guarantees that joint predictions are enumerated in best-first order. This particular search cost formulation is known as A* search, which guarantees optimality. To convert the Equation 5.5 to a search cost that we minimize, we take the negative logarithm and obtain

$$f(n) = f_1(n) + f_2(n) \tag{5.11}$$

$$f_1(n) \triangleq -\log p(\mathcal{X}|\{\mathbf{z}_i\}_{i=1}^N) \tag{5.12}$$

$$f_2(n) \triangleq -\log p(\{\mathbf{z}_i\}_{i=1}^N|O, C) \tag{5.13}$$

Both terms of the cost function will have corresponding incurred and heuristic costs. At a given search node, the incurred costs represent the costs associated with the trajectory selections (i.e., discrete assignments) that have already been made. The heuristic costs represent an optimistic estimate of the cost to assign trajectories to the remaining agents.

We first look at the second term of the cost function, $f_2(n)$. According to the factorization in Equation 5.2, this becomes

$$f_2(n) = \sum_{i=1}^{N} -\log p(\mathbf{z}_i = z_i|O, C) \tag{5.14}$$

This can be computed from the weights output by MTR based on the factorization in Equation 5.2, as described in Section 5.3.3, by substituting the corresponding weights w_i into $p(\mathbf{z}_i = z_i|O, C)$. The first part of the search cost, $g_2(n)$, is the cost for the partial assignment at node n, shown in Equation 5.16. The second part of the search cost, $h_2(n)$, is an admissible heuristic cost that estimates the remaining cost to reach a full assignment (full joint trajectory set). An admissible heuristic to approximate this cost would be to simply assume all unassigned agents take their most likely trajectory. To summarize,

$$f_2(n) = g_2(n) + h_2(n) \tag{5.15}$$

$$g_2(n) = \sum_{i \in n} -\log p(\mathbf{z}_i = z_i|O, C) \tag{5.16}$$

$$h_2(n) = \sum_{j \notin n} -\log \max_{z_j \in \mathcal{Z}_j} p(\mathbf{z}_j = z_j|O, C) \tag{5.17}$$

For $f_1(n)$, we wait until reaching a node with a full assignment before computing collision-free likelihood. Although our factorization in Equation 5.9 enables us to compute the likelihood after two discrete assignments, the computation is expensive so we compute the full collision-free likelihood only at leaf nodes. This is reflected in Equation 5.19. For the heuristic, we initialize it to $h_1(n) = 0$, which is an optimistic assumption that remaining assignments will be collision-free. As search progresses, $h_1(n)$ will be increased to provide a heuristic that more closely estimates the collision-aware posterior. We describe this process in Section 5.4.2. To summarize,

$$f_1(n) = g_1(n) + h_1(n) \tag{5.18}$$

$$g_1(n) = \begin{cases} -\log p(\mathcal{X}|\{\mathbf{z}_i\}_{i=1}^N), & \text{if } n \text{ is a full trajectory set} \\ 0, & \text{otherwise} \end{cases} \tag{5.19}$$

$$h_1(n) = 0 \tag{5.20}$$

5.4.2 Extracting Bounding Conflicts using Probability of Collision

During the enumeration process, we extract bounding conflicts to gradually tighten the heuristic cost for collision-free likelihood, $h_1(n)$. This acts as a correction step on the prior to reduce the error between the prior and collision-aware posterior. To extract bounding conflicts, we consider the trajectory selections that contribute to a lower collision-free likelihood (or higher cost). As a result, we can use our collision-free likelihood factorization in Equation 5.9 and calculations from Section 5.3.4 to determine a subset of trajectory selections that contribute to lower collision-free likelihood. We can then associate a tighter cost bound with these trajectory selections and use the bound in the remainder of the search process to update $h_1(n)$. This idea of extracting bounding conflicts during A* search is known as A* with bounding conflicts (A*BC), first developed in [34].

The goal is to identify contributors to low probability $p\left(\mathcal{X}|\{\mathbf{z}_i\}_{i=1}^N\right)$. We can do this by taking the factorization from Equation 5.9. We can extract a bounding conflict for each pair of discrete assignments $\mathbf{z}_i = z_i$ and $\mathbf{z}_j = z_j$ where $p\left(\mathcal{X}_{i,j}|\mathbf{z}_i = z_i, \mathbf{z}_j = z_j\right) < 1$. The corresponding cost bound will equal $-\log p\left(\mathcal{X}_{i,j}|\mathbf{z}_i = z_i, \mathbf{z}_j = z_j\right)$. In the rest of the enumeration process, any partial assignment containing assignments of $\mathbf{z}_i = z_i$ and $\mathbf{z}_j = z_j$ will have a heuristic cost $h_1(n)$ of at least $-\log p\left(\mathcal{X}_{i,j}|\mathbf{z}_i = z_i, \mathbf{z}_j = z_j\right)$. This cost is a tighter lower bound than the initial $h_1(n) = 0$.

By extracting bounding conflicts and improving the heuristic, we expect to skip over regions of the search tree that are considered unlikely according to collision likelihood. We expect bounding conflicts to be particularly useful on highly interactive multi-agent scenarios. The more interactions involving low and unlikely times-to-collision, the more generalization we expect to see from bounding conflicts, leading to fewer search nodes expanded.

5.5 Experiments

In this section, we evaluate the prediction accuracy of our method. We do so by training the marginal trajectory predictor, MTR [15], on a real world driving dataset, applying our method to form joint trajectory predictions, and evaluating the joint predictions. First, we introduce the dataset and model details and then present our results.

5.5.1 Dataset and Method Details

We train and validate MTR on the Waymo Open Motion Dataset [59] following the hyperparameters in [15]. The dataset contains $487k$ driving scenes in the training split, 44097 scenes in the regular validation split, and 43479 scenes in the interactive validation split, all at $10Hz$. The regular validation split consists of up to eight agents, and we use this to evaluate scalability with enumeration. The interactive validation split consists of two agents labelled to be interacting. We use the interactive split to evaluate the prediction performance of our collision-aware model. In a driving scene, each agent of interest has an observed history of up to one second. Ground truth future trajectories are provided for up to eight seconds into the future. To construct the kernel density estimate for TTC_{min}, we compute the TTC_{min} using the subset of training scenes that contain two agents labelled as objects of interest, indicating interactive behaviour. The scaling factor for our kernel density estimate is $\eta = 4.65$. We use the official evaluation tool to calculate metrics including mean average precision (mAP), miss rate, minimum average displacement error (minADE) and minimum final displacement error (minFDE). Following the specifications of the Waymo benchmark, we enumerate $K = 6$ joint trajectory sets per traffic scenario.

The architecture of the model is exactly as described in [15]. To align with the design of MTR and the evaluation for the Waymo dataset, we use $h = 8$ seconds as the time interval over which the discrete state is assumed constant. In this work, the discrete state represents the discrete choice of a particular marginal trajectory, which intuitively represents a maneuver. We run A*BC to enumerate joint predictions for the first time interval. This corresponds to the provided ground truth in the dataset, and we use this to evaluate the joint outputs of our method. It is possible to extend this method to multiple time horizons, and we describe strategies in Section 5.7.2, but we leave this as future work.

5.5.2 Analyzing the Efficiency of A*BC

Here, we analyze the efficiency of A*BC compared to A* and an exhaustive search baseline. All three search methods enumerate in best-first order according to the same collision-aware distribution, namely $p(\{\mathbf{z}_i\}_{i=1}^{N}|\mathcal{X}, O, C)$. Exhaustive search computes the collision-aware posterior for every possible K^N combinations of mode assignments and outputs the best K and A* search is done without extracting bounding conflicts. For this experiment, we use the regular validation split of the Waymo data consisting of up to eight agents. We look at efficiency benefits 3-agent and 4-agent scenarios and include results for exhaustive search on these splits. We also filter the data for scenarios that are more interactive by looking at networked yielding agents (as done in Chapter 4). We report results on subsets containing *at least* three and *at least* four agents engaged in a networked yielding interaction. We omit the exhaustive search baseline for the networked yield subsets and the full validation set due to the the exponential time cost required with more agents. We obtain joint predictions on these subsets, and report the average number of expanded nodes and wall time per traffic scenario in Table 5.1. Experiments were run in a docker container allotted eight threads of an Intel i9-10980XE CPU and a NVIDIA RTX A5500 GPU.

From Table 5.1, we can see that extracting bounding conflicts during the search process allows us to enumerate the top K joint trajectories with fewer node expansions, as expected. In A*BC, extracting bounding conflicts allows us to generalize insight we gain about a full mode assignment to tighten the search heuristic as search progresses. This effectively reorders the search queue to avoid expanding unlikely nodes early in the search process. Overall, we see that best-first enumeration with A*BC scales better to more agents than A*, reducing

Subset	Search Method	Nodes Expanded	Wall Time (sec)
3-agent	Exhaustive	216	3.58
	A*	13.23	0.72
	A*BC	12.80	0.49
4-agent	Exhaustive	1296	44.51
	A*	16.60	1.36
	A*BC	16.16	0.95
3+ yield network	A*	27.78	4.42
	A*BC	27.00	4.39
4+ yield network	A*	32.24	5.71
	A*BC	31.16	5.14
Validation (full)	A*	19.01	2.06
	A*BC	18.52	1.91

Table 5.1: Average number of search nodes expanded and wall time (seconds) per traffic scenario with exhaustive search, A*, and A*BC for joint prediction on the validation split. The bottom section of the table are for data splits involving up to eight agents.

wall time by over 30% on three and four-agent scenarios. For the full validation and network yield sets which have up to eight agents, there is a smaller reduction in wall time. This is likely because when there are more agents in the scene, a greater percentage of them are non-interactive, whereas the three and four-agent subsets have a higher percentage of agents that are interacting. In a traffic scenario with many agents, it is unlikely that all are involved in an interaction, and therefore we gain less generalizable information with the bounding conflicts in proportion to the full prediction space. Regardless, we see an improvement in nodes expanded and wall time.

Note that in this section, comparison of prediction accuracy and error metrics between exhaustive search, A*, and A*BC is irrelevant. This is because all three search methods produce joint trajectories in best-first order according to our collision-aware distribution. Prediction metrics are therefore the same for all three alternatives.

5.5.3 Comparing Prediction Performance of the Collision-Agnostic and Collision-Aware Distributions

In this section, we compare the prediction performance of the collision-agnostic distribution from MTR (Equation 5.2) and the collision-aware distribution in our method (Equation 5.3) to evaluate how well our collision-avoidance method models interactive joint behaviour. Note that in this section, we are only comparing prediction performance metrics for the two distributions and the choice of best-first enumeration algorithm is irrelevant because they will produce the same joint trajectories.

We report results on the interactive validation split of the Waymo dataset [59]. The interactive set consists of driving scenes in which two interacting agents of interest are identified for the prediction task. We further show performance on subsets of this data to evaluate performance on specific interaction types, including yield, follow, and non-interactive. Details on how these scenarios are filtered are presented in Appendix A. We also evaluate on the regular validation set consisting of up to eight agents, and subsets of the regular validation set that are filtered for networked interaction scenarios, specifically networked yielding scenarios. Scenarios with networked interactions are more complex and since our model of interactions aims to represent such joint behaviour, we evaluate performance on these. We show results for two splits containing networked yielding scenarios – one where networks consist of at least 3 agents and the other with at least 4 agents. The results are shown in the first section of Table 5.2.

Data Split	Distribution	mAP (↑)	minADE (↓)	minFDE (↓)	Miss Rate (↓)
Validation Interactive (Full)	collision-agnostic	0.2065	0.9372	2.1134	0.4346
	collision-aware	0.1170	1.0178	2.3550	0.4926
Yield	collision-agnostic	0.1306	1.0356	2.3542	0.5006
	collision-aware	0.0624	1.1201	2.6095	0.5615
Follow	collision-agnostic	0.2129	0.9355	2.0800	0.4235
	collision-aware	0.1078	1.0192	2.3273	0.4833
Non-Interactive	collision-agnostic	0.3756	0.6417	1.4495	0.3342
	collision-aware	0.2838	0.6756	1.5329	0.3415
Validation (Full)	collision-agnostic	0.1982	1.1248	2.5320	0.6960
	collision-aware	0.1626	1.1732	2.6924	0.7229
3+ yield network	collision-agnostic	0.0561	1.2580	2.9460	0.8768
	collision-aware	0.0183	1.3040	3.1038	0.8938
4+ yield network	collision-agnostic	0.1023	1.3246	3.0153	0.9291
	collision-aware	0.0331	1.3747	3.2096	0.9427

Table 5.2: Joint prediction metrics on various subsets of the regular and interactive validation splits. The top section of the table corresponds to two-agent data and the bottom section consists of scenarios with up to eight agents. ↑ indicates higher is better and ↓ indicates lower is better.

Based on the results from Table 5.2, we find that our collision-aware distribution does not perform as well as the collision-agnostic distribution. We can see that among the two-agent data splits, our approach on the non-interactive subset of data has the smallest performance degradation compared to the collision-agnostic distribution. This suggests that our time-to-collision-based criteria for determining likelihood is not an accurate reflection of interactive behaviour. There can be several reasons for this. First, the collision avoidance criteria based on minimum time-to-collision could be assigning higher costs to yielding or following scenarios because these interactions can involve closer agent proximity. As a result, the most accurate maneuver selection is assigned a higher cost, leading to poorer performance. Second, abstracting joint future behaviour into a single value – the minimum time-to-collision

– removes a lot of information about the behaviours of two interacting agents. It would be interesting to investigate as future work other ways to compute collision-aware joint trajectory likelihoods.

5.5.4 Qualitative Results

In Figure 5.5, we show a few examples of joint predictions made by our method. Each row in the image array corresponds to one traffic scenario and each image corresponds to one joint prediction sample. The shade of blue represents the associated future time step. Our method outputs a variety of possible joint futures without collisions while capturing different maneuvers per agent.

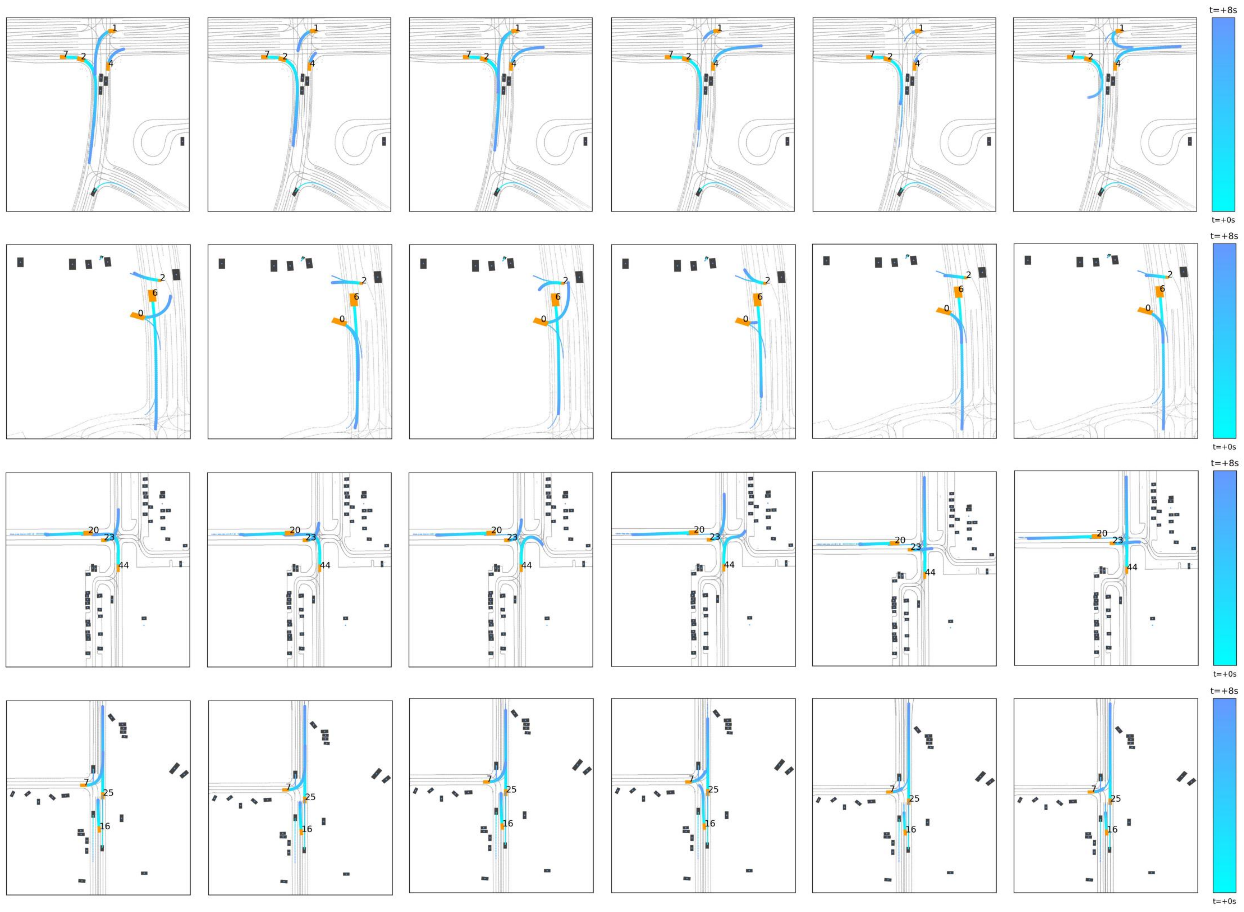


Figure 5.5: Examples of joint prediction samples output by our method. Each row corresponds to a traffic scenario and each image corresponds to one joint prediction sample. Target agents are in orange and other surrounding agents are in black. Thick blue lines represent predicted trajectories and thin blue lines are ground truth.

5.6 Related Work in Multi-Agent Prediction

A different method that addresses multi-agent prediction from the perspective of combining marginal predictions is introduced in [64]. This method represents interactions between agents as a graph and edges are associated with a $k \times k$ pairwise potential matrix, representing joint consistency between k marginal trajectories per agent. This pairwise method is more tractable than one that jointly models all agents in the scene. The reduction of the shared space into a set of pairwise joint spaces is most similar to our method, though ours relies instead on a handcrafted heuristic for enumeration.

A number of other methods approach prediction by factoring the joint distribution of agent futures and addressing conditional prediction [5], [7], [42], [53]. The resulting joint trajectories are limited to a query future trajectory. At the core of these predictors is a single-agent predictor that has strong performance, such as [22] and [14]. Similar to this idea of building on powerful single-agent predictors, we leverage MTR [15] for multi-agent prediction.

Another group of methods aim to predict the full joint distribution by assuming agents' states at time t are conditionally independent given all agents' states up to $t-1$ [17], [35], [65]. In our method, by modelling the traffic scene as concurrent hybrid models, we make a similar assumption but with the discrete modes. We assume agents' future discrete modes are conditionally independent given all agents' past states. A difference is that at each time step in the prediction horizon, the prior methods autoregressively condition on all agents' predictions from previous time steps and assume independent transitions for one step. On the other hand, in our setup, we assume modes are fixed for a duration of $h = 8\,seconds$, so we roll out a future trajectory for the full prediction period and leverage a strong single-agent predictor to maintain accuracy within h, without conditioning on other agents' predictions at every time step.

5.7 Limitations and Future Work

This chapter is a first step towards leveraging A*BC for joint prediction with traffic scenarios modelled as sets of concurrent agent-level hybrid models. We developed a method to extract bounding conflicts for enumerating over the space of possible joint predictions, given marginal trajectories of agents. There are some limitations of our method which provide interesting future directions.

5.7.1 Improving the Exact Inference Model

The performance and efficiency of A*BC on the trajectory prediction task depends on the distribution we are trying to approximate. To improve prediction performance, it would be most effective to improve the joint distribution, specifically the shared variable $\mathcal{X}$. This variable is directly involved in the cost computations and bounding conflict extraction procedure, both of which are key in providing performance and efficiency benefits.

First, we made some simplifying assumptions about pairwise collision avoidance to enable factorization for bounding conflict extraction. We assume minimum time-to-collision was sufficient to represent such behaviour. In reality, however, interactions involve elements beyond collision avoidance. For instance, traffic rules and environmental context (lane lines, crosswalks, intersections, drivable area, etc.) affect interaction outcomes and agent behaviour [44], [45], [66]. An interesting future direction would be to incorporate these into the inference problem, and determining a different way to factor the posterior distribution. This would form a model that is more representative of reality. When considering scene compliance, pre-pruning of trajectories can also be considered [21], [67]. Additionally, it would be interesting to use neural networks and explore learning a cost function from data. Ultimately, it is difficult to develop a full taxonomy of joint driving behaviours, so this would be a crucial step in improving performance.

5.7.2 Scaling to Multiple Prediction Intervals

Our method primarily focuses on scaling from single-agent to multi-agent prediction and assumes the discrete mode does not change over the full prediction horizon. However, our concurrent hybrid model paves the way to model traffic scenarios as concurrent probabilistic hybrid automata [28], in which discrete modes change. Our framework enables straightforward scaling to more than one prediction horizon. This can take the form of longer prediction horizons or subdividing the 8-second horizon in this chapter into multiple shorter intervals. To incorporate multiple prediction intervals into our method, the same A*BC procedure can be adopted with one minor addition. A discrete transition function could be learned to model how high-level behaviours (i.e., the discrete variables) change from one time horizon to the next. That is, a neural network can be trained to represent $T(z) = p(\mathbf{z}_t | \mathbf{z}_{t-h} = z, \mathbf{x}_{t-h:t-1})$.

The MTR backbone can be modified to experiment with different prediction horizons. In this chapter, we used a prediction horizon h of eight seconds to follow MTR [15] and align with the Waymo benchmark. However, agents can take a variety of maneuvers within this time interval. As a result, it would be interesting to investigate performance when the method is modified to handle the eight seconds in smaller intervals with a learned transition function between intervals.

5.8 Conclusion

In this chapter, we first developed a model for joint behaviour that assumes the likelihood of an agent's behaviour is conditioned on it being collision-free with other agents. We then introduced a method for factored inference based on A*BC. Our method begins by enumerating according to a collision-agnostic prior of future behaviour. During enumeration, we extracted bounding conflicts to correct the prior and reduce the error between the prior and our collision-aware posterior.

In our experiments, we showed that A*BC has a lower wall time and enables us to expand fewer nodes than A*, demonstrating better scalability. We also showed that our collision-aware distribution method achieves worse prediction performance than the collision-agnostic distribution. In particular, our method struggles in scenarios that are more interactive, suggesting room for improvement in the joint behaviour model. Future work could investigate developing a more sophisticated method of computing collision probability, or design a different distribution based on other factors in addition to collision. Another next step could be to accomplish joint prediction for multiple prediction intervals.

Chapter 6

Comparing Learned Factorization and Sampling with Collision-Aware Best-First Enumeration

In this Chapter, we compare our two methods. We show joint prediction performance metrics along with wall time. For computing average wall times, we set the evaluation batch size to one scene for both methods. We present the results on the regular validation split of the Waymo dataset [59], which consists of 44097 traffic scenarios of up to eight target agents. We also present the results for two subsets containing networked yielding scenarios – one where networks consist of at least 3 agents and the other with at least 4 agents. Past observations are provided for up to one second and ground truth futures for up to eight seconds, both at $10Hz$. Experiments were run in a docker container allotted eight threads of an Intel i9-10980XE CPU and a NVIDIA RTX A5500 GPU. Results are summarized in Table 6.1.

From this table, we see that our approach in Chapter 4 is faster, suggesting better scalability. This can be explained by the fact that computations and sampling in Chapter 4 can be accelerated on a GPU, whereas the search process in enumeration involves steps that do not benefit from GPU acceleration. Nonetheless, we showed in Chapter 5 that our method enabled us to generalize information during search and achieve a wall time improvement over A*.

We also observe that the method in Chapter 4 performs better on joint prediction metrics overall, especially on the complex interactive splits. This can likely be attributed to a more representative factorization of the traffic scene, which is identified using a trained neural network rather than completely assumed beforehand, as in Chapter 5. The method in Chapter 4 is also designed in a way where the learned factorization can model interactions in a general manner rather than focusing on collision avoidance behaviour, which does not fully capture all the nuances of interaction in driving.

The comparisons presented in this chapter lead to potential avenues for future work, which we discuss in Section 7.2.

Data	Method	mAP (↑)	minADE (↓)	minFDE (↓)	Miss Rate (↓)	Wall Time (↓)
3+ yield network	Chapter 4	0.0243	1.2419	2.8993	0.8810	0.71
	Chapter 5	0.0183	1.3040	3.1038	0.8938	4.39
4+ yield network	Chapter 4	0.0600	1.2969	3.0276	0.9267	1.04
	Chapter 5	0.0331	1.3747	3.2096	0.9427	5.14
Validation (Full)	Chapter 4	0.1264	1.1450	2.6176	0.7266	0.53
	Chapter 5	0.1626	1.1732	2.6924	0.7229	1.91

Table 6.1: Comparison of joint prediction metrics and wall time (seconds) of our two methods on the Waymo validation dataset. All data splits consist of up to eight agents.

Chapter 7

Conclusions

7.1 Summary of Contributions

In this thesis, we presented two methods for joint multi-agent prediction that leveraged factoring to address the challenges of interaction coverage and scalability. In our first approach (Chapter 4), we modelled interactions among multiple agents by grouping agents according to tightly coupled behaviours and assuming interactions between groups are sparse. We used these groups and sparse inter-group relationships to factor the joint distribution in a way that enables simple and efficient sampling of discrete interaction outcomes. We learned to make this decomposition using neural networks and showed that our method achieves superior interaction mode coverage and sample efficiency compared to prior work, while maintaining state-of-the-art prediction errors.

In our second approach (Chapter 5), we assumed that agents interact in a way that avoids collision. We modelled agents as concurrent hybrid models and formed a factored collision-aware joint distribution. With this factored distribution, we leveraged A*BC to approximate the collision-aware distribution. We began enumerating joint trajectory candidates according to a simpler, collision-agnostic prior distribution obtained from [15]. Throughout the enumeration process, we learned bounding conflicts to correct the collision-agnostic prior and gradually reduce the error between the distribution used for enumeration and the posterior. The use of bounding conflicts enabled us to skip over regions of the search tree that are unlikely. We compared A* with A*BC and found that A*BC expanded fewer search nodes and required a lower run time, and is therefore better at addressing scalability. We compared prediction performance of our joint collision-aware predictions with collision-agnostic predictions and found that ours led to worse errors and accuracy. Nonetheless, this work provides a first step towards scalable collision-aware prediction using best-first enumeration.

We compared the performance of our approaches in Chapter 6. We found that our first method performed better on prediction metrics and required a lower runtime. These findings provide some insights that pave the way for future work, which we discuss in Section 7.2.

7.2 Future Work

There are several potential avenues for future work. For our learned factorization method in Chapter 4, our experiments revealed that despite maintaining state-of-the-art performance on unweighted error metrics, our method performed worse than prior work on mean average precision (mAP), the only metric that considers joint trajectory weights. This result indicates that improving the weights would be an important next step to improving mAP, as mentioned in Section 4.7.

For Chapter 5, we discussed in detail in Section 5.7.1 the ways to improve prediction performance. Incorporating other elements into the inference model for joint prediction, such as road structure, instead of limiting it to collision avoidance could help performance. One caveat would be that the target distribution must take a factored form that enables simple bounding conflict extraction. Otherwise, the time cost incurred by bounding conflict extraction could outweigh the benefits of skipping over search nodes. Another potential direction, described in Section 5.7.2, would be to extend the model of the traffic scene to hybrid automata. Instead of assuming a constant discrete mode throughout the entire 8-second prediction period, it could be subdivided into multiple intervals and a discrete mode transition function could be learned.

Another possibility could be to combine enumeration with sampling [30] for approximate inference. This could be applied to either the learned factorization (Chapter 4) or collision-aware concurrent hybrid model (Chapter 5) to further explore options for scalability. For instance, with the hybrid model in Chapter 5, enumeration can be used to obtain the top few joint trajectories and the remaining ones can be sampled. The number of enumerated versus sampled predictions can be adjusted according to any potential time and resource constraints.

We could also explore combining enumeration and sampling on the learned factorization in Chapter 4. This would allow us to take advantage of a learned hybrid model of the traffic scene, rather than a model based only on prior assumptions. There are several technical challenges with this. To enumerate according to a learned factorization, an open problem is to consider how to handle the fact that different discrete variable assignments in DAG predecessors change the discrete distribution at successor nodes. Conditioning structure across discrete variables is not currently handled in our method in Chapter 5 as we assumed conditional independence. If attempting to also use enumeration during neural network training, an important step would be to ensure all operations are differentiable.

Appendix A

Filtering for Interactions

In this appendix, we describe how we filter for particular interaction types throughout this thesis.

A.1 Yield

As identified by prior work such as [5], [6], yielding interactions are prevalent in traffic scenarios. To filter for these, we consider agent i to be yielding to agent j if the following criteria are met:

- The paths of agents i and j overlap.
- Agent j arrives at the intersecting area first.
- Agent i has a non-increasing speed.

A.2 Follow

When one agent follows another, their behaviour could be influenced by the agent in front. For example, if the agent in front slows down, the following agent is also likely to slow down. Agent i is considered to be following agent j if

- The paths of agents i and j overlap.
- Agent j arrives at the intersecting area first.
- The duration of time that both agents spend within the intersecting area is at least t seconds.

In our implementation, we set $t = 1$.

A.3 Intersect

Intersecting trajectories can help us identify interactions because they indicate the need for agents to share space on the road. This filter is used to include any interactions that may not be identified by the more specific yield or follow filters. Agent i is considered to have a trajectory that intersects with agent j if

- The paths of agents i and j overlap.
- Agent j arrives at the intersecting area first.
- Agent i arrives at the intersecting area within t seconds after agent j.

In our implementation, we use $t = 2$.

A.4 Non-Interactive

This filter is to identify pairs of agents whose behaviours are likely to be independent of each other. We use the following criterion to consider agent pairs as non-interactive:

- The paths of the two agents do not come within d meters of each other within any t-second window.

In our implementation, we use $d = 20$ and $t = 6$.

References

[1] L. Song, G. Yu, J. Yuan, and Z. Liu, "Human pose estimation and its application to action recognition: A survey," *Journal of Visual Communication and Image Representation*, vol. 76, 2021.

[2] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *ECCV*, 2012.

[3] J. Guan, Y. Yuan, K. M. Kitani, and N. Rhinehart, "Generative hybrid representations for activity forecasting with no-regret learning," in *CVPR*, 2020.

[4] P. Felsen, P. Agrawal, and J. Malik, "What will happen next? forecasting player moves in sports videos," in *ICCV*, 2017.

[5] Q. Sun, X. Huang, J. Gu, B. Williams, and H. Zhao, "M2I: From factored marginal trajectory prediction to interactive prediction," in *CVPR*, 2022.

[6] Y. Ban, X. Li, G. Rosman, I. Gilitschenski, O. Meireles, S. Karaman, and D. Rus, "A deep concept graph network for interaction-aware trajectory prediction," in *ICRA*, 2022.

[7] L. Rowe, M. Ethier, E.-H. Dykhne, and K. Czarnecki, "FJMP: Factorized joint multi-agent motion prediction over learned directed acyclic interaction graphs," in *CVPR*, 2023.

[8] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," 2018.

[9] C. Graber and A. Schwing, "Dynamic neural relational inference," in *CVPR*, 2020.

[10] S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun, "Implicit latent variable model for scene-consistent motion forecasting," in *ECCV*, 2020.

[11] X. Huang, G. Rosman, I. Gilitschenski, A. Jasour, S. G. McGill, J. J. Leonard, and B. C. Williams, "Hyper: Learned hybrid trajectory prediction via factored inference and adaptive sampling," in *ICRA*, 2022.

[12] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms," in *IVS*, 2018.

[13] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, *et al.*, "TNT: Target-driven trajectory prediction," in *CoRL*, 2020.

[14] J. Gu, C. Sun, and H. Zhao, "DenseTNT: End-to-end trajectory prediction from dense goal sets," in *ICCV*, 2021.

[15] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Motion transformer with global intention localization and local movement refinement," in *NeurIPS*, 2022.

[16] S. Shi, L. Jiang, D. Dai, and B. Schiele, "MTR++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[17] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "Precog: Prediction conditioned on goals in visual multi-agent settings," in *ICCV*, 2019.

[18] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, "It is not the journey but the destination: Endpoint conditioned trajectory prediction," in *ECCV*, 2020.

[19] L. Fang, Q. Jiang, J. Shi, and B. Zhou, "TPNet: Trajectory proposal network for motion prediction," in *CVPR*, 2020.

[20] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "MultiPath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," in *CoRL*, 2019.

[21] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "CoverNet: Multimodal behavior prediction using trajectory sets," in *CVPR*, 2020.

[22] B. Varadarajan, A. Hefny, A. Srivastava, *et al.*, "MultiPath++: Efficient information fusion and trajectory aggregation for behavior prediction," in *ICRA*, 2022.

[23] B. Kim, S. H. Park, S. Lee, E. Khoshimjonov, D. Kum, J. Kim, J. S. Kim, and J. W. Choi, "LaPred: Lane-aware prediction of multi-modal future trajectories of dynamic agents," in *CVPR*, 2021.

[24] H. Song, D. Luan, W. Ding, M. Y. Wang, and Q. Chen, "Learning to predict vehicle trajectories with model-based planning," in *CoRL*, 2021.

[25] X. Boyen and D. Koller, "Tractable inference for complex stochastic processes," in *UAI*, 1998.

[26] B. Ng, L. Peshkin, and A. Pfeffer, "Factored particles for scalable monitoring," in *UAI*, 2002.

[27] B. Ng, A. Pfeffer, and R. Dearden, "Factored sampling for efficient tracking of large hybrid systems," *Harvard Computer Science Group Technical Report*, 2005.

[28] M. W. Hofbaur and B. C. Williams, "Mode estimation of probabilistic hybrid systems," in *International Workshop on Hybrid Systems: Computation and Control*, Springer, 2002, pp. 253–266.

[29] M. Hofbaur and B. Williams, "Hybrid estimation of complex systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 5, pp. 2178–2191, 2004.

[30] L. Blackmore, S. Funiak, and B. C. Williams, "A combined stochastic and greedy hybrid estimation capability for concurrent hybrid models with autonomous mode transitions," *Robotics and Autonomous Systems*, vol. 56, no. 2, pp. 105–129, 2008.

[31] B. J. Frey and D. J. C. MacKay, "A revolution: Belief propagation graphs with cycles," in *NIPS*, 1997.

[32] R. McEliece, D. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of pearl's "belief propagation" algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 140–152, 1998.

[33] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *UAI*, 1999.

[34] E. M. Timmons and B. C. Williams, "Best-first enumeration based on bounding conflicts, and its application to large-scale hybrid estimation," *Journal of Artificial Intelligence Research*, 2020.

[35] C. Tang and R. R. Salakhutdinov, "Multiple futures prediction," in *NeurIPS*, 2019.

[36] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *CVPR*, 2016.

[37] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control," in *ECCV*, 2020.

[38] H. Cui, V. Radosavljevic, F. Chou, T. Lin, T. Nguyen, T. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *ICRA*, 2019.

[39] S. Casas, W. Luo, and R. Urtasun, "IntentNet: Learning to predict intention from raw sensor data," in *CoRL*, 2018.

[40] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *CVPR*, 2020.

[41] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *ECCV*, 2020.

[42] J. Ngiam, V. Vasudevan, B. Caine, *et al.*, "Scene Transformer: A unified architecture for predicting future trajectories of multiple agents," in *ICLR*, 2022.

[43] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[44] J. Patrikar, S. Veer, A. Sharma, M. Pavone, and S. Scherer, "RuleFuser: An evidential bayes approach for rule injection in imitation learned planners and predictors for robustness under distribution shifts," *arXiv:2405.11139*, 2024.

[45] S. Veer, A. Sharma, and M. Pavone, "Multi-predictor fusion: Combining learning-based and rule-based trajectory predictors," *arXiv:2307.01408*, 2023.

[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[47] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[48] B. M. Ng, "Factored inference for efficient reasoning of complex dynamic systems," Ph.D. dissertation, Harvard University, 2006.

[49] R. Zhou, H. Zhou, H. Gao, M. Tomizuka, J. Li, and Z. Xu, "Grouptron: Dynamic multi-scale graph convolutional networks for group-aware dense crowd trajectory forecasting," in *ICRA*, 2022.

[50] Y. Chen, C. Liu, X. Mei, B. E. Shi, and M. Liu, "HGCN-GJS: Hierarchical graph convolutional network with groupwise joint sampling for trajectory prediction," in *IROS*, 2022.

[51] J. Li, C. Hua, J. Park, H. Ma, V. Dax, and M. J. Kochenderfer, "EvolveHypergraph: Group-aware dynamic relational reasoning for trajectory prediction," *arXiv:2208.05470*, 2022.

[52] C. Xu, M. Li, Z. Ni, Y. Zhang, and S. Chen, "GroupNet: Multiscale hypergraph neural networks for trajectory prediction with relational reasoning," in *CVPR*, 2022.

[53] E. Tolstaya, R. Mahjourian, C. Downey, B. Vadarajan, B. Sapp, and D. Anguelov, "Identifying driver interactions via conditional behavior prediction," in *ICRA*, 2021.

[54] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *ICLR*, 2017.

[55] X. Zheng, B. Aragam, P. Ravikumar, and E. P. Xing, "DAGs with NO TEARS: Continuous optimization for structure learning," in *NeurIPS*, 2018.

[56] I. Ng, A. Ghassami, and K. Zhang, "On the role of sparsity and DAG constraints for learning linear dags," in *NeurIPS*, 2020.

[57] Y. Yu, J. Chen, T. Gao, and M. Yu, "DAG-GNN: Dag structure learning with graph neural networks," in *ICML*, 2019.

[58] L. A. Thiede and P. P. Brahma, "Analyzing the variety loss in the context of probabilistic trajectory prediction," in *ICCV*, 2019.

[59] S. Ettinger, S. Cheng, B. Caine, *et al.*, "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," in *ICCV*, 2021.

[60] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *CVPR*, 2018.

[61] X. Huang, S. G. McGill, J. A. DeCastro, L. Fletcher, J. J. Leonard, B. C. Williams, and G. Rosman, "DiversityGAN: Diversity-aware vehicle motion prediction via latent semantic sampling," *IEEE RA-L*, vol. 5, no. 4, pp. 5089–5096, 2020.

[62] A. Wang, X. Huang, A. Jasour, and B. C. Williams, "Fast risk assessment for autonomous vehicles using learned models of agent futures," in *Robotics: Science and Systems*, 2020.

[63] Y. Chen, B. Ivanovic, and M. Pavone, "ScePT: Scene-consistent, policy-based trajectory predictions for planning," in *CVPR*, 2022.

[64] W. Luo, C. Park, A. Cornman, B. Sapp, and D. Anguelov, "JFP: Joint future prediction with interactive multi-agent modeling for autonomous driving," in *CoRL*, 2022.

[65] S. Khandelwal, W. Qi, J. Singh, A. Hartnett, and D. Ramanan, “What-if motion prediction for autonomous driving,” *arXiv:2008.10587*, 2020.

[66] S. Casas, C. Gulino, S. Suo, and R. Urtasun, “The importance of prior knowledge in precise multimodal prediction,” in *IROS*, 2020.

[67] J. Sun, C. Yuan, S. Sun, S. Wang, Y. Han, S. Ma, Z. Huang, A. Wong, K. P. Tee, and M. H. A. Jr, “ControlMTR: Control-guided motion transformer with scene-compliant intention points for feasible motion prediction,” in *ITSC*, 2024.