Karlsruhe Institute of Technology

Prior Knowledge Integration for Feasible and Interpretable Trajectory Prediction

Bachelor Thesis

Marius Baden

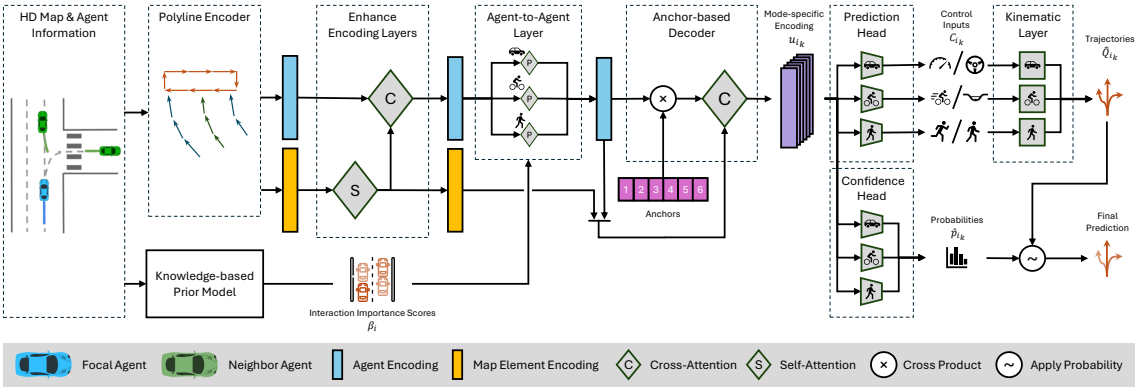
Department of Computer Science
KASTEL Institute of Information Security and Dependability
and
FZI Research Center for Information Technology

Reviewer:	Prof. Dr. R. Reussner
Second reviewer:	Prof. Dr.–Ing. J. M. Zöllner
Advisor:	M.Sc. Ahmed Abouelazm

Research Period: 01. August 2024 – 01. November 2024

Prior Knowledge Integration for Feasible and Interpretable Trajectory Prediction

by
Marius Baden



Bachelor Thesis
November 2024

FZI

Bachelor Thesis, FZI
Department of Computer Science, 2024
Reviewers: Prof. Dr. R. Reussner, Prof. Dr.–Ing. J. M. Zöllner

Abteilung Technisch Kognitive Assistenzsysteme
FZI Research Center for Information Technology

Affirmation

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, *Marius Baden*
November 2024

Abstract

Trajectory prediction is crucial for autonomous driving to navigate safely by anticipating the movements of surrounding road users. However, current deep learning models often lack alignment with human reasoning, leading to predictions that are physically infeasible or are not logical to a human. To address this challenge, recent research has incorporated prior expert knowledge, like the social force model for modeling interactions and kinematic models for physical realism. However, these approaches do not evaluate the impact of the added expert knowledge on the network's interpretability and mainly focus on priors that either suit vehicles or pedestrians but do not generalize to all agent types. In contrast, we propose incorporating priors for interactions and kinematics of all agent classes – specifically vehicles, pedestrians, and cyclists – with class-specific layers to capture the differences in the rationale for each class. To improve interaction interpretability, we guide an agent-to-agent attention layer with rule-based interaction importance scores. We compare two prior models for calculating these scores and two integration methods for guiding attention. To ensure kinematic feasibility, we let the network predict control inputs, from which kinematic models then compute trajectories. We benchmark our proposed modifications on the Argoverse 2 Motion Forecasting dataset, using the state-of-the-art deep neural network HPTR as our baseline. Our experiments demonstrate that we improve interpretability without degrading prediction accuracy, finding a correlation between incorrect predictions and a divergence of the model's attention from our interaction prior. Even though incorporating the kinematic models causes a slight decrease in accuracy, they more than compensate by eliminating infeasible motions present in both the dataset and particularly in the baseline model's predictions, where up to 88% of predictions are impossible in reality. Thus, our approach ensures that predicted trajectories are both interpretable and kinematically feasible, contributing to safer and more reliable trajectory prediction.

Contents

1 Introduction

Road injuries remain a significant public health concern, with the World Health Organization (WHO) reporting them as the leading cause of death among individuals aged 5 to 29 in 2019 and accounting for 1.2 million fatalities globally in 2021 [64]. Autonomous vehicles have the potential to drastically reduce these numbers by making traffic safer and more efficient. A critical challenge in achieving this potential lies in scene understanding, which includes predicting the future trajectories of surrounding road users. This task, referred to as *trajectory prediction* in the literature, is essential for planning safe and collision-free paths. It mirrors the intuitive predictions human drivers make based on their complex understanding of other road users' behaviors, thus enabling humans to drive safely in others' presence. An example of a trajectory prediction task is illustrated in Figure 1.1.

1.1 Motivation: Missing Alignment

Given that road users are human, ensuring accurate trajectory predictions is a safety-critical challenge. For example, incorrect predictions about a pedestrian's movement on a crosswalk can quickly endanger lives. While it may seem like maintaining a large safety distance at all times could solve this issue, this approach is impractical, especially in congested traffic or along sidewalks where space is limited [51]. Autonomous vehicles must navigate these environments effectively while ensuring safety, which requires sophisticated trajectory prediction, akin to the predictions made by human drivers.

Early research applied knowledge-based methods to predict trajectories in traffic scenarios [32]. These methods define simple rules and consider the physics of motion. As a result, they are fast to compute and often guarantee physically feasible predictions. Moreover, the reasoning behind the model's predictions is explainable [32]. However, knowledge-based methods struggle when road users start to interact [30] as it is challenging for them to capture the individual and non-deterministic interaction behavior in fixed rules.

Instead, researchers have shifted their focus towards deep learning methods, particularly neural networks, which largely outperform knowledge-based methods and set the state-of-the-art on modern trajectory prediction benchmarks [54, 26, 37, 24]. Neural networks address the limitations of knowledge-based methods as they are highly flexible and can thus capture complex interaction patterns from large datasets [30].

Nevertheless, the data-driven nature of deep learning methods also presents challenges. A significant issue is that the predictions of these models are not always aligned with human predictions. By being *aligned*, we mean that a human driver presented with the same traffic scene could come up with the same prediction as the model, i.e., predict the same future trajectories

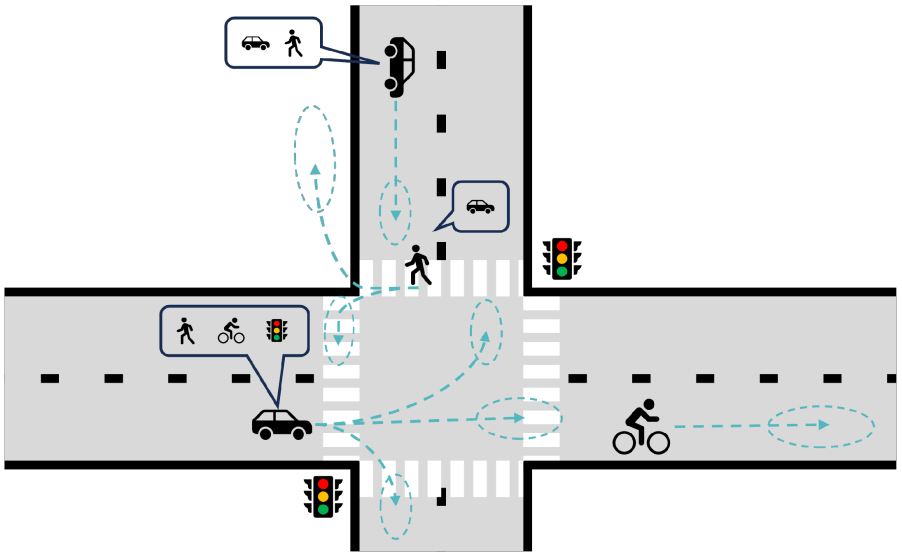


Figure 1.1: Trajectory prediction in a traffic scene with multiple predictions per agent. The arrows illustrate the potential future trajectories of the agents in the scene. The ellipses represent a corresponding probability distribution of the agent's goal position. Adapted from [51].

for the road users. In other words, we want to be able to trust our models. Therefore, it does not matter whether the prediction is a few percent more accurate on average; it does matter whether the prediction is completely wrong or unexpected in some cases. However, state-of-the-art neural networks do not satisfy this requirement. Because these networks are purely numerical, their learning objectives can be defined by loss functions, but the features they learn from the data cannot be controlled or easily understood without additional measures [57]. This can lead to predictions that, while accurate on average, may fail momentously in specific cases. Recent work demonstrated that even the latest neural networks are prone to making obviously unaligned predictions, like predicting a car to drive off-road for no apparent reason [5, 47]. Consequently, recent survey papers suggest that while state-of-the-art deep learning methods are sufficiently accurate to model human behavior, they must become more aligned with human prediction to be effective and safe for autonomous driving [57, 62, 13].

1.2 Problem: Feasible and Interpretable Trajectory Prediction

To improve the alignment of deep learning trajectory prediction, we propose focusing on the two closely related concepts of feasibility and interpretability. We define a trajectory prediction as *feasible* if it adheres to the constraints that reality imposes on trajectories [56]. For example, the physics of motion, referred to as *kinematics*, is one such constraint. We define a trajectory prediction as *interpretable* if a human "can understand the cause of the decision" [41] to predict a certain trajectory instead of any other. In other words, for an interpretable prediction, humans can comprehend why the model makes this prediction.

Both feasibility and interpretability contribute to improved alignment. Firstly, any prediction that is infeasible would usually not be predicted by a human. For instance, consider a kinemat-

ically infeasible prediction where a car traveling at 100 km/h suddenly reverses for a second before continuing forward. Such a prediction would require impossible accelerations and decelerations, making it misaligned.

Secondly, interpretability allows researchers to understand the reasoning behind a prediction. If the reasoning mirrors human reasoning, the prediction is more likely to be aligned with human expectations. Hence, interpretability helps researchers evaluate the alignment of their models' predictions.

Previous research has attempted to improve feasibility and interpretability by integrating knowledge of traffic and driving into deep learning methods [13]. The literature refers to this knowledge as *expert knowledge*, *prior knowledge*, or simply a *prior* [13, 58, 65]. As the alignment problem demonstrates, deep learning methods are powerful tools but lack theoretical guarantees [21]. In contrast, knowledge of driving from robotics is physically precise. Consequently, researchers introduced hybrid approaches that fuse deep learning with expert knowledge [13].

The knowledge researchers integrated into their models is diverse. In pedestrian trajectory prediction, a popular prior is the Social Force Model [27], a rule-based model for simulating pedestrian trajectories. Researchers have integrated the forces from this model as a prior to determine which agents to consider and how they interact [28, 68, 35]. These studies demonstrate that a Social Force Model prior improves both the accuracy and interpretability of predictions.

However, they miss the opportunity to evaluate how the prior improved their prediction. This is partly because they integrate the prior in a way that makes the prior's influence on the prediction uninterpretable. Furthermore, these methods have so far focused solely on pedestrian trajectory prediction because the Social Force Model is a model for pedestrians. In response, this thesis aims to integrate an *interaction prior* for interactions among both pedestrians and vehicles. The prior shall allow us to interpret its impact on the prediction and, therefore, monitor the alignment with human prediction, which is represented by the prior.

In vehicle trajectory prediction, kinematic models of vehicles were the first idea for prior knowledge to incorporate. A kinematic model describes how control inputs, like acceleration and steering for a car, affect the agent's motion, like velocity and change in position. By embedding these models into deep learning predictors, researchers have ensured kinematic feasibility in their predictions [11, 21, 48]. As a certain type of feasible, a prediction is *kinematically feasible* if it adheres to the kinematic motion constraints of the agent for which the trajectory is predicted [21]. We refer to the agent for which the trajectory is predicted as the *focal agent* [55].

While these methods enhance the realism and hence the alignment of vehicle trajectory predictions, they often overlook suitable kinematic models for pedestrians. In contrast, this thesis aims to find and integrate appropriate *kinematic priors* in the form of kinematic models for both vehicles and pedestrians to guarantee kinematic feasibility across different agent classes.

Thus, this thesis addresses the following research question: *Can integrating prior knowledge improve the alignment of trajectory predictions with human expectations by enhancing interpretability and feasibility through the use of interaction priors and kinematic models*?

The rest of the thesis is structured as follows: After this introduction, Chapter 2 presents a detailed review of the state-of-the-art methods for trajectory prediction, with a focus on deep

learning approaches and prior knowledge integration. Chapter 3 outlines the methodology proposed in this thesis, including the problem definition, model architecture, and the integration of expert priors. The metrics and dataset for evaluating the effectiveness of the proposed methods as well as the parameter setup of our model are described in Chapter 4. In Chapter 5, we present and discuss the experiments conducted to assess the performance of our model in terms of accuracy, kinematic feasibility, and interpretability. Finally, Chapter 6 concludes this thesis by highlighting our findings and discussing potential avenues for future work.

2 State of the Art

Before we dive into the details of our method, in this section, we review the previous work in trajectory prediction. First, we give a short overview of how the state-of-the-art has developed in recent years. To begin, we briefly introduce knowledge-based prediction approaches before advancing to deep learning approaches. Second, we review why and how previous research integrated prior knowledge into trajectory prediction. Here, we distinguish between prior knowledge of interactions and prior knowledge of kinematics.

2.1 Overview of Previous Work in Trajectory Prediction

We begin by discussing the state-of-the-art's development, starting with early knowledge-based approaches that do not require deep learning. Next, we move from the first deep learning approaches via the milestones in deep learning architecture to the current state-of-the-art.

2.1.1 Knowledge-based Approaches

Particularly in pedestrian trajectory prediction, research on agent trajectories started with observing agents to gain knowledge about agent dynamics and behavior [32]. To make trajectory predictions, researchers then created simulation models from this knowledge based on deterministic rules. We introduce two such models relevant to this thesis: the simple Constant Velocity Model (CVM) and the physics-based Social Force Model (SFM).

Constant Velocity Model (CVM)

The Constant Velocity Model is one of the simplest approaches for trajectory prediction [46]. It assumes that the agent continues traveling in a straight line at a constant velocity, measured from the agent's past trajectory. Typically, research papers use the velocity derived from the distance traveled in the last time step.

The CVM offers several advantages. Its simplicity makes it easy to implement and very fast to compute compared to more complex trajectory prediction methods. Additionally, the CVM can serve as a reasonable baseline for trajectory prediction, particularly given the high data imbalance in most trajectory datasets. Trajectories, where agents maintain a constant velocity or follow a straight path, are often the most common. Notably, Scholler et al. [50] demonstrated that the CVM can even outperform recent deep learning methods on datasets like ETH/UCY [43, 34] when no additional context information beyond past trajectories is provided.

However, the CVM is significantly limited by its simplicity, primarily due to its lack of context consideration [46]. It does not take into account contextual information such as map details or

the presence of other agents. For example, in vehicle trajectory prediction, the CVM cannot follow curved lanes or turns.

Social Force Model (SFM)

Knowledge-based prediction methods can rely on a lot more knowledge than the CVM's simple assumption of moving straight ahead. For example, they can include context information and thus tackle the CVM's main drawback.

A popular knowledge-based prediction method that incorporates context information is the Social Force Model [27]. The SFM is an acceleration-based prediction approach popular in pedestrian simulation and trajectory prediction. The core idea is that an agent's motion results from the sum of forces influencing the individual. Each pedestrian agent is subject to attractive forces, which draw them toward their goal or a group they are walking with, and repulsive forces, which push them away from obstacles and other agents. This concept is inspired by the way particles move due to physical forces.

For SFM-based trajectory prediction, the social forces are virtual. This means that it does not model physical forces acting on the agents but rather the forces perceived in human reasoning as a result of the traffic environment, such as the presence of other agents and road boundaries [62]. For instance, when a driver approaches an intersection to make an unprotected left turn and sees a fast oncoming vehicle, they may wait to avoid a collision. Although no physical force is applied by the oncoming vehicle, the driver reacts as if such a force exists. Modeling these virtual forces, which intuitively explain social interactions, is the core idea of the social force model.

The SFM is mathematically defined as the sum of three social forces [27]: the driving force F_{drive}, which motivates the agent to move toward a goal with a desired velocity; the repulsive force F_{rep}, which prevents collisions with other agents and obstacles, maintaining a safe distance; and the attractive force F_{att}, which draws agents toward other agents of interest (AOI), for example, a friend they are walking with. For a focal agent i, the total force F_{tot} can be expressed as follows [27]:

$$F_{tot_i} = F_{drive_i} + \sum_{j \in neighbors_i \cup obstacles_i} F_{rep_{ij}} + \sum_{j \in AOI_i} F_{att_{ij}}. \tag{2.1}$$

The repulsive force F_{rep} from other agents is the most commonly used force in advanced works [68, 28, 3, 35, 20]. For a fixed focal agent i, each neighbor j has its own repulsive potential V_{sfm}, which has equipotential lines forming an ellipse pointing in j's direction of motion [27]:

$$V_{sfm}(r_i, r_j, v_j) = V^0_{sfm} \cdot e^{-\frac{1}{\tau_{sfm}} b_{sfm}(r_i, r_j, v_j)}, \tag{2.2}$$

where $r_i = [x_i, y_i]^T \in \mathbb{R}^2$ is the 2D Cartesian position of i, $v_i = [v_{x_i}, v_{y_i}]^T \in \mathbb{R}^2$ is the 2D velocity of i, while $V^0_{sfm} \in \mathbb{R}$ and $\tau_{sfm} \in \mathbb{R}$ are constant factors for calibrating the SFM. $b_{sfm}(r_i, r_j, v_j)$

represents the semi-minor axis of the potential field's ellipse. It is defined as follows [27]:

$$2 \cdot b_{sfm}(r_i, r_j, v_j) = \sqrt{(||d_{ij}|| + ||d_{ij} + v_j \Delta t||)^2 - (||v_j|| \Delta t)^2}, \quad [2.3]$$

where $d_{ij} = r_j - r_i \in \mathbb{R}^2$ is the difference in position of i and j while Δt is the duration of one simulation step.

The force acting on agent i is the derivative of the potential function [27]:

$$f_{rep_{ij}} = -\nabla_{d_{ij}} V_{sfm}(b_{sfm}(r_i, r_j, v_j)). \quad [2.4]$$

To obtain $F_{rep_{ij}}$ from $f_{rep_{ij}}$, additional weighting is applied based on which agents are in i's field of view.

Numerous extensions to the SFM, such as the Headed Social Force Model or the Social Vehicle Force Model, aim to enhance the SFM for specific applications like simulating pedestrian flow or traffic in shared open spaces, respectively.

The primary benefit of the SFM over the CVM is its ability to incorporate the focal agent's surroundings, especially interactions with other agents, while still ensuring physical feasibility. This results in realistic simulations that, to some extent, align with real-world scenarios. However, the accuracy of the SFM highly depends on the calibration of the model's parameters. The ideal calibration varies significantly between datasets and can even differ from scenario to scenario [28]. For example, parameters like the desired speed per agent are not clear without any other form of prediction.

Discussion of Knowledge-based Approaches

Besides the SFM and CVM, there are more advanced knowledge-based methods such as velocity-based models like Optimal Reciprocal Collision Avoidance (ORCA) [6] and decision-based models with cellular automata [32]. Yet, they mostly share the following properties with the discussed approaches.

Firstly, knowledge-based methods offer several advantages. They provide deterministic predictions, ensuring physically feasible outputs that are understandable to humans. This deterministic nature gives us clear and interpretable reasoning behind each prediction, which is an essential benefit for safety-critical applications like autonomous driving. Moreover, due to their rule-based nature, these methods generalize well across various scenarios and datasets, making them robust and reliable in diverse conditions.

However, the deterministic nature of knowledge-based methods also imposes limitations. They are often too rigid to capture the complex and individually different behaviors observed in real-world scenarios, leading to less accurate predictions [22]. Furthermore, knowledge-based methods predict a single future trajectory, while in reality, multiple future trajectories might be highly likely [51].

2.1.2 Deep Learning Approaches

Unlike knowledge-based methods that rely on hand-crafted functions, deep learning approaches, particularly neural networks, are data-driven. They can learn patterns directly from data without the need for rule-based models. Due to their superior accuracy, deep learning methods have garnered significant attention in trajectory prediction in recent years [32].

In this section, we first introduce the use of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) in trajectory prediction. Then, we highlight the advancements of graph-based approaches over CNNs and the advancements of Transformers on graphs over RNNs. Lastly, we discuss the state of the art in deep learning approaches. For each approach, we introduce the concept and give a brief overview of its advantages and disadvantages.

Convolutional and Recurrent Neural Networks

Early work in deep learning trajectory prediction employed Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) as standalone models to predict trajectories. However, these models are now primarily used as building blocks within more complex architectures to encode past trajectories and environmental information.

CNNs make predictions at the image level [51]. They process scene images with convolutional layers to extract agent positions and relevant scene information. CNNs are often used as components in advanced architectures. For instance, models like MTP [12], Trajectron++ [48], and HoliGraph with map autoencoder [25] utilize CNNs to extract local map information for enhancing their trajectory prediction capabilities with detailed environmental context.

RNNs, on the other hand, make predictions at the coordinate level [51]. They process an agent's past trajectory as a sequence of positions over time, learning to predict future positions based on this sequence. A notable milestone in pedestrian trajectory prediction is the Social LSTM model [1]. This model uses one RNN instance per agent in the scene to learn the agent's state and predict their future positions. A key innovation of the Social LSTM is the Social Pooling Layer, which shares the hidden states between neighboring RNNs by pooling the hidden states of RNNs within a certain radius. This allows the model to explicitly capture interactions between pedestrians, yielding more accurate and socially-aware trajectory predictions.

In summary, CNNs and RNNs significantly improved accuracy over traditional knowledge-based approaches [13]. Still, both model architectures have notable limitations. RNNs face the exploding and vanishing gradient problem, which complicates the training process and can affect model performance. CNNs often process a substantial portion of the image that contains irrelevant information because most pixels in the image have no meaning for the prediction. This reduces CNNs' computational efficiency and increases the risk of overfitting predictions on irrelevant details of the scene image. Furthermore, the convolution operation inherent in CNNs focuses on local regions, making it challenging to account for distant agents approaching the focal agent quickly.

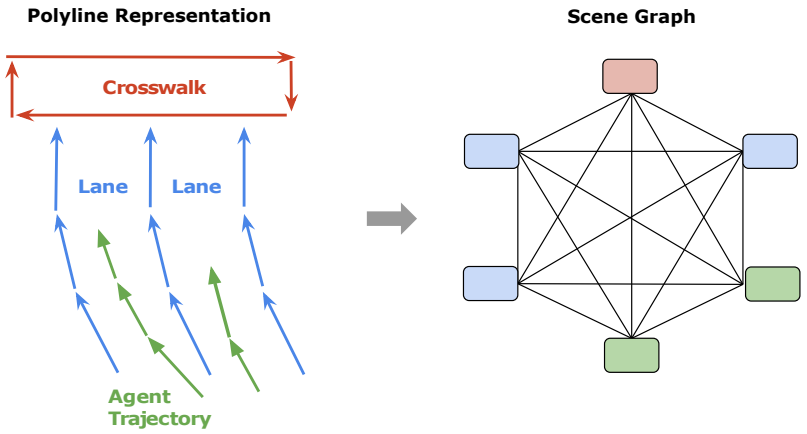


Figure 2.1: VectorNet [19] polyline scene representation and resulting scene graph with two agents, one crosswalk, and three lane boundaries. Adapted from [19].

Graph-based Approaches

Advancing towards more sophisticated neural network architectures, the challenge of representing agents in a scene becomes crucial [51]. Research has focused on graph-based approaches due to their superior scene representation capabilities compared to CNNs. In this section, we highlight where these scene representation capabilities come from.

Most graph-based approaches begin with pre-processing to detect scene objects and determine their relevant attributes, such as position, velocity, and object type [13]. This information is then compiled into a scene graph, where each agent or object corresponds to a node. Each node's attributes are encoded as hidden embeddings, which may include past trajectories, relative velocity, yaw rate, distance, relative positions, agent class type, and environmental information like road boundaries and traffic signs [51]. The edges in the scene graph model interactions between nodes. Based on the scene graph, multiple neural networks can be applied to process the graph's information into a trajectory prediction. For example, Graph Neural Networks (GNNs), RNNs, and Multi-Layer Perceptrons (MLPs) are commonly used [51].

A significant advantage of scene graphs is their ability to treat each scene object differently after detection [51]. This allows using different encoders for various elements, such as vehicles and lanes. This is a notable improvement over CNNs that process all pixels uniformly. Moreover, scene graphs only include relevant objects as nodes, addressing the issue of processing irrelevant pixels.

Two prominent graph-based approaches are VectorNet [19] and LaneGCN [37]. They represent different methods of integrating an agent's state over time into a hidden embedding in the scene graph. VectorNet models every agent as a sequence of lines connecting each time step's agent position. Similarly, static scene objects like crosswalks are modeled as a sequence of lines that mark their boundary. These sequences of lines, called polylines, each represent exactly one agent or object in the scene as illustrated in Figure 2.1. VectorNet then applies GNNs to each polyline to obtain one node embedding per polyline.

LaneGCN [37], on the other hand, treats agents and static objects differently. Agents are represented as sequences of positions over time. LaneGCN applies 1D CNNs to each agent

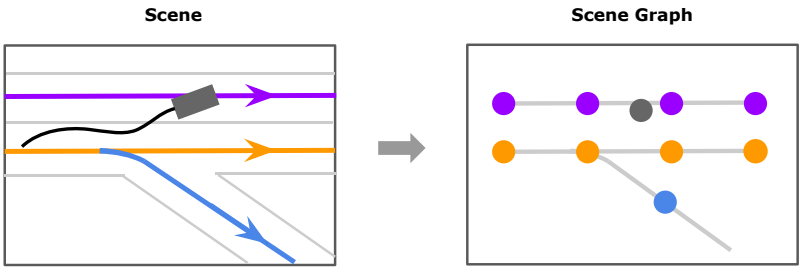


Figure 2.2: LaneGCN [37] scene graph representation with one agent (grey) and three lanes (orange, blue, purple). Adapted from [37].

sequence to get one embedding per agent. Lanes, the only static scene objects considered by LaneGCN, are represented as sequences of points along the centerlines of lanes, as shown in Figure 2.2. Note that this is different from VectorNet [19], which uses the lane boundaries instead of centerlines. LaneGCN also does not merge lanes into a single embedding per lane, as VectorNet does. Instead, each centerline point is added to the scene graph as a separate node.

Overall, graph-based approaches mark a significant advancement in trajectory prediction. They offer flexibility in encoding node features differently for various nodes, ensure that every node in the graph is relevant, and explicitly model interactions, allowing the model to learn how neighboring agents' features influence the focal agent's prediction. These capabilities lead to improved accuracy compared to previous methods. However, graph-based approaches can be relatively slow to compute because applying the commonly used GNNs requires iterating over graph connections, which can be computationally intensive.

Transformers

Transformers have demonstrated exceptional performance in various fields, particularly in Natural Language Processing (NLP) [60]. Their ability to model sequences with high accuracy makes them a natural fit for trajectory prediction, which is inherently a sequence-based problem. Thus, it is no surprise that Transformers also succeed in trajectory prediction. Transformers applied on top of graph-based approaches are leading the rankings in state-of-the-art benchmarks [54, 42, 15, 63]. Transformers consist of multiple layers of the attention mechanism followed by normalization and an MLP. Essentially, they improve over previous graph-based approaches by enhancing the prediction process based on the scene graph. The prediction, previously done by MLPs, CNNs, or RNNs, is now performed by Transformers.

We introduce three leading Transformer-based models that exemplify the advancements in trajectory prediction relevant to our paper: Wayformer [42], MTR [54], and HPTR [69]. Wayformer is a notable example due to its simplicity and high performance. Its architecture is depicted in Figure 2.3. It first concatenates all input information about agents and the map [42]. A Transformer encoder takes the concatenation to generate a scene embedding for each focal agent. This scene embedding is then passed to a Transformer decoder that predicts multiple possible future trajectories for the focal agent. Each predicted trajectory, a so-called *mode*, is to be understood as a possible future for the focal agent. Only one of the modes can become true.

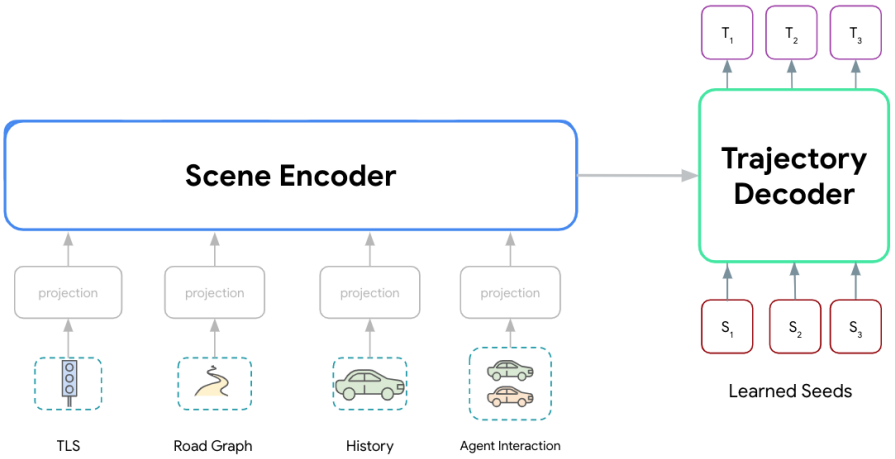


Figure 2.3: Wayformer's [42] simple Transformer architecture. Both Scene Encoder and Trajectory Decoder are Transformer networks. T_1, T_2, T_3 are three predicted modes. S_1, S_2, S_3 are the seeds corresponding to one mode. Traffic light state (TLS), road graph, agent history, and neighboring agents for agent interactions are the information that is concatenated to form the Scene Encoder's input. Adapted from [42].

The goal behind predicting multiple modes is to improve the prediction in situations where it is unclear which of a handful of possible behaviors the focal agent will choose. For example, an agent could turn right, continue straight, or turn left at a junction. For each mode, Wayformer learns one seed for predicting the mode based on the scene embedding. Despite its effectiveness, Wayformer requires a large model, comprising 60 million trainable parameters, to perform well [17].

MTR [54], or Motion Transformer, adopts a more complex architecture than Wayformer but can in turn achieve similar performance with only 16.5 million parameters [17]. Like Wayformer, MTR uses a Transformer to encode all agents and map information into a scene embedding [54]. However, it introduces an additional step: it employs an MLP to predict the future intentions of where each agent wants to go. It then encodes these intentions into the scene embedding as additional information for the prediction. As in Wayformer, a Transformer decoder then uses this scene embedding to predict multiple modes. However, compared to Wayformer, MTR introduces more detailed modeling to the Transformer decoder. For example, it incorporates static and dynamic motion queries in the decoder's attention to improve its ability to predict suitable modes.

HPTR (Hierarchical Predictive Transformer) [69] aims to maintain the performance of Wayformer and MTR while reducing inference time and improving scalability, making the model more suitable for real-time operation in autonomous vehicles. Similar to MTR, it has 15.2 million parameters [69]. HPTR uses separate Transformer encoders for each type of scene information: a map object encoder, a traffic light encoder, and an agent encoder. Additionally, it employs a Transformer decoder to capture the interactions between agents and map elements by applying cross-attention between the two. Here, they use k-nearest neighbor attention instead of regular attention. That means they compute the cross-attention only between elements that are spatially close to each other in the scene. This ensures a sense of locality for interaction.

Like Wayformer, HPTR uses a Transformer decoder to predict multiple modes from the scene embedding.

The advantages of utilizing Transformers in trajectory prediction are numerous. Firstly, they possess strong sequence modeling capabilities: they outperform RNNs by solving issues such as the exploding or vanishing gradient problem [51]. They also process the entire input sequence in parallel, which makes their inference considerably faster than that of RNNs. Secondly, Transformers are a well-understood architecture that has demonstrated superior performance across various fields. Overall, this has led to Transformers being the most accurate trajectory prediction models in the Argoverse2 [63] and WOMD [15] benchmarks.

Discussion of Deep Learning Approaches

On balance, current state-of-the-art deep learning trajectory prediction methods offer several key advantages. They achieve high accuracy, can capture multiple possible modes with their respective probabilities, and effectively model how neighboring agents influence the focal agent's predictions.

However, significant challenges remain unaddressed in these models. One critical drawback is their poor generalizability. While these models perform well on the datasets they are trained on, they often struggle to adapt to new or adversarial scenarios. For instance, a recent study revealed that even well-regarded models like LaneGCN [37] can make implausible off-road predictions when the scene's map is altered in adversarial but realistic ways [5]. This limitation raises serious concerns about their reliability in real-world applications, where models must handle scenarios not represented in the training data.

Most concerning is the potential for unaligned predictions, as defined in Section 1.1, which undermines trust in these models, particularly in safety-critical applications. While state-of-the-art models have demonstrated outstanding performance on benchmarks, indicating that their predictions are generally accurate, the accuracy metric does not make any statement on instances where predictions are erroneous. In these rare but significant cases, the models can produce physically impossible or inaccurate predictions with safety-critical consequences [47]. The worst part of this problem is that, due to the black-box nature of deep learning, the models provide no indication of whether a given prediction is reliable or flawed. The issue arises because state-of-the-art models make their predictions regression-based. The model aims to predict an ordered set of points that are, on average, as close as possible to the training data's ground truth trajectories. That means the predicted points do not necessarily have to be spatially close to each other. For example, there is no reason why the prediction cannot be a straight line for the first three points while the fourth point is placed on the other end of the map. Similarly, the prediction is not bound to physics. For instance, the prediction can suggest a pedestrian will cross a crosswalk at 300 km/h. This unpredictability poses a serious risk in autonomous driving, where a single erroneous prediction could lead to life-threatening decisions. In the crosswalk example, the planning software might see no reason to decelerate before the crosswalk because, based on the 300 km/h prediction, the pedestrian will not be in the way. Thus, despite the impres-

sive accuracy of deep learning predictions, their lack of feasibility and interpretability remains a significant barrier to their safe deployment in autonomous systems.

2.2 Aligned Trajectory Prediction

To address the lack of feasibility and interpretability in deep learning approaches discussed in Section 2.1.2, some researchers have explored hybrid models that combine deep learning with knowledge-based approaches [13]. These hybrid models aim to leverage the strengths of both paradigms. They attempt to integrate the flexibility and accuracy of deep learning with the physically grounded and interpretable nature of knowledge-based methods from Section 2.1.1.

Works on hybrid models often aim to improve predictions in other quality metrics than accuracy [22]. For example, they argue it is worth sacrificing some accuracy for significantly better physical feasibility [21].

This section provides an overview of how prior knowledge has been integrated into trajectory prediction models in previous research. Motivated by Section 1.1, we focus on two main categories: *interaction priors* to improve interpretability and *kinematic priors* to ensure feasibility. Based on that, we highlight the gaps in existing research that our work intends to address.

2.2.1 Interaction Priors

One effective approach to enhance the alignment of predictions is by using priors for social interactions to improve the interpretability of the model's outputs. This section presents how interactions are modeled in the state-of-the-art, what priors can be integrated, and how these priors can be integrated. Finally, it highlights the gap in current works on these problems.

Interaction Modeling

First, we present how interactions are modeled in state-of-the-art neural networks for trajectory prediction before discussing how we can introduce interaction priors into them. In modern graph-based and transformer-based deep learning models, social interactions are typically modeled in the following pipeline of three steps [62]. First, an encoder computes an embedding for each agent, capturing relevant information from both the agent's history and its interactions with the environment. Next, these embeddings are fused using an attention mechanism that determines the importance of each neighboring agent to the focal agent's future trajectory. Finally, the fused embeddings, now enriched with interaction information, are used as input for a decoder that predicts the future trajectory of the focal agent.

The attention mechanism within this framework plays a crucial role by assigning scores that reflect the importance of each agent in the scene. Thus, these attention scores can be interpreted as an *interaction importance score*, indicating how much influence one agent has on another's predicted trajectory. For example, XHGP [8] utilizes this interpretation by visualizing the attention scores to help explain which agents a graph-based neural network considers most influential in shaping the focal agent's predicted path.

Given the central role of attention scores in determining the final prediction, it is logical to introduce human driving knowledge as a prior at this stage. First, because the attention scores are inherently interpretable, they provide a clear and human-understandable basis for designing priors. This allows us to encode domain-specific knowledge directly into the model in a way that humans can easily understand and validate. Second, the attention scores have a significant impact on the prediction because they decide the final composition of the embeddings that are passed to the decoder. These embeddings form the foundation for the model's decoder to make predictions, meaning that the attention scores effectively shape the model's decision-making process. For these two reasons, integrating an interaction prior into the attention mechanism can guide the model toward more interpretable and human-aligned predictions.

Prior Models

This leads us to the question of how to obtain such an interaction prior. As a prior model, we require a deterministic algorithm that yields an interaction importance score for each neighbor of the focal agent based on rules and driving knowledge. This is often realized through hand-crafted functions.

Social Force Model (SFM) One prior model for social interactions well-established in the literature is the SFM [27], which we introduced in Section 2.1.1 as a knowledge-based approach for trajectory prediction. The forces in the SFM can be interpreted as interaction importance scores that reflect the influence of various neighbors and other factors on an agent's trajectory. These scores naturally lend themselves to integration with attention mechanisms in deep learning models. However, while the SFM has been widely used for pedestrian trajectory prediction and integrated into some deep learning models, it has not yet been employed as an interaction prior for attention mechanisms. Instead, it has mainly been used as an additional input or as inspiration for model architectures:

For instance, ForceFormer [68] integrates the SFM for pedestrian trajectory prediction by extending the AgentFormer model with a goal estimation module and a social force module that handles interactions. Similarly, SF-GRU [35] uses the SFM as an input in vehicle trajectory prediction. SFEM-GCN [14] extends its inputs to both vehicles and pedestrians, incorporating knowledge-based inputs inspired by the SFM into a graph convolutional network. However, these models typically justify the inclusion of the SFM by showing improved accuracy without examining how the model utilizes these additional inputs. This limitation arises from the way the SFM is integrated, making it difficult to interpret or monitor the model's decision-making process.

Some approaches have attempted to incorporate the SFM directly into the architecture of a neural network. For example, the models SFM-NN [3] and SFMGNet [28] treat each equation of the SFM as a neural network layer. This approach allows the parameters and weights, typically calibrated manually in the SFM, to be learned during training. While these models perform well on pedestrian trajectory prediction, particularly on synthetic data, their application to vehicle trajectory prediction remains unexplored. The rigid structure of the SFM may limit its flexibility,

making it challenging to extend this approach to more complex, real-world scenarios involving vehicles.

Social Knowledge-Guided Graph Attention Convolutional Network (SKGACN) Another prior model for social interactions is the one used in SKGACN [39]. The SKGACN model generates interaction importance scores that capture the spatial and motion-related factors influencing social interactions among pedestrians. Specifically, the importance of the interaction between two pedestrians, i and j, is calculated based on three key elements: the Euclidean distance between them, their respective speeds, and the visual angles relative to each other. The visual angle γ_{ij} from i to j is the angle between the heading of i and the vector connecting i to j. The unnormalized interaction importance score $\hat{\beta}_{ij}^{\text{skgacn}}$ at time step t is mathematically defined as follows [39]:

$$\hat{\beta}_{ij}^{\text{skgacn}} = \begin{cases} \frac{||v_i|| \cos \gamma_{ij} + ||v_j|| \cos \gamma_{ji}}{||d_{ij}||} & \text{if } \cos \gamma_{ij} > 0, \\ \frac{||v_i|| \cos \gamma_{ij}}{||d_{ij}||} & \text{if } \cos \gamma_{ij} \leq 0. \end{cases} \quad [2.5]$$

Here, $||v_i||$ represents the speed of pedestrian i and $||d_{ij}||$ is the Euclidean distance between pedestrians i and j. Note that γ_{ij} is not symmetric, i.e., in general $\gamma_{ij} \neq \gamma_{ji}$, and hence $\hat{\beta}_{ij}^{\text{skgacn}} \neq \hat{\beta}_{ji}^{\text{skgacn}}$. The final interaction importance score $\beta_{ij}^{\text{skgacn}}$ is obtained by normalizing $\hat{\beta}_{ij}^{\text{skgacn}}$ with the softmax function. This ensures all scores fall within the range $[0, 1]$ and that $\sum_j \beta_{ij}^{\text{skgacn}} = 1$.

SKGACN's neural network then incorporates this interaction importance score into its attention score calculations. Specifically, it multiplies the agent embeddings with the prior scores during the attention mechanism's computation [39]. We discuss the integration process in more detail in the next paragraphs.

Compared to the SFM, the SKGACN model is easier and faster to compute as it does not rely on gradients of potential fields [39]. Additionally, it takes into account the velocity and heading of the focal agent, which the SFM does not. However, a significant limitation of the SKGACN approach is that the score is tailored for pedestrians, leaving its applicability to vehicle trajectory prediction or mixed vehicle and pedestrian trajectory prediction unclear.

Prior Integration as Mixture of Experts

Once an interaction prior is established, the next challenge is integrating it into the prediction process. The process of combining a neural network's prediction with a prior can be considered a mixture of experts, where one expert is the neural network and the other is the knowledge-based prior model. While the mixture of experts approach is relatively new to trajectory prediction, it has been explored more extensively in other fields of deep learning. In the literature, we identified four categories of prior integration methods relevant to our application of interaction importance scores for trajectory prediction.

Contextual Prior One method of integrating priors is to use the prior solely as an input. The prior and the neural network's embedding are combined, for example, through a Multi-Layer Perceptron (MLP). In natural language processing (NLP), this approach has been used

effectively in models like AffectiveAttention [40], which includes lexicon-based features as a prior in the self-attention mechanism of a Transformer. Another example is K-BERT [38], which significantly improves over the BERT model by integrating domain knowledge in this manner. The primary advantage of the contextual prior approach is that it is the most flexible. It allows the model to freely decide how to use the prior. However, this flexibility is also a drawback, as the model is not bound to use the prior and could potentially ignore it during training.

Loss Function Another approach is to integrate the prior by adding a loss term that regularizes the attention scores to align with the prior. For instance, the BAM model [16] in image captioning uses a loss function that encourages the model to focus on areas of the image deemed important by a prior. While this method ensures that the model learns to incorporate the prior in its attention mechanism, it also introduces the complexity of learning the prior alongside the primary task, which can complicate the training process.

Multiply-and-Renormalize A more direct approach is to multiply the prior with the embeddings before computing the attention scores or to element-wise multiply the prior with the attention scores after they have been computed from the embeddings. This method has been used in models like BERT-SIM [65], which integrates a word similarity prior into the attention mechanism of a Transformer. The advantage of the multiply-and-renormalize approach is that it makes the prior's influence on the attention mechanism more explicit, allowing researchers to track how the prior affects the model's decisions. However, this method is less flexible, as the multiplication can effectively mask parts of the neural network's prediction where the prior is close to zero. This potentially reduces the model's ability to learn from the data.

Gating The final method is gating, where the model uses gating values to decide how much weight to assign to the prior versus the neural network's output. This approach is utilized in the Attentive Mixture of Experts (AME) model [53]. It is an attention mechanism for various applications that apply a gating on multiple experts' results. Here, a separate gating network computes the weights for a weighted sum over the experts' results to compute the final result. The gating network takes the experts' inputs and their respective outputs as input to compute the gating values as its output. Similarly, the AffectiveAttention model [40] uses gating values derived from an MLP to re-weight the attention mechanism's input embeddings based on a prior. Overall, the gating method offers more flexibility than the multiply-and-renormalize method while being more explicit than the contextual prior method: it allows the model to dynamically determine the prior's contribution, which can be directly measured by the gating value. However, it also shares the drawback of the contextual prior approach, where the model might learn to ignore the prior unless an additional loss function is introduced to enforce its use.

In the context of trajectory prediction, these four methods have seen limited but growing applications. SRGAT [10], for example, uses MLPs to process expert knowledge before it affects the attention scores, thus employing a contextual prior approach. Consequently, SRGAT does not

need to define a prior in the form of an interaction importance score. Instead, it simply passes knowledge-based features like velocity and relative directions. SKGACN [39], as mentioned earlier, uses a multiply-and-renormalize approach by multiplying the interaction importance score with the neural network's predicted attention score. Finally, the SR-LSTM model [67] for pedestrian trajectory prediction uses a gating mechanism, although it does not integrate a prior. Instead, it uses gating to determine which features of the agent embedding to use for prediction.

Research Gaps

Despite significant advances in integrating interaction priors into trajectory prediction models, several key gaps remain unaddressed.

Firstly, existing research has primarily focused on pedestrian-specific interaction priors, with little attention given to vehicle trajectory prediction. Models like the SFM [27] and its variations, such as those used in ForceFormer [68], SFEM-GCN [14], and SFMGNet [28], have been extensively applied to pedestrian scenarios but not adequately tested or extended to vehicles. This leaves a gap in mixed scenario modeling, where both vehicles and pedestrians interact. To address this, we investigate the design of an interaction prior that works effectively for both vehicles and pedestrians. Additionally, we compare the performance of a pedestrian-specific interaction prior when applied to vehicles and pedestrians versus our proposed general interaction prior.

Another gap lies in the limited exploration of integrating priors into attention mechanisms within trajectory prediction models. While various methods have been developed in other domains, such as natural language processing, few have been applied to trajectory prediction. For instance, methods like SFM-NN [3], and SFMGNet [28] have proven effective in pedestrian scenarios but lack the generality needed for mixed scenarios involving vehicles and pedestrians. Moreover, these integration methods often do not provide a clear interpretation of how the prior influences model predictions, which is crucial for building trust in safety-critical applications like autonomous driving. To overcome these limitations, we explore and evaluate different integration methods, such as those used in Attentive Mixture of Experts [53], BERT-SIM [65], and SKGACN [39], focusing on their effectiveness and interpretability in mixed scenarios.

Finally, there is a lack of rigorous evaluation of how integrated priors impact prediction alignment with human expectations. Models such as ForceFormer [68], SF-GRU [35], and SFEM-GCN [14] often incorporate priors without systematically assessing their influence on prediction alignment or investigating whether deviations from the prior correlate with erroneous predictions. This gap hinders our understanding of the reliability and safety of these models. To tackle this gap, we analyze how the addition of an interaction prior affects predictions: whether it merely serves as another input or actively improves alignment with human-like predictions.

2.2.2 Kinematic Priors

Another approach to enhance the alignment of predictions with human-like predictions is by enforcing kinematic feasibility through priors on the focal agents' kinematics. As introduced in

Section 1.2, kinematic feasibility ensures that the predicted trajectories adhere to the physical constraints of motion, making them more realistic and reliable. While other forms of feasibility, such as map feasibility and social feasibility, have been explored and are desirable [5, 47], they are not a priority of this thesis. To improve alignment, our goal is to first establish kinematic feasibility because kinematically infeasible predictions conflict the most with human reasoning compared to map-infeasible or socially infeasible predictions.

This section focuses on kinematic feasibility. We introduce four approaches for achieving kinematic feasibility, discuss the form of prior kinematic knowledge they require, and highlight the gaps in current research on kinematically feasible predictions.

Kinematically Feasible Prediction

Several recent works have achieved trajectory prediction with guaranteed kinematic feasibility. We identified four categories of approaches among the most relevant works for vehicle and pedestrian trajectory prediction.

Post Processing One approach to ensuring kinematic feasibility is post-processing the predictions. This method involves running checks on the predicted trajectories to determine whether they are kinematically feasible. If a prediction fails a check, a fallback mechanism is used to adjust the trajectory. For instance, SafetyNet [61] employs post-processing detection and a handcrafted fallback to ensure both kinematic and map feasibility.

The advantage of post-processing is that it does not require any changes to the neural network, making it easy to implement as a separate module after the network's prediction [61]. However, a significant drawback is that when a trajectory is deemed infeasible, it is unclear how to generate an accurate alternative prediction, which can limit the effectiveness of this method.

Rule-based Trajectory Planner Another approach is to use a neural network to determine a goal for the focal agent and then employ a rule-based planning algorithm to plan a path to that goal. A neural network then predicts the velocity along the planned path, resulting in a trajectory. An example is PTNet [21], which combines the traditional Pure Pursuit path-tracking algorithm with graph-based neural networks to generate a path and an acceleration profile along the path. Similarly, Flash [2] builds on PTNet by introducing a maneuver-based ensemble of neural networks to generate the path.

One advantage of this approach is that it divides the problem into three steps, each with an interpretable result [2]. It also improves the model's sample efficiency since it does not need to learn the laws of agent motion — the deterministic planner handles that [21]. However, the method is not flexible in how the agent reaches the goal, as that is determined by the algorithm and cannot be adapted to the dataset. This works well as long as the goals are close to the focal agent's current position but makes it challenging to predict trajectories with a longer time horizon.

Set-based Approaches Set-based approaches offer an alternative by treating trajectory prediction as a classification problem rather than a regression problem. In this approach, a set of trajectories is pre-generated offline, and the prediction task becomes predicting the probabilities of the trajectories from this set instead of generating a trajectory [51]. For instance, the RESET [49] model introduces an efficient algorithm to extract the trajectory set from the training data and uses LaneGCN [37] as the classification header to select the prediction. Similarly, PRIME [56] uses a Frenét planner to generate a kinematically and map-feasible set of trajectories and employs an RNN and attention to select the most suitable trajectories.

The primary advantage of set-based approaches is that they can easily enforce any kind of constraints on the predictions by only including trajectories in the set that fulfill these constraints [49]. The set also enables them to effectively avoid mode collapse [51, 56]. However, this approach introduces additional prediction error since the exact necessary trajectory may not be included in the pre-generated set. Hence, to mitigate this error, these methods require a large set of trajectories to achieve accuracy. For example, RESET uses a set of 2000 trajectories for its final performance [49].

Action-Space Predictions Action-space predictions differ from the other approaches by predicting a sequence of control inputs, such as acceleration and steering angles, instead of directly predicting the trajectory as a sequence of positions [31]. These control inputs are then integrated over time to produce a trajectory. Intuitively, action-space prediction means placing the network behind the steering wheel and letting it drive to produce a trajectory rather than letting it draw a path on a 2D plane and assign a velocity to every point.

Models such as DKM [11], SSP-ASP [31], Trajectron++ [48], and MultiPath++ [59] follow this approach by incorporating a kinematic layer that calculates the trajectory from the predicted control inputs using a rule-based kinematic model. The kinematic layer is fully deterministic and differentiable, with no learnable parameters [11]. The loss is computed by comparing the output trajectory of the kinematic layer with the ground-truth trajectory. Consequently, even though the model's predictions are now control inputs, the dataset's ground truth remains as trajectories. In other words, action-space prediction does not require altering the dataset.

The advantage of action-space prediction lies in its flexibility and modularity. First, it can be easily integrated into any existing network by adding a kinematic layer after the prediction head, provided the prediction head can be adjusted to predict in action-space rather than trajectory space [11]. This allows for enforcing kinematic constraints without modifying the core architecture of the neural network. Second, action-space prediction offers flexibility in the choice of kinematic models [48] and allows the use of different kinematic models for different agents. However, this approach introduces additional computational demands during forward and backward propagation since the kinematic layer must integrate control inputs step-by-step for each predicted time step.

Kinematic Models

To enforce kinematic feasibility, all four of the approaches discussed rely on a fundamental understanding of the focal agents' kinematics. This understanding is typically formalized through kinematic models [11, 48, 59, 21, 45, 7, 36]. These models are a cornerstone in control theory, robotics, and, more recently, trajectory prediction frameworks.

A kinematic model defines the mathematical relationship between control inputs, such as steering angle, acceleration, and velocity, and the resulting changes in the agent state, including position, orientation, and speed, over time [31]. Grounded in the principles of physics, these models ensure that an agent's motion remains physically feasible. The choice of the kinematic model varies depending on the agent type and involves trade-offs in terms of accuracy, computational complexity, and the need for parameter estimation. Naturally, we divide kinematic models by the type of agent for which they are applied.

Vehicle Kinematic Models For vehicle trajectory prediction, three kinematic models are commonly used.

The **Two-Axle Vehicle Model** provides the most accurate representation of a vehicle's dynamics, modeling the vehicle with two axles and four wheels [11]. This model is particularly suited for problems where precise physical modeling is critical, as it closely mirrors the actual physical behavior of vehicles. However, the model's accuracy comes at a cost: it requires the estimation of numerous vehicle-specific parameters, such as width, length, and wheelbase. These parameters often need to be estimated online, which introduces additional complexity and sources of error during real-time trajectory prediction. The parameters can only be estimated based on tracking data, which already contains some error itself. Thus, the estimated parameters also introduce error to the kinematic computations, which dilutes the Two-Axle Vehicle Model's accuracy.

A commonly used simplification of the Two-Axle Vehicle Model is the **Bicycle Model**, where the vehicle is represented by a single wheel located at the midpoint of each axle [31, 36, 52]. This model strikes a balance between complexity and accuracy, offering a simpler yet sufficiently accurate approximation for many problems. The Bicycle Model reduces the number of parameters that need to be estimated to only the wheelbase length. Thereby it lowers the computational burden. Still, the wheelbase needs to be estimated, which introduces additional error to the kinematics if the estimation is inaccurate.

The **Unicycle Model** further simplifies vehicle dynamics by reducing the vehicle to a single, centrally located, steerable wheel [48, 59]. This model eliminates the need for parameter estimation entirely and is computationally more efficient, although it is less accurate. Nevertheless, works like Trajectron++ [48] argue and demonstrate that the Unicycle Model is sufficiently accurate for the needs of trajectory prediction in practice.

Pedestrian Kinematic Models For pedestrian trajectory prediction, the literature is less agreed upon the choice of the kinematic model.

Some approaches opt to forgo a kinematic model altogether, allowing for **free prediction** of pedestrian positions without enforcing any kinematic constraints [11]. This approach offers maximum flexibility, enabling the model to capture a wide range of pedestrian behaviors. However, the missing guarantee of physical feasibility for pedestrians can be problematic, as discussed in Section 1.1.

Alternatively, the **Single Integrator Model** provides a middle ground by allowing any 2D velocity as control inputs, enabling a broad range of movements while still imposing a maximum speed limit [48]. This model introduces a sense of local change and continuity in movement sequences, which can be beneficial for modeling pedestrian behavior. However, it still permits physically infeasible accelerations, limiting its feasibility guarantees.

The **Unicycle Model** has also been adapted for pedestrian prediction [45, 7, 46]. Some papers argue that pedestrians exhibit non-holonomic motion characteristics when moving in traffic [4, 7]. Hence, they choose the Unicycle Model because it enforces the motion to move on smooth turns instead of abrupt changes in direction, even though such abrupt changes are physically possible for pedestrians. For example, a pedestrian can stop walking forward, turn by 120°, and continue that way. The Unicycle Model would prohibit such a motion. Thus, at the same time, the non-holonomic nature is a disadvantage of the Unicycle Model for pedestrians. It imposes control inputs, such as a steering angle, that are not natural for pedestrians, making it less intuitive for capturing pedestrian movement.

Finally, the **Bicycle Model** has been used to model pedestrian dynamics, treating them just like vehicles [36]. While this approach simplifies the modeling process by applying a consistent framework across all agent types, it is generally not recommended for pedestrians due to its less realistic representation of pedestrian dynamics. The Bicycle Model's assumptions, such as the presence of a wheelbase, do not apply to pedestrians. This raises questions about the accuracy of the modeling.

Research Gaps

Despite significant progress in kinematically feasible trajectory prediction, several critical gaps remain unaddressed in the current body of research. One of the most notable gaps is the lack of properly evaluated kinematic models for pedestrians. Much of the existing research has focused predominantly on vehicles, often neglecting the need for realistic and robust kinematic models for pedestrians. The selection of pedestrian kinematic models in current literature is often ad hoc, with little explanation or justification provided. Moreover, there is a conspicuous absence of comprehensive comparative studies to determine which kinematic models are most effective for pedestrian prediction. This practice can lead to suboptimal predictions, as the unique motion characteristics of each agent type are not adequately captured. Instead, we suggest addressing this gap with a more nuanced approach, where suitable kinematic models are employed for each class of agent.

Another gap in the literature is the lack of integration between feasibility and interpretability in interaction models. Current research often treats these two aspects in isolation, focusing either on ensuring kinematic feasibility or on enhancing interpretability. However, as discussed in

Section 1.1, the alignment of trajectory predictions with human-like reasoning benefits significantly more when both feasibility and interpretability are addressed together. Consequently, we propose to combine a kinematically feasible decoder with an interpretable interaction encoder, offering the potential to improve the overall alignment and reliability of trajectory predictions in autonomous systems.

3 Method

In this chapter, we present our methodology to improve the feasibility and interpretability of deep learning-based trajectory prediction models for autonomous driving. The proposed method integrates expert knowledge in the form of priors for both social interactions and agent-specific kinematics. The approach is designed to enhance the alignment between the model's predictions and human expectations while ensuring that the predicted trajectories are kinematically feasible for each agent class.

The chapter is structured as follows. First, we formally define the problem of trajectory prediction. Following this, we provide an overview of the core concepts that guide our approach, including our contributions, an introduction to our backbone, and a high-level system overview. Finally, we mathematically define the class-specific interpretable encoder and feasible decoder layers.

3.1 Problem Definition

To make the problem of trajectory prediction tangible, we follow the mathematical formulation outlined by Schuetz et al. [51]. Consider a traffic scene with N_{AG} agents and N_{MP} map elements, where our goal is to predict the future trajectory of a specific focal agent $i \in \{1,\dots,N_{AG}\}$. We are given a sequence of observed states for agent i over T_o observation time steps, denoted as $S_i = [s_i^1, s_i^2, \dots, s_i^{T_o}]$, where $s_i^t \in \mathbb{R}^{dim_{AG}}$ represents the state of agent i at time t. The task is to predict the future trajectory $Q_i = [q_i^{T_o+1}, q_i^{T_o+2}, \dots, q_i^{T_o+T_s}]$ over a future time horizon T_s, where $q_i^{T_o+1} = (x_i^{T_o+1}, y_i^{T_o+1})$ is the agent's position immediately following the observation period.

The observed states s_i most importantly include the agent's position but may also encompass other relevant properties such as velocity, heading, and additional features derived from sensor inputs [51]. In addition to the focal agent state observations S_i, we assume that additional context information is provided, namely the state observations of all other agents S_j for $j \in \{1,\dots,N_{AG}\}$ and the features of map elements $M_l \in \mathbb{R}^{dim_{MP}}$ in the scene for $l \in \{1,\dots,N_{MP}\}$. Together, agents and map elements form the prediction contexts $S = [S_1,\dots,S_{N_{AG}}]$ and $M = [M_1,\dots,M_{N_{MP}}]$.

The challenge lies in learning a function f that accurately maps the observed states S_i to a corresponding future trajectory Q_i while considering the context S and M, i.e., $Q_i \stackrel{!}{=} f(S_i, S, M)$. However, predicting the exact future trajectory of an agent is inherently difficult, even for humans, due to the uncertainty and variability in real-world environments. As a result, most research in deep learning approaches has shifted towards modeling a probability distribution $P(Q_i|S_i, S, M)$ over possible future trajectories rather than predicting a single future trajectory [51]. This distribution can be multimodal, meaning it can capture multiple likely outcomes that reflect the different choices an agent might make in the future, so-called *modes*. An example

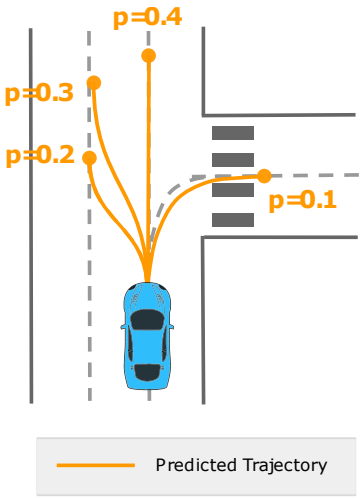


Figure 3.1: Multimodal trajectory prediction for an agent who could choose to either continue straight, switch lanes, or turn right at an intersection. All four modes can be represented within a single prediction's distribution. Adapted from [70].

of multimodal prediction is illustrated in Figure 3.1. By sampling from the learned distribution $P(Q_i|S_i,S,M)$, the model can generate multiple plausible trajectories, providing a more robust and flexible prediction [51].

In the context of autonomous driving, trajectory prediction models utilize additional input information beyond the past positions of the agents. This includes map knowledge, which provides essential environmental context such as road layouts, lane boundaries, and static obstacles, as well as the positions and movements of other agents in the scene. These inputs are crucial for modeling the complex interactions that naturally occur between agents in dynamic traffic environments [51].

Agents are typically categorized into three main classes for trajectory prediction tasks: vehicles (*veh*), pedestrians (*ped*), and cyclists (*cyc*) [15]. Notably, bicycles and motorcycles are grouped together under the cyclist category due to their similar maneuverability in traffic situations. However, for simplicity, our primary focus is on differentiating between vehicles and pedestrians to demonstrate the effect of using class-specific priors for trajectory prediction. This approach can be readily extended to other taxonomies of agent classes as needed.

3.2 Concept Overview

In Section 2.1.2, we saw that current state-of-the-art models achieve high accuracy but lack alignment. However, as elaborated in Section 1.1, alignment is crucial to safety-critical applications like autonomous driving. It allows us to build trust in the networks by understanding some of the reasoning inside the black-box technology. The key idea is that even if the network makes an incorrect prediction, that prediction is still reasonable according to what a human would expect. As we have argued in Section 1.2, feasibility and interpretability are two concepts that can significantly improve the model's alignment. Both draw the network's decision-making process

closer to how a human would reason when making decisions. Hence, in this thesis, we propose to improve the alignment of neural networks with guaranteed per-class kinematic feasibility and interpretable social interactions.

3.2.1 Contributions

The core approach of this thesis is to integrate expert knowledge and introduce per-class experts to make the prediction a mixture of experts. Specifically, we search for expert knowledge in the form of one prior for social interactions and one prior per agent class for the kinematics.

For social interactions, we compare two priors that determine the importance of each surrounding agent for a given prediction. We take one prior from recent literature [39] and propose a second novel prior that more closely represents human intuition. Without a prior, the network estimates the influence of surrounding agents using an attention-based encoder. It predicts an attention score for each agent. This score determines the agent's impact on the encoded information for the focal agent's prediction. We integrate the prior by combining its knowledge-based interaction importance scores with the network's predicted attention scores. We realize this combination as a mixture of experts further discussed in Section 3.3.

This integration guides the network to learn interaction importance in a manner consistent with the prior. The knowledge-based interaction importance scores model human understanding of which agents matter for interaction. Hence, by integrating the prior, the prediction gains interpretability not only of which agents the network attends to but also of how that attention relates to how a human would reason in this scenario. This enables us to monitor the prediction's alignment. In particular, we can detect where the network's predicted attention scores strongly diverge from the prior scores, i.e., where the model's interaction reasoning strongly diverges from human reasoning.

For kinematics, we integrate the well-established unicycle model [33] for vehicles and compare the unicycle, double integrator, and single integrator models [48] for pedestrians. As examined in Section 2.2.2, the unicycle [46, 7, 45] and single integrator model [48] are the two most popular options for modeling pedestrian kinematics in previous research. Additionally, we propose using the double integrator as a compromise between the two. Through comparison, we identify the kinematic model with the smallest gap to reality in modeling pedestrian movements. Without such a kinematic model, the network predicts trajectories as sequences of (x, y) coordinate pairs. By integrating a kinematic model, we instead let the network predict control parameters like acceleration and steering angle for each step. The network's predictions are then propagated through an additional kinematic layer [11] to obtain the final output as sequences of (x, y) coordinate pairs. We detail integrating the kinematic model in Section 3.4.

As a result, our approach guarantees kinematic feasibility. By identifying the most realistic kinematic model and distinguishing the kinematic models of vehicles and pedestrians, we reduce the prediction's gap to reality and thus improve alignment with human predictions.

We propose a two-step solution, namely integrating priors for interaction encoding and trajectory decoding, because it addresses alignment in both stages of the model: the encoder and the decoder. Adapting both stages is essential to ensure comprehensive alignment with human pre-

dictions. Without both solution steps, one could undermine the other. A feasible decoder without an interpretable encoder might produce predictions that are physically possible but are still unlikely to be predicted by a human because the reasoning is wrong. Conversely, an interpretable encoder without a feasible decoder may result in predictions that, despite being well-reasoned in the bigger picture, are physically impossible and hence would not be predicted by a human. This demonstrates that both cases lead to unaligned predictions. That is why we suggest tackling alignment in the encoder and decoder simultaneously.

3.2.2 Backbone

We choose the **Hierarchical Predictive Transformer with Relative Pose Encoding (HPTR)** model [69] as the backbone for our trajectory prediction network. In this section, we first introduce the technical details of HPTR that are relevant to our thesis and then explain the rationale behind this choice.

HPTR Details

As discussed in Section 2.1.2, HPTR's Transformer architecture is designed for efficient and scalable trajectory prediction. A distinguishing feature of HPTR, compared to traditional Transformer-based models, is its novel attention mechanism, K-Nearest Neighbor Attention with Relative Pose Encoding (KNARPE) [69]. KNARPE limits attention to the K-nearest neighbors of each source agent i, reducing the computational complexity of standard attention mechanisms by focusing on relevant agents in the scene. The K-nearest neighbors are the K agents $J_i \subseteq \{1,\dots,N_{AG}\} \setminus \{i\}$ in the scene that are closest in position to i measured by the Euclidean distance. For each source agent i, HPTR calculates the relative pose rel_{ij} between agent i and each of its K-nearest neighbors j [69]:

$$\text{rel}_{ij} = (x_{ij}, y_{ij}, \theta_{ij}) = (x_j - x_i, y_j - y_i, \theta_j - \theta_i). \quad [3.1]$$

Here, (x_i, y_i) is the 2D position of agent i, θ_i the heading angle of agent i, (x_{ij}, y_{ij}) represents the relative 2D position of agent j with respect to i, while θ_{ij} is the relative heading angle of agent j with respect to i.

This relative pose effectively captures the spatial relationships between agents. To incorporate these spatial relationships into the attention mechanism, HPTR transforms rel_{ij} into a higher-dimensional space using Relative Pose Encoding (RPE). The RPE is defined as follows [69]:

$$\text{RPE}(\text{rel}_{ij}) = \text{concat}(\text{PE}(x_{ij}), \text{PE}(y_{ij}), \text{AE}(\theta_{ij})), \quad [3.2]$$

where $\text{PE}(x)$ and $\text{PE}(y)$ are sinusoidal position encodings of the relative position [69]. $\text{AE}(\theta)$ is a sinusoidal angular encoding for the relative orientation [69]. After computing the RPEs, HPTR calculates attention over the K-nearest neighbors by using the embedding u_i of each agent i as the query and $(u_j, \text{RPE}(\text{rel}_{ij}))$ as keys and values for each neighbor j.

HPTR's KNARPE mechanism combines the strengths of Graph Neural Networks (GNNs) and Transformers. As explained in Section 2.1.2, the core principle of GNNs is that a node's features are updated by aggregating information from its neighboring nodes. This process consists of two main steps: *aggregation* and *node update*. During the aggregation phase, each node gathers information from its neighbors by collecting their feature vectors. In the node update step, each node updates its own feature vector based on the aggregated information received from its neighbors.

HPTR mirrors this process through KNARPE. The K-nearest neighbor mechanism serves as the *aggregation* function, identifying the relevant neighbors and retrieving their embeddings. The RPE mimics the edge features of a GNN, representing the relationships between agents through relative positions and headings. By encoding these relationships with sinusoidal and angular encodings, HPTR captures the spatial dependencies between agents, much like GNNs capture graph-based relationships. Finally, KNARPE performs the *node update* by modifying the agent's embedding based on all neighbors' embeddings.

Backbone Choice

HPTR is particularly well-suited for our thesis for several reasons. First, HPTR's integration of the GNN-inspired approach to local interactions fits well with our goal of aligning interactions with human expectations. Conceptually, interactions between agents in traffic always occur locally: agents that are not within each other's proximity do not interact. By restricting interactions to local neighborhoods, HPTR already takes a first step toward aligning interactions with human reasoning.

Additionally, HPTR's modular attention layer architecture makes it straightforward to introduce an agent-to-agent attention layer guided by an interaction importance prior, which is central to the contributions of our thesis. HPTR also achieves state-of-the-art performance with a relatively small number of parameters compared to other Transformer-based models. For instance, while Wayformer requires approximately 60 million parameters, HPTR operates with only 15.2 million parameters [17, 69]. This efficiency, combined with its scalability and fast inference thanks to KNARPE, ensures that HPTR is well-suited for real-time autonomous driving applications. Lastly, HPTR benefits from a high-quality open-source implementation, which makes it easy to experiment with and extend for our purposes.

In summary, HPTR's novel KNARPE mechanism, combined with its efficiency and adaptability, makes it the ideal choice for our backbone. It allows us to integrate expert knowledge for improving the feasibility and interpretability of trajectory predictions, while also providing state-of-the-art performance.

3.2.3 Model Architecture

Building upon the chosen HPTR architecture [69], our neural network is structured as depicted in Figure 3.2. It is divided into two main components: the encoder and the decoder.

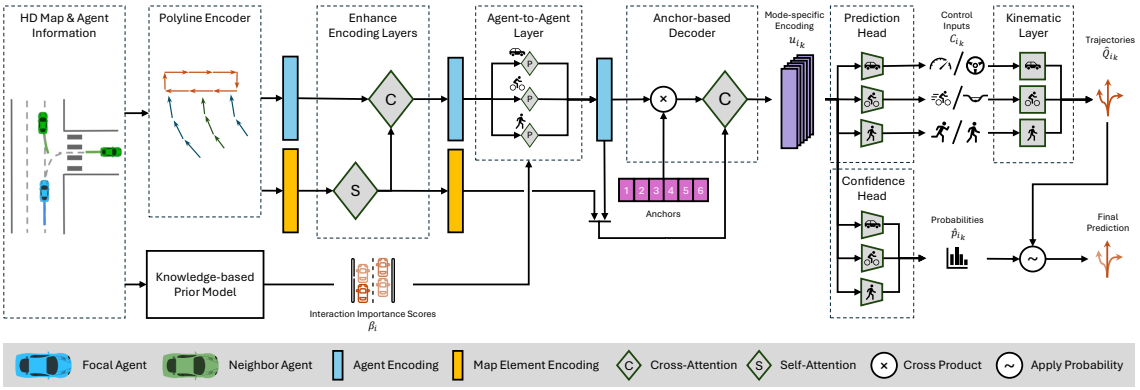


Figure 3.2: Network overview. The scene is first encoded in agent and map element embeddings and enhanced by transformer layers. Then, the prior-integrated and class-specific agent-to-agent layer captures interactions between agents. Next, transformer layers generate mode-specific embeddings based on the agent embeddings. Subsequently, class-specific MLP heads predict a sequence of control inputs and a confidence value per mode. Finally, a kinematic layer deterministically calculates the predicted trajectories resulting from the control inputs.

Encoder As in HPTR, the encoder first represents the scene consisting of agents S and map elements M using polylines, a technique adapted from VectorNet [19]. This representation eliminates the temporal dimension, allowing the network to focus exclusively on spatial relations in all subsequent layers, as discussed in Section 2.1.2. The next step involves a series of transformer layers. The first six layers enhance the map element embeddings through map-to-map attention, which refines the features of each map element based on its K-nearest neighbor map elements, such as lanes and crosswalks. Following this, two map-to-agent attention layers are employed to further refine the agent embeddings by integrating relevant environmental information.

Next, we propose an agent-to-agent attention layer extending HPTR's architecture which we detail in Section 3.3. This layer enhances the agent embeddings based on the interactions between the agents in the scene. To guide the attention toward aligned social interactions, we provide this layer with the interaction prior by incorporating the prior-knowledge-based interaction importance scores. The layer has one instance per agent class, where each instance is trained to become an expert for vehicles, pedestrians, or cyclists, respectively.

Decoder For each focal agent i, the agent embedding u_i computed by the encoder is then passed to the decoder to generate predictions. As in HPTR, the decoder begins by generating mode-specific embeddings u_{i_k} by concatenating a set of K_{AC} learnable anchors. Each anchor corresponds to a possible future mode $k \in \{1, \ldots, K_{AC}\}$. Two all-to-anchor transformer layers then aggregate information from the encoder's embeddings of all other agents and map elements.

From here, we change HPTR's architecture as we detail in Section 3.4. First, as prefigured in Section 3.2.1, instead of a sequence of coordinates, an MLP prediction head generates a sequence of control inputs $C_{i_k} = [c_{i_k}^{T_o+1}, \ldots, c_{i_k}^{T_o+T_s}]$ per agent-anchor embedding u_{i_k}. The control inputs are predicted in a one-shot manner, meaning that the control inputs for all time steps are predicted as a single concatenated output in one forward pass of the MLP. Alongside this, an

MLP confidence head predicts a probability distribution over these K_{AC} modes, indicating the likelihood $\hat{p}_{i_k}$ of each mode occurring.

Finally, we introduce a kinematic layer to HPTR's architecture. This layer integrates the control inputs C_{i_k} to obtain the K_{AC} predicted trajectories $\hat{Q}_{i_k}$. The kinematic layer is differentiable and contains no learnable parameters. To achieve this, it recurrently applies agent-class-specific kinematic models, embodying one expert per class. Following the class-specific kinematic layer instances, we also employ class-specific instances of the MLP prediction head. Ultimately, the combination of the trajectories produced by the kinematic layer and the probability distribution predicted by the MLP confidence head forms the multimodal output distribution $\hat{P}(\hat{Q}_{i_k}|S_i,S,M) = \hat{p}_{i_k}$.

3.3 Interpretable Encoder

As discussed in Section 1.1, the importance of interpretable trajectory prediction lies in its alignment with human expectations. In complex traffic scenarios, agents like vehicles and pedestrians interact with each other in ways that significantly influence their future trajectories. Humans intuitively understand these interactions based on their knowledge of traffic rules, behaviors, and social dynamics. However, neural networks, despite their ability to learn from vast amounts of data, do not naturally provide insight into the reasoning behind their predictions. Without additional interpretability mechanisms, it becomes difficult to assess whether the model's internal logic aligns with what a human would expect, making the predictions less trustworthy, especially in safety-critical applications like autonomous driving.

In this section, we introduce our method for making interactions more interpretable. Therefore, we first present the class-specific Transformer layer with agent-to-agent attention. Then, we define two possible interaction priors we employ to align that layer's attention. Finally, we propose the integration of priors into the Transformer layer.

3.3.1 Class-Specific Agent-to-Agent Transformer Layer

Our goal is to improve the network's agent-to-agent interaction capabilities. Thus, we introduce a new, separate agent-to-agent attention Transformer layer. This layer's primary task is to explicitly focus on agent-to-agent interactions after the initial map and agent embedding have been computed.

While the exchange of information between agent embeddings is already possible as part of other HPTR layers, a separate final layer simplifies the model design. It isolates the changes required for interpreting social interactions, ensuring the rest of the model remains unchanged and compatible with standard HPTR. Hence, it allows us to enforce aligned attention only in this final interaction stage while leaving the attention in earlier layers unconstrained.

Making agent-to-agent attention the final step in the encoder is sensible because interactions are the central challenge of trajectory prediction. Without the interactions between agents, approaches that do not require deep learning would mostly suffice for trajectory prediction [50, 22]. Accordingly, we recognize the encoding of the interactions between agents as the most important

step of the encoder. Thus, we make the agent-to-agent interaction encoding the encoder's last step, where it has the most prominent impact on the final agent embeddings and, consequently, on the prediction.

Additionally, we introduce one instance of this new layer per agent class. Each instance can be considered an expert for its corresponding class. We argue that class-specific interaction layers are essential to more aligned interactions because different agent classes exhibit different interaction behavior. For example, a pedestrian on a sidewalk may pay little attention to passing cars but will be more influenced by oncoming other pedestrians. A simple feature indicating the agent's class cannot capture the distinct interaction behavior. Moreover, class-specific layers help address the data imbalance in trajectory prediction datasets, where the vast majority of agents are vehicles. Without a separate layer for pedestrians and cyclists, the model will predominantly learn vehicle interactions, neglecting the interaction patterns of the other classes. Even if the dataset was balanced, the different scales of the loss for the different agent types would still lead to pedestrian behavior being mostly ignored. As vehicles move faster and farther, pedestrian prediction's displacement error is always low compared to that of vehicle predictions. Hence, the loss pedestrians contribute will mostly be ignored compared to the vehicles' loss when updating the weights. Additionally, since the decoder is already class-specific, having a corresponding class-specific encoding ensures consistency in the model's decision-making process.

For the scope of this thesis, the final agent-to-agent attention layer is the only place where we introduce class-specific layer instances. This shall serve as a proof of concept since this layer is the most important layer of the encoder in our understanding. The concept of class-specific layers could be readily expanded to most other encoder layers, too. However, the trade-off between performance and model size would be crucial for such an expansion.

To implement this, we define one expert layer instance for each of the three agent classes. For a given agent i, the network selects the expert layer corresponding to its agent class to compute the attention for i to the other agents in the scene. Mathematically, the agent-to-agent attention layer for a focal agent i is defined as

$$l_{\text{ag-ag}}(u_i, U) = \begin{cases} l_{\text{ag-ag}_{veh}}(u_i, U) & \text{if agent\_class}(i) = veh, \\ l_{\text{ag-ag}_{ped}}(u_i, U) & \text{if agent\_class}(i) = ped, \\ l_{\text{ag-ag}_{cyc}}(u_i, U) & \text{if agent\_class}(i) = cyc. \end{cases} \quad [3.3]$$

Here, $U = \{u_1, \ldots, u_N\}$ denotes the set of all agent embeddings and $u_i \in U$ the embedding of agent i. Then, $l_{\text{ag-ag}}$ denotes the final agent-to-agent attention layer which uses its first argument as the query and its second argument as the keys and values. $l_{\text{ag-ag}_{class}}$ are three Transformer encoder layer instances that are identical in their computational process but have different instances of weights. The computation process of these layers is the same as for any other Transformer encoder layer in HPTR.

3.3.2 Interaction Priors

Our next step in making interactions more interpretable is to incorporate prior knowledge through a rule-based *interaction importance score*. This prior knowledge is then combined with the agent-to-agent layer's attention mechanism to guide the layer toward more aligned interactions. Here, we were inspired by the mixture of expert approaches in other fields, as introduced in Section 2.2.1. As a prerequisite, we must choose an appropriate prior model for the task.

To ensure a comprehensive analysis, we introduce and compare two different interaction prior models. The first model is SKGACN's [39]. It is a natural choice since it is the only prior model for interaction importance that has been applied in trajectory prediction. However, as discussed in Section 2.2.1, SKGACN is specifically designed for pedestrian interactions, making its effectiveness unclear when applied to mixed traffic scenarios involving both pedestrians and vehicles. Therefore, we search for a second prior as an alternative to SKGACN's. By comparing the results from both priors, we aim to analyze how the quality of the prior affects model performance.

SKGACN Prior

SKGACN provides a straightforward rule-based method for calculating interaction importance scores, which is exactly what we need for our approach to interpretable interactions. As introduced in the Section 2.2.1, SKGACN computes interaction importance scores based on agents' relative distance, speed, and heading angle. However, despite its simplicity, SKGACN has notable limitations when applied to traffic involving both pedestrians and vehicles.

As SKGACN was designed for pedestrian interactions, it does not always align well with human intuition of vehicle-vehicle or vehicle-pedestrian interactions. For example, it assumes that pedestrians do not need to pay attention to pedestrians behind them. Particularly, it ignores the speed and heading of agents behind the focal agent [39]. While this assumption may be valid for pedestrians walking in a pedestrian zone, it does not extend to vehicles. In traffic, vehicles must account for other agents approaching from behind, which SKGACN fails to model effectively.

Additionally, SKGACN scales the interaction importance score inversely with distance, meaning that agents far away always matter less than those nearby [39]. For example, a parked car close to the focal agent usually receives a higher importance score than a rapidly approaching vehicle from a greater distance. This issue is problematic in traffic scenarios where dynamic agents, not static objects, require greater attention.

DG-SFM Prior

Given these shortcomings of SKGACN, we searched for a prior that more closely reflects human expectations to see whether a better-aligned prior results in better-aligned trajectory predictions.

Motivation Our search led us to the well-established approach of modeling interactions with the Social Force Model (SFM). The SFM has been widely adopted for pedestrian trajectory

prediction. As introduced in Section 2.2.1, it simulates social forces based on agents' proximity and velocities. With extensions like the Social-Force-based Vehicle Model, it has also been applied to vehicles [29, 14].

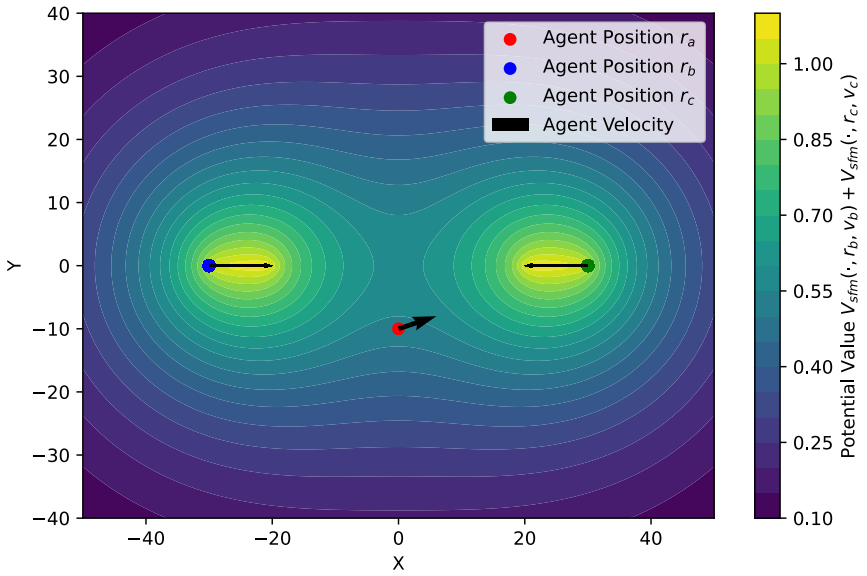
However, directly using the SFM's social forces as interaction importance scores would introduce notable challenges. Firstly, the SFM does not consider the heading and speed of the focal agent. That means, it has no sense of the focal agent's driving motion as depicted in Figure 3.3a. Secondly, the social force potential's front and back are symmetric relative to the semi-minor axis. Consequently, the SFM gives the same score to agents driving towards the focal agent as to those driving away from the focal agent. Figure 3.3b illustrates this problem.

Both problems would lead to interaction importance scores that are misaligned with human intuition. Hence, we propose a novel prior model: the *Directed-Gradient Social Force Model* (DG-SFM). It extends the SFM by tailoring its formulation to fit our use case of interaction importance scores. With the DG-SFM, we aim to overcome the shortcomings of SKGACN by leveraging the modeling concept of the SFM to account for the dynamics of both pedestrians and vehicles.

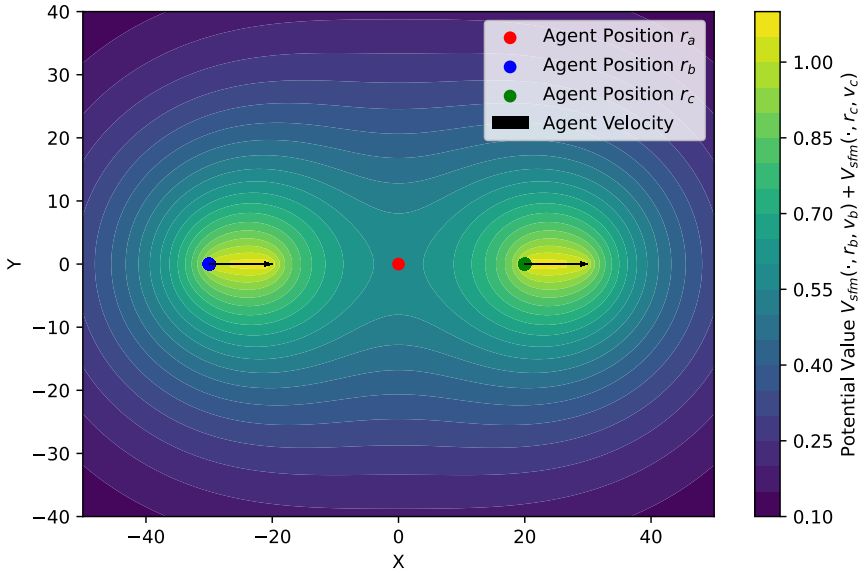
Definition The intuition behind the DG-SFM model is that the importance of agents for interaction with the focal agent depends on their speed and whether their distance to the focal agent is increasing or decreasing. For example, a vehicle quickly approaching from far behind should be considered highly important, as it poses a potential risk of collision. In contrast, a vehicle traveling in the neighboring lane at the same speed as the focal agent should have medium importance since it is neither gaining nor losing distance relative to the focal agent. Finally, a parked vehicle on the side of the road that the focal agent has already passed should have little to no importance, as an interaction is highly unlikely after having passed it. The same intuition is also applicable to pedestrians and cyclists, making DG-SFM suitable for all classes of agents.

To formalize this intuition, we build upon the Social Force Model (SFM) by using its concept of elliptical social repulsive potentials as a way to define each agent j's personal space. In this formulation, the higher the potential value that another agent has within j's force field, the more dangerous that agent is to j, reflecting the level of interaction importance. An agent can be significant for interactions in two key ways. First, if agent j is within the focal agent's personal space, the focal agent must monitor j's actions closely to be aware of interactions that could be triggered by j. Second, if the focal agent intrudes into agent j's personal space, the focal agent actively triggers an interaction with j. Then, it becomes crucial for the focal agent to evaluate whether j reacts according to their expectations.

Mathematically, we define the DG-SFM as consisting of two components, following the two reasons from the aforementioned intuition for interaction importance. Let us consider the task of calculating the interaction importance scores of all neighboring agents $j \in J_i \subseteq \{1, \ldots, N_{AG}\} \setminus \{i\}$ for the focal agent i.



(a) The heading and velocity of agent a do not influence the SFM potential.



(b) The heading of agent b and c, towards or away from agent a, does not influence the SFM potential, except for a constant shift in position.

Figure 3.3: The SFM social repulsive potential field $V_{\text{sfm}}(\cdot, r, v)$ of two agents b and c in two scenarios. In both scenarios, focal agent a experiences the same force and potential value from both agents, even though, intuitively, b is significantly more important for a's trajectory prediction.

The first component $\hat{\beta}_{ij}^{\text{dg-sfm}_A}$ gives importance to agents j with respect to how close they are to i, measured using their value in i's repulsive potential field:

$$\hat{\beta}_{ij}^{\text{dg-sfm}_A} = V_{\text{egg-sfm}}(r_j, r_i, v_i). \tag{3.4}$$

Here, $r_i = [x_i, y_i]^T \in \mathbb{R}^2$ is the 2D Cartesian position of i at the last observation time $t = T_o$. $v_i = [v_{x_i}, v_{y_i}]^T \in \mathbb{R}^2$ is the 2D velocity of i at $t = T_o$. $V_{\text{egg-sfm}}$ describes a repulsive potential

modified from the SFM's V_{sfm}, which we introduce later. $V_{\text{egg-sfm}}$ calculates the value of the first parameter's agent in the potential field of the second parameter's agent.

The second component $\hat{\beta}_{ij}^{\text{dg-sfm}_B}$ gives importance to agents j with respect to how fast i is approaching them. Therefore, we compare the value of i in j's potential at the current time and at the time after $N_{DG} \in \mathbb{N}$ time steps into the future:

$$\hat{\beta}_{ij}^{\text{dg-sfm}_B} = V_{\text{egg-sfm}}(r_i^*, r_j^*, v_j) - V_{\text{egg-sfm}}(r_i, r_j, v_j). \quad [3.5]$$

Here, r_i^* and r_j^* are the estimated future positions that we obtain from applying the CVM. Especially for the short time horizon of N_{DG} time steps, the CVM is a sufficiently good approximation of agent motion [50]. Therefore, we calculate the estimated future positions as

$$\forall l \in \{i, j\}.\ r_l^* = r_l + N_{DG} \cdot v_l \Delta t. \quad [3.6]$$

Here, Δt is the time between agent state observations, for example, between $s_i^{T_o}$ and $s_i^{T_o+1}$. As we pick N_{DG} to be small, e.g., $N_{DG} = 1$, $\hat{\beta}_{ij}^{\text{dg-sfm}_B}$ can be understood as a discrete directed gradient of j's repulsive potential in the direction of i's velocity with respect to time. Hence the name Directed Gradient Social Force Model.

To combine the two components, we add them with a configurable weight $\tau_{\text{dg-sfm}_{\text{sum}}} \in [0, 1]$:

$$\hat{\beta}_{ij}^{\text{dg-sfm}} = \tau_{\text{dg-sfm}_{\text{sum}}} \cdot \hat{\beta}_{ij}^{\text{dg-sfm}_A} + (1 - \tau_{\text{dg-sfm}_{\text{sum}}}) \cdot \hat{\beta}_{ij}^{\text{dg-sfm}_B}. \quad [3.7]$$

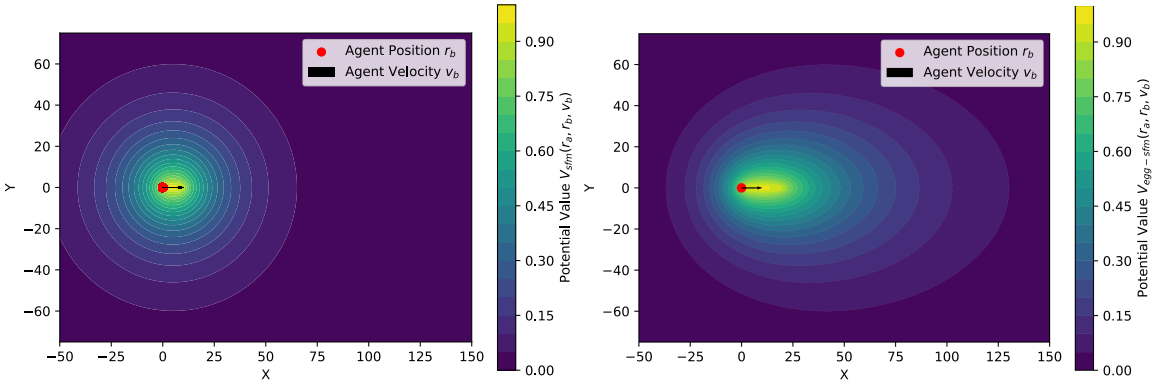
Additionally, we reduce the importance of stationary agents, such as parked cars or pedestrians waiting at a traffic light, compared to moving agents by a constant penalty factor $\tau_{\text{dg-sfm}_{\text{standing}}} \in [0, 1]$. This is based on the intuition that it is generally safer to pass a stationary agent, as standing is often a sign that the agent has yielded the right of way or is inactive. Accordingly, we calculate

$$\hat{\hat{\beta}}_{ij}^{\text{dg-sfm}} = \begin{cases} \hat{\beta}_{ij}^{\text{dg-sfm}} & \text{if } ||v_j|| > 0.1 \text{ m/s}, \\ \hat{\beta}_{ij}^{\text{dg-sfm}} \cdot \tau_{\text{dg-sfm}_{\text{standing}}} & \text{else.} \end{cases} \quad [3.8]$$

Lastly, similar to SKGACN, we obtain the final interaction importance scores $\beta_{ij}^{\text{dg-sfm}}$ by normalizing $\hat{\hat{\beta}}_{ij}^{\text{dg-sfm}}$ with a softmax function over all neighbors $j \in J_i$. Hence, we ensure that $\beta_{ij}^{\text{dg-sfm}} \in [0, 1]$ and that $\sum_{j \in J_i} \beta_{ij}^{\text{dg-sfm}} = 1$.

Completing our definition, we adapt the shape of the repulsive potential so that it grows in an egg shape along the agent's velocity direction, rather than consisting of symmetric ellipses, as illustrated in Figure 3.4. Therefore, we introduce $V_{\text{egg-sfm}}(r_a, r_b, v_b)$ to describe the strength of agent b's repulsive potential field evaluated at the position of agent a. To obtain an egg shape, it shifts the position of b further in the velocity direction the bigger the distance from a to b:

$$V_{\text{egg-sfm}}(r_a, r_b, v_b) = V_{\text{sfm}}(r_a, \tilde{r}_b, N_{DG} \cdot v_b), \quad [3.9]$$



(a) The potential field $V_{\text{sfm}}(\cdot, r_b, v_b)$.

(b) The potential field $V_{\text{egg-sfm}}(\cdot, r_b, v_b)$.

Figure 3.4: A comparison of the potential fields of the regular SFM (a) and the proposed DG-SFM (b). Both plots show the potential field of an agent b at position $r_b = [0,0]^T$ with velocity $v_b = [10,0]^T$ m/s. The color in the plot indicates the strength of the potential at the position indicated by the x- and y-axis measured in meters.

$$\tilde{r}_b = \begin{cases} r_b + \frac{v_b}{||v_b||} \cdot ||r_b - r_a|| & \text{if } ||v_b|| > 0.1 \text{ m/s}, \\ r_b & \text{else.} \end{cases} \quad [3.10]$$

Here, $\tilde{r}_b$ is the shifted position of b. Moreover, by multiplying the velocity with N_{DG}, we scale the length of the ellipse so that its scale fits the number of time steps we take in the discrete directed gradient.

Comparison To Existing Priors DG-SFM offers several advantages over existing models. First, unlike the SFM, DG-SFM considers the heading and velocity of the focal agent, allowing for more informed interaction scores. Second, thanks to the egg-shaped repulsive potential, DG-SFM provides asymmetric importance scores between front and back, where agents in front of the focal agent are treated differently from those behind, ensuring a more realistic modeling of interactions in vehicle traffic. Third, as opposed to SKGACN, DG-SFM does look behind: It accounts for agents approaching the focal agent from behind, which intuitively makes sense for driving. For example, as illustrated in Figure 3.5, a fast vehicle quickly approaching from behind receives a high score from DG-SFM, while SKGACN gives the vehicle a low score because it is behind the focal agent. Finally, DG-SFM's interaction scores are not inversely proportional to distance, allowing agents farther away but approaching quickly to receive higher scores, addressing the limitations of SKGACN's distance-based scaling. An agent very close to the focal agent can receive a low score if it is close but not in the way of the focal agent. For instance, as illustrated in Figure 3.6, DG-SFM assigns a nearby parked car a low score because another agent in the scene is more relevant, while SKGACN gives the parked car a high score due to its proximity.

Despite these advantages, DG-SFM also presents some disadvantages when compared to existing priors. First, DG-SFM is a more complex model, introducing additional hyperparameters such as the weight between the two importance components or the penalty factor for stationary agents. These hyperparameters require tuning, increasing the effort needed for model calibra-

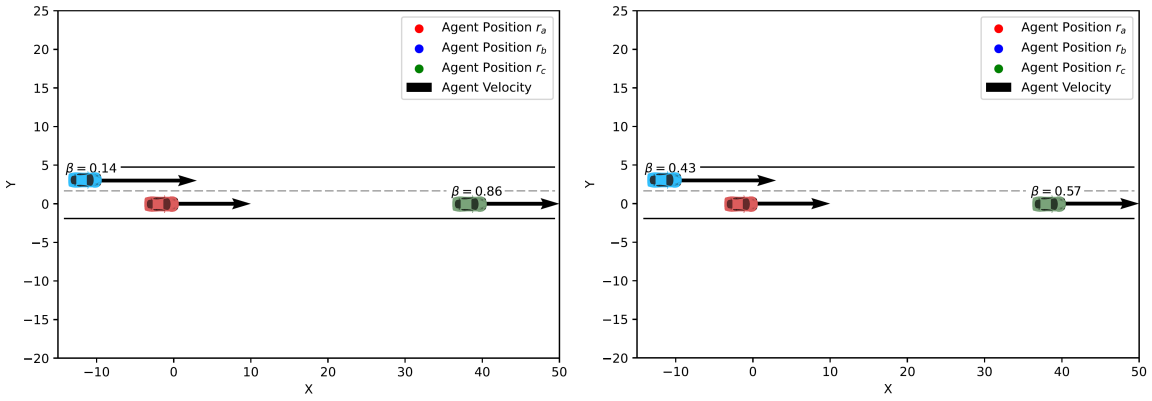


Figure 3.5: A comparison of the interaction importance scores β calculated by SKGACN (left) and DG-SFM (right) for an overtaking scenario. A fast agent b is quickly approaching the focal agent a in the left lane from behind. Another agent c is driving ahead of a with the same speed as a. b receives a high score from DG-SFM, while SKGACN gives b a low score because it is behind the focal agent. Intuitively, b should not be irrelevant for a's trajectory prediction, because b will overtake a and will thus prevent a from moving into the left lane.

tion. Moreover, the added complexity also increases the implementation effort compared to simpler approaches like SKGACN. Second, just like SKGACN and the SFM, DG-SFM relies on heuristic rules for calculating interaction importance. While these rules are grounded in human intuition, they may not capture all possible traffic scenarios, particularly in highly dynamic or unpredictable environments where human reasoning might fail to provide optimal predictions. As a result, the DG-SFM model may struggle to generalize to all possible traffic contexts.

In summary, the DG-SFM model addresses the gaps left by SKGACN and provides a more aligned, intuitive, and human-like interaction importance score. Still, it is only a heuristic and does not provide ideal scores for all scenarios. We evaluate the theoretical advantages and disadvantages in Section 5.2.3 by comparing SKGACN and DG-SFM in real traffic scenarios to assess the practical benefits of a more aligned interaction prior.

3.3.3 Interaction Prior Integration via Mixture of Experts

In the previous two sections, we have introduced an agent-to-agent attention layer and established a prior model for what the attention scores of this layer should look like to align with human intuition. The final question remaining is how to integrate the prior model's interaction importance score into the deep learning layer.

It is logical to introduce human driving knowledge as a prior in this stage for two reasons. First, because the attention scores are inherently interpretable, they provide a clear and human-understandable basis for designing priors. This allows us to encode domain-specific knowledge directly into the network in a way that humans can easily understand and validate. Second, the attention scores have a significant impact on the prediction because they decide the final composition of the embeddings that are passed to the decoder. These embeddings form the foundation for the network's decoder to make predictions, meaning that the attention scores effectively shape the network's decision-making process. For these two reasons, integrating a

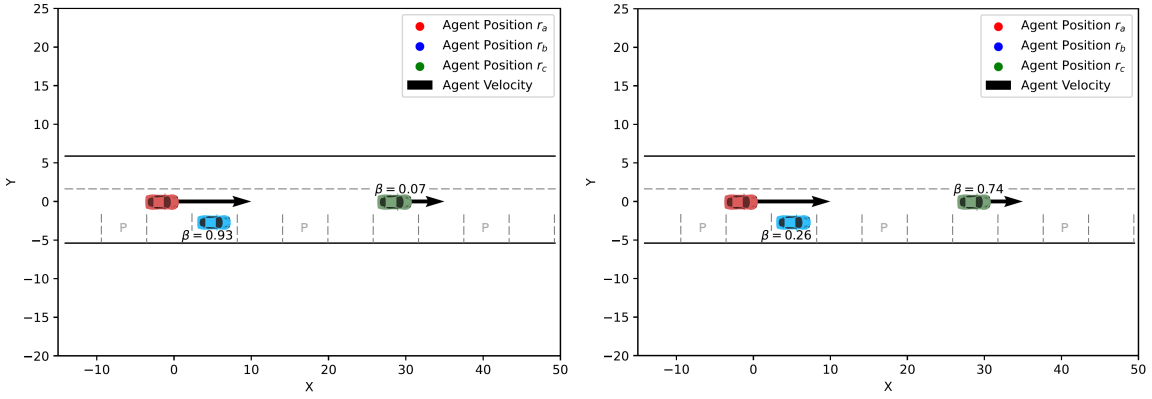


Figure 3.6: A comparison of the interaction importance scores β calculated by SKGACN (left) and DG-SFM (right) for passing a parking car. An agent b is parked in a roadside parking space in front of the focal agent a. The focal agent is catching up with another agent c ahead of it. b receives a low score from DG-SFM, while SKGACN gives it a high score due to its proximity. Intuitively, c should be more relevant for a's trajectory prediction than b, because a will need to take some action to avoid crashing into c soon. b, on the other hand, will most likely not affect a. Still, there is a chance that b pulls out of the parking space, which is why the importance of b should also not be too low.

prior into the attention mechanism can guide the network toward more interpretable and human-aligned predictions.

Interaction knowledge integration into deep learning has rarely been attempted in trajectory prediction [13]. To the best of our knowledge, no paper has integrated an interaction prior into the attention mechanism for mixed vehicle, pedestrian, and cyclist trajectory prediction. The only related work integrating an interaction prior into the attention mechanism is SKGACN [39]. However, SKGACN is limited to pedestrian trajectory prediction only and does not argue why they chose their multiply-and-renormalize approach. Hence, in Section 2.2.1, we have examined methods of prior knowledge integration into the attention mechanism in other fields of deep learning. We choose two promising integration methods to compare the effect of the integration method on the interpretability and alignment of the prediction. We call these two approaches multiply-and-renormalize and gating-and-loss.

Multiply-and-Renormalize

The multiply-and-renormalize approach multiplies the network's predicted attention scores with the prior interaction importance scores. It re-normalizes the product by dividing by the sum of the product's scores. We call the resulting attention scores that we get from applying the prior integration method the *combined attention scores*. The Transformer layer then uses the combined attention scores instead of its predicted attention scores for its further computations. Mathematically, the combined attention score $\alpha_{ij}^{\text{combined}}$ for an agent i and its neighbor j is defined as

$$\alpha_{ij}^{\text{combined}} = \frac{\alpha_{ij}^{\text{predicted}} \cdot \beta_{ij}}{\sum_k \alpha_{ik}^{\text{predicted}} \cdot \beta_{ik}}, \qquad [3.11]$$

where $\alpha_{ij}^{\text{predicted}} \in [0,1]$ represents the predicted attention score of the network for agent i and its neighbor j, and $\beta_{ij} \in [0,1]$ is the prior's interaction importance score for the same agent pair. The re-normalizing denominator ensures that the combined attention scores sum to 1.

We took this integration method from SKGACN [39] who were originally inspired by BERT-SIM [65], as discussed in Section 2.2.1.

Gating-and-Loss

The gating-and-loss approach gives the network more freedom in how to use the prior compared to multiply-and-renormalize. Gating-and-loss computes the combined attention scores from a neighbor-wise weighted sum of the network's predicted attention scores and the prior's interaction importance scores. Therefore, a separate gating MLP computes a gating value $\sigma_{ij} \in [0,1]$ per neighbor j for each agent i. The gating values control the degree to which the prior influences the predicted attention scores. Mathematically, the combined attention score $\alpha_{ij}^{\text{combined}}$ for agent i and its neighbor j is defined as

$$\hat{\alpha}_{ij}^{\text{combined}} = \sigma_{ij} \cdot \alpha_{ij}^{\text{predicted}} + (1 - \sigma_{ij}) \cdot \beta_{ij}, \tag{3.12}$$

$$\alpha_{ij}^{\text{combined}} = \frac{\hat{\alpha}_{ij}^{\text{combined}}}{\sum_k \hat{\alpha}_{ik}^{\text{combined}}}, \tag{3.13}$$

where $\alpha_{ij}^{\text{predicted}} \in [0,1]$ represents the predicted attention score of the network for agent i and its neighbor j, and $\beta_{ij} \in [0,1]$ is the prior's interaction importance score for the same agent pair. Here again, we re-normalize the combined attention scores.

The gating MLP computes the gating values σ_i for focal agent i based on the focal agent embedding u_i, its K-nearest neighbors' embeddings u_j, the network's predicted attention scores $\alpha_i^{\text{predicted}}$ and the prior's interaction importance scores β_i as

$$\sigma_i = \text{sigmoid}\left(\mathbf{W}^{\text{gating}} \cdot \text{concat}\left(u_i, \text{concat}_{j \in J_i}(u_j), \alpha_i^{\text{predicted}}, \beta_i\right) + \mathbf{b}^{\text{gating}}\right) \in [0,1]^{|J_i|}. \tag{3.14}$$

Here, $J_i \subseteq \{1, \ldots, N_{AG}\}$ are the K-nearest neighbors of i, $\alpha_i^{\text{predicted}} \in [0,1]^{|J_i|}$ are the predicted attention scores of agent i for all neighbors $j \in J_i$, $\beta_i \in [0,1]^{|J_i|}$ are the interaction importance scores of all neighbors $j \in J_i$ for a fixed focal agent i, $\mathbf{W}^{\text{gating}} \in \mathbb{R}^{|J_i| \times \text{hidden}}$ is the weight matrix of the gating MLP, and $\mathbf{b}^{\text{gating}} \in \mathbb{R}^{|J_i|}$ is the bias vector of the gating MLP. This formulation allows the network to decide the prior's influence online per scene and agent.

In addition to the gating mechanism, we introduce an auxiliary loss component. As discussed in Section 2.2.1, without such a loss, the network could bypass the prior by always predicting a gating value of $\sigma_{ij} = 1$. Thus, we introduce an additional loss component that minimizes the distance between combined attention scores and the prior's scores. As both scores can be understood as probability distributions, the Kullback–Leibler divergence is a natural choice for this loss. The loss term encourages the combined attention scores to be close to the prior's scores when averaged across all attention heads. This still permits variation in individual heads while generally adhering to the guidance of the prior. The additional loss term $\mathscr{L}_{\text{attn}}$ for the prediction

of focal agent i is computed as

$$\mathscr{L}_{\text{attn}} = \frac{1}{|J_i|} D_{KL}\left(\beta_i \,\middle\|\, \frac{1}{H}\sum_{h=1}^{H} \alpha_i^{h,\text{combined}}\right), \tag{3.15}$$

$$D_{KL}(\beta_i \,||\, \alpha_i) = \sum_{j \in J_i} \beta_{ij} \cdot log\left(\frac{\beta_{ij}}{\alpha_{ij}}\right), \tag{3.16}$$

where H is the number of attention heads, $\alpha_i^{h,\text{combined}} \in [0,1]^{|J_i|}$ are the combined attention scores of agent i for all neighbors $j \in J_i$ in the h-th attention head, and D_{KL} is the Kullback–Leibler divergence.

This integration method was inspired by AffectiveAttention's Attentional Feature-based Gating [40] and Attentive Mixture of Experts' [53] success in interpreting feature importance with the gating approach.

We selected multiply-and-renormalize and gating-and-loss for their contrasting characteristics in terms of complexity and flexibility. On the one hand, The multiply-and-renormalize method is a simple, easy-to-integrate solution. Since SKGACN is, to the best of our knowledge, the only existing work that applies prior integration to attention mechanisms for trajectory prediction, it is a natural choice to examine their method in our context before introducing new methods.

On the other hand, gating-and-loss offers a more sophisticated approach, better aligned with the intuition of providing the network with guidance that it is encouraged but not strictly required to follow. The additional loss function gently pushes the network to incorporate the prior's interaction scores while maintaining the flexibility to learn more complex behaviors across different attention heads. This balance makes gating-and-loss an appealing alternative to the more rigid multiply-and-renormalize approach.

3.4 Kinematically Feasible Decoder

In the second half of our network, the decoder, we aim to make the prediction kinematically feasible. This is essential for making trajectory predictions that align with the physical realities of motion, which is critical for safe autonomous driving. Without kinematic constraints, predicted trajectories can involve physically impossible movements, which would not occur in the real world. By enforcing kinematic feasibility, we not only improve the realism of predicted trajectories but also bring them closer to human-like predictions, thereby enhancing trust and safety in autonomous systems.

In this section, we detail how we propose to make the predictions of our network kinematically feasible. Following the insights from the state-of-the-art in Section 2.2.2, we aim to achieve this through action-space prediction using kinematic models. Accordingly, we first define the kinematic models that formulate our prior knowledge about the agents' physics of motion. Second, we introduce the kinematic layer that integrates the kinematic models into the network by applying them to update the agents' state based on predicted control inputs.

3.4.1 Kinematic Priors

To ensure that our predicted trajectories adhere to physical constraints, we require kinematic models that formalize the physics of agent motion. These models define how sequences of control inputs, such as acceleration and steering commands, influence the state of an agent over time, including its position, velocity, and orientation. We have introduced commonly used kinematic models in Section 2.2.2.

Vehicle Kinematic Model

For vehicles, we adopt the unicycle model as our kinematic model. Although the bicycle model has been more prevalent in previous trajectory prediction research, Trajectron++ [48] demonstrated that the Unicycle Model is sufficiently accurate for trajectory prediction tasks. The Unicycle Model is an attractive choice over the bicycle model due to two central benefits. First, it is free of agent-specific parameters: it requires no parameter estimation, such as wheelbase length or vehicle dimensions, thereby simplifying implementation and reducing potential sources of error [48]. Second, it is computationally more efficient: with simpler equations and no online parameter estimation needed, it demands less computational resources, which is beneficial for real-time applications [48].

For a mathematical definition, we adopt the dynamically extended unicycle model from Trajectron++ [48]. Unlike simpler unicycle models, the dynamically extended version uses acceleration a and heading rate ω as control inputs, which correspond to real-world vehicle control inputs like the accelerator pedal and steering wheel. The unicycle's continuous-time dynamics equation defines how the model connects the agent state $s = [x, y, \theta, ||v||]^T$ with the control inputs $c = [\omega, a]^T$ as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{||v||} \end{bmatrix} = \begin{bmatrix} ||v|| \cos(\theta) \\ ||v|| \sin(\theta) \\ \omega \\ a \end{bmatrix}. \tag{3.17}$$

Here, $[x, y]^T$ represents the 2D position of the agent, θ is the agent heading measured as the angle to the x-axis, and $||v||$ is the speed of the agent. The control inputs are the change of the heading ω, the so-called heading rate, and the acceleration a of the agent in the heading direction.

In discrete time, with a sampling interval Δt, we assume that the controls are piecewise constant between time steps. We use the following discretized state update equation from Trajectron++ [48]:

$$\begin{bmatrix} x^{t+1} \\ y^{t+1} \\ \theta^{t+1} \\ ||v^{t+1}|| \end{bmatrix} = \begin{bmatrix} x^t \\ y^t \\ \theta^t \\ ||v^t|| \end{bmatrix} + \begin{bmatrix} ||v^t|| \cdot D_S^t + a^t \sin(\theta^t + \omega^t \Delta t) \frac{\Delta t}{\omega^t} + \frac{a^t}{\omega^t} \cdot D_C^t \\ -||v^t|| \cdot D_C^t - a^t \cos(\theta^t + \omega^t \Delta t) \frac{\Delta t}{\omega^t} + \frac{a^t}{\omega^t} \cdot D_S^t \\ \omega^t \Delta t \\ a^t \Delta t \end{bmatrix}, \tag{3.18}$$

$$D_S^t = \frac{\sin(\theta^t + \omega^t \Delta t) - \sin(\theta^t)}{\omega^t}, \tag{3.19}$$

$$D_C^t = \frac{\cos(\theta^t + \omega^t \Delta t) - \cos(\theta^t)}{\omega^t}. \quad [3.20]$$

For numerical stability and to avoid singularities when $\omega \to 0$, the following alternative formulation is used for small heading rates of $|\omega| \leq \varepsilon = 10^{-3}$ [48]:

$$\begin{bmatrix} x^{t+1} \\ y^{t+1} \\ \theta^{t+1} \\ ||v^{t+1}|| \end{bmatrix} = \begin{bmatrix} x^t \\ y^t \\ \theta^t \\ ||v^t|| \end{bmatrix} + \begin{bmatrix} ||v^t|| \cos(\theta^t)\Delta t + 0.5 a^t \cos(\theta^t)(\Delta t)^2 \\ ||v^t|| \sin(\theta^t)\Delta t + 0.5 a^t \sin(\theta^t)(\Delta t)^2 \\ 0 \\ a^t \Delta t \end{bmatrix}. \quad [3.21]$$

Pedestrian Kinematic Models

Selecting an appropriate kinematic model for pedestrians is less straightforward than for vehicles, as the literature does not converge on a standard model. Pedestrians have mostly been treated as a secondary consideration in kinematically feasible trajectory prediction research [11, 36, 21]. To the best of our knowledge, no work except for Trajectron++ [48], explicitly focused on kinematically feasible pedestrian trajectory prediction. To address this gap, we compare kinematic models typically used for pedestrians in trajectory prediction literature. Our study of the state-of-the-art in Section 2.2.2 showed that the unicycle model and the single integrator model are the most promising.

However, both models come with significant limitations when applied to pedestrian motion. The single integrator model, while offering flexibility, is often too unconstrained: it permits unrealistic accelerations, allowing pedestrians to instantaneously transition from a standstill to full speed, which is not physically possible. On the other hand, the unicycle model imposes overly strict constraints by enforcing non-holonomic motion, where pedestrians cannot make abrupt directional changes or move sideways when walking forward. Therefore, we propose the double integrator model as a new alternative that strikes a balance between the two other models. Our goal is to compare these three models to identify which best captures pedestrian motion while ensuring kinematic feasibility.

Single Integrator Model It controls the agent state $s = [x, y]^T$ with control input $c = [v_x, v_y]^T$. The state update equation is

$$\begin{bmatrix} x^{t+1} \\ y^{t+1} \end{bmatrix} = \begin{bmatrix} x^t \\ y^t \end{bmatrix} + \begin{bmatrix} v_x^t \\ v_y^t \end{bmatrix} \Delta t, \quad [3.22]$$

where $[x^t, y^t]^T \in \mathbb{R}^2$ is the agent's 2D position at time t, $[v_x^t, v_y^t]^T \in \mathbb{R}^2$ is the agent's velocity vector at time t.

Unicycle Model It controls the agent state $s = [x, y, \theta, ||v||]^T$ with control input $c = [\omega, a]^T$. We use the same formulation for pedestrians as for vehicles in Equations (3.18) to (3.21).

Double Integrator Model It controls the agent state $s = [x, y, v_x, v_y]^T$ with control input $c = [a_x, a_y]^T$. The state update equation is

$$\begin{bmatrix} x^{t+1} \\ y^{t+1} \\ v_x^{t+1} \\ v_y^{t+1} \end{bmatrix} = \begin{bmatrix} x^t \\ y^t \\ v_x^t \\ v_y^t \end{bmatrix} + \begin{bmatrix} v_x^t \\ v_y^t \\ a_x^t \\ a_y^t \end{bmatrix} \Delta t + \frac{1}{2} \begin{bmatrix} a_x^t \\ a_y^t \\ 0 \\ 0 \end{bmatrix} (\Delta t)^2, \quad [3.23]$$

where $[x^t, y^t]^T \in \mathbb{R}^2$ is the agent's 2D position at time t, $[v_x^t, v_y^t]^T \in \mathbb{R}^2$ is the agent's velocity vector at time t, $[a_x^t, a_y^t]^T \in \mathbb{R}^2$ are the agent's acceleration along x- and y-axis at time t.

Note that this formulation of the acceleration differs from the one in the unicycle model. The double integrator allows acceleration in any direction while the unicycle only allows acceleration in the current direction of motion.

The double integrator model serves as a middle ground between the flexibility of the single integrator and the constraints of the unicycle. Compared to the single integrator, it enhances physical feasibility by constraining acceleration, preventing unrealistic sudden changes in speed. For example, the single integrator allows a pedestrian to instantaneously accelerate from standing still to full-speed sprinting. Compared to the unicycle, it allows behavior prohibited by the unicycle, like sudden changes in direction. For example, the unicycle cannot model behaviors like a pedestrian quickly turning around or moving sideways to avoid an obstacle.

3.4.2 Kinematic Prior Integration via Action-Space Prediction

To integrate the chosen kinematic models into our decoder, we employ an action-space prediction approach. Instead of predicting future trajectories directly, the network predicts sequences of control inputs, also referred to by others as actions, which are then propagated through the respective kinematic models to generate the future trajectories. This method ensures that the predicted trajectories are inherently kinematically feasible.

Kinematic Layer

In the mathematical definition of the kinematic layer, we follow DKM [11] because it is a well-established choice in the state-of-the-art [11, 48]. The network predicts the control inputs in one shot while it applies the kinematic models recurrently. Thus, for each agent i and mode k, the network predicts control inputs over the prediction horizon T_s:

$$C_{i_k} = [c_{i_k}^{T_o+1}, c_{i_k}^{T_o+2}, \ldots, c_{i_k}^{T_o+T_s}], \quad [3.24]$$

where $c_{i_k}^t \in \mathbb{R}^{\text{control-dim}_{\text{kinematic}_i}}$ represents the predicted control inputs at time t, specific to the kinematic model of the agent's class with $\text{control-dim}_{\text{kinematic}_i}$ as the dimension of the kinematic model's control inputs.

The kinematic layer then computes the future states by recursively applying the control inputs starting from the last observed state $s_i^{T_o}$:

$$\hat{s}_{i_k}^{t+1} = f_{\text{kinematic}}(\hat{s}_{i_k}^{t}, c_{i_k}^{t+1}) \quad \text{for } t \geq T_o, \tag{3.25}$$

$$\forall k \in K_{AC}.\ \hat{s}_{i_k}^{T_o} = s_i^{T_o}, \tag{3.26}$$

where $f_{\text{kinematic}}$ is the state update equation defined by the agent's kinematic model in Equations (3.18), (3.22) and (3.23). $\hat{s}_{i_k}^{t} \in \mathbb{R}^{\text{state-dim}_{\text{kinematic}_i}}$ is the predicted state of agent i and mode k at time t, specific to the kinematic model of the agent's class. Here, $\text{state-dim}_{\text{kinematic}_i}$ is the dimension of the kinematic model's agent state.

Finally, the kinematic layer extracts the predicted trajectories $\hat{Q}_{i_k}$ from the predicted future states. As all of our kinematic models' agent states contain the agent position in 2D Cartesian coordinates, the extraction operation is as easy as discarding all other state information besides the 2D Cartesian coordinates:

$$\hat{Q}_{i_k} = [\hat{q}_{i_k}^{T_o+1}, \hat{q}_{i_k}^{T_o+2}, \ldots, \hat{q}_{i_k}^{T_o+T_s}] = f_{\text{extract}}\left([\hat{s}_{i_k}^{T_o+1}, \hat{s}_{i_k}^{T_o+2}, \ldots, \hat{s}_{i_k}^{T_o+T_s}]\right), \tag{3.27}$$

$$f_{\text{extract}}(\hat{s}_{i_k}^{t}) = f_{\text{extract}}(x_{i_k}^{t}, y_{i_k}^{t}, \ldots) = (x_{i_k}^{t}, y_{i_k}^{t}) = \hat{q}_{i_k}^{t}. \tag{3.28}$$

Here, f_{extract} is the extraction operation that is applied element-wise to the predicted future states, $(x_{i_k}^{t}, y_{i_k}^{t})$ is the agent position for agent i and mode k at time t, and $\hat{q}_{i_k}^{t}$ is the t-th step of predicted trajectory $\hat{Q}_{i_k}$.

Control Input Constraints

To guarantee that the kinematic layer produces only kinematically feasible trajectories, we impose limits on the control inputs C_{i_k} predicted by the network. We enforce acceleration and heading rate limits by applying a tanh function on the MLP prediction heads' outputs and rescaling to the range between the limits. For the unicycle's and double integrator's velocity limit, we clip the acceleration of each step so that the maximum velocity cannot be exceeded. For the single integrator, which does not provide any accelerations, we use tanh directly on the prediction heads' predicted velocities just as for acceleration and heading rate limits. We derive the concrete limits from physical constraints and empirical observations of vehicle and pedestrian behaviors.

Vehicles For vehicles, we constrain the acceleration and heading rate to realistic ranges. The acceleration a^t is limited to the interval $[-8, 8]$ m/s^2, which is widely accepted in the literature as a reasonable approximation of a vehicle's physical capabilities [21, 11]. This range accounts for typical maximum deceleration (braking) and acceleration rates that most vehicles can achieve under normal driving conditions.

Limiting the heading rate ω^t is more complex because vehicles have a minimum turning radius depending on their speed. Following the approach of PTNet [21], we limit the curvature

κ^t instead of directly constraining the heading rate. The curvature κ^t relates to the heading rate ω^t and speed $||v^t||$ as

$$\omega^t = \kappa^t \cdot ||v^t||. \tag{3.29}$$

By constraining the curvature to a feasible range, we limit the heading rate in a way that reflects the vehicle's physical turning capabilities at different speeds. For example, driving the same turn at a higher speed results in a higher heading rate. As the limit for the curvature κ, we use PTNet's $[-0.3, 0.3]$ 1/m [21].

Pedestrians Defining control input limits for pedestrians is less straightforward than for vehicles. As, to the best of our knowledge, no previous work has guaranteed kinematic feasibility for pedestrian trajectory prediction, there are no limits to refer to from previous work. Trajectron++ came close with pedestrian kinematic models but left the control inputs unconstrained [48].

To establish reasonable limits, we refer to studies on human walking and acceleration behavior. Studies measuring human walking and acceleration behavior suggest that the maximum acceleration for pedestrians ranges between 8 m/s^2 and 10 m/s^2 [18, 66]. For comparison, Usain Bolt reached a peak acceleration of approximately 9.5 m/s^2 during his world-record 100-meter sprint at the 2009 IAAF world championships [23]. Considering that such acceleration represents the upper limit of human capability, we set the acceleration limits for pedestrians to $a \in [-8, 8]$ m/s^2, accommodating both rapid deceleration and acceleration in typical scenarios.

Similarly, we limit the speed for pedestrians. Here, we refer to Usain Bolt's top speed during the same sprint of approximately 12 m/s. Again, considering this the upper limit of human capability, we expect a pedestrian to have a speed $||v|| \in [0, 10]$ m/s [23].

To validate and refine these limits, we perform an empirical analysis in the evaluation chapter, specifically in Section 5.3.1, measuring the maximum acceleration, velocity, and curvature observed in the dataset's ground truth trajectories.

Class-Specific Prediction Heads

As we have introduced class-specific kinematic models in the kinematic layer, it makes sense to also introduce class-specific instances of the MLP prediction head. Each prediction head then predicts the control inputs for exactly one kinematic model. Hence, each prediction head instance can be seen as an expert for one class. Consequently, the overall control input prediction becomes a mixture of experts. This design choice is motivated by the same reasoning as the class-specific agent-to-agent Transformer layers in Section 3.3.1: A single feature indicating the class is not enough for a single prediction head to learn distinct kinematic model behaviors.

Formally, for the focal agent i and each mode k, the prediction head $l_{\text{pred-head}}$ generates the control inputs C_{i_k} based on the agent-anchor embedding u_{i_k}:

$$C_{i_k} = l_{\text{pred-head}}(u_{i_k}) = \begin{cases} l_{\text{pred-head}_{veh}}(u_{i_k}) & \text{if agent\_class}(i) = veh, \\ l_{\text{pred-head}_{ped}}(u_{i_k}) & \text{if agent\_class}(i) = ped, \\ l_{\text{pred-head}_{cyc}}(u_{i_k}) & \text{if agent\_class}(i) = cyc. \end{cases} \tag{3.30}$$

Here, $l_{\text{pred-head}_{class}}$ are three separate MLP instances that are identical in their computational process but have different instances of weights. Each instance predicts control inputs for the kinematic model of its agent class.

4 Experimental Setup

In this chapter, we detail the experimental setup used to evaluate the performance of our proposed method for enhancing the alignment of trajectory predictions in terms of feasibility and interpretability. We describe the setup for training our network on a GPU cluster to evaluate its performance in the evaluations chapter.

The chapter is organized as follows. First, we provide a detailed overview of the training data, including the dataset used and its preprocessing. Next, we outline the training configuration, describing the loss functions, hyperparameters, interaction prior calibration, and the networks we train for comparison. Additionally, we briefly report the training time and training environment. Finally, we introduce the metrics used in the evaluation chapter to assess the network's effectiveness, including how we measure interpretability and feasibility.

4.1 Training Data

The training process requires the dataset to provide several key data inputs, as defined in Section 3.1: the focal agent's state observations S_i, the ground-truth future trajectory of the focal agent Q_i, the context observations S and M of the surrounding agents and map, and the agent class labels, denoted as agent_class(j) for $j \in \{1, \ldots, N_{AG}\}$.

We use the Argoverse 2 Motion Forecasting Dataset (Argoverse 2) [63], a widely respected benchmark in trajectory prediction. The dataset includes 250,000 urban traffic scenes recorded from the perspective of Argo AI's test vehicles. The data was collected during 763 driving hours in six U.S. cities, spanning a total of 2,220 kilometers of unique roadways. These recordings were captured using high-resolution cameras and LiDAR sensors, providing detailed 2D position, velocity, and orientation data per time step for each agent.

The dataset is divided into three splits: 200,000 samples for training, 25,000 samples for validation, and 25,000 for testing. Each scene focuses on one focal agent, with the goal of predicting this agent's future trajectory based on a five-second history of observations. The dataset provides ground-truth trajectories for a six-second future. The sampling interval between time steps is $\Delta t = 0.1$ seconds, providing $T_o = 50$ observation steps and $T_h = 60$ prediction steps. This setup supports accurate trajectory prediction over short-term windows. The ranking metric on this dataset is the brier-minFDE metric. As shown in Table 4.1, the number of samples per focal agent class is heavily biased toward vehicles.

We selected Argoverse 2 for several reasons. First, it is one of two datasets already integrated with our backbone network, HPTR [69]. We chose Argoverse 2 over HPTR's other dataset, the Waymo Open Motion Dataset [15], because preliminary analysis indicated that Waymo's dataset contains more tracking errors, which could adversely affect the training of our kinematic layer.

Table 4.1: Number of scenes per class of the focal agent in that scene for the Argoverse 2 dataset [63].

	Vehicles	Pedestrians	Cyclists
Training Split	183,000	13,000	3,700
Validation Split	23,000	1,560	440

Moreover, Argoverse 2 is one of the most recent and widely used benchmarks for trajectory prediction [51]. It is recognized as a challenging and diverse dataset, making it a valuable benchmark for testing complex networks like ours.

Provided the data from the dataset, preprocessing is crucial to convert it into a format suitable for training. The Argoverse 2 dataset has already processed its sensor data into 2D positions, velocities, and orientations. From this data, we apply HPTR's preprocessing, which involves picking a random origin position and heading for the coordinate system of each scene in the training data. Note that HPTR's pairwise-relative pose representation then transforms all poses online. Consequently, during prediction, the coordinate system for each scene is set such that the origin aligns with the focal agent's last observed position $r_i^{T_o}$. The x-axis is oriented along the focal agent's heading in that state. This gives us local poses centered on the focal agent for each scene, which benefits pose invariance and thus simplifies prediction.

4.2 Training Configuration

The training configuration for our network is carefully designed to ensure that the network effectively learns to predict. This section provides an overview of the loss functions and hyperparameters used in training our network.

4.2.1 Loss

We employ a combination of position loss, confidence loss, and additional loss functions for attention-based prior integration.

Most importantly, a position loss function optimizes the accuracy of the predicted future trajectories. For this, following the position loss definition of LaneGCN [37], we use the Huber loss. It is a robust regression loss commonly used in trajectory prediction tasks. The Huber loss smooths out large errors, making it more robust than the L2 loss, which can be sensitive to outliers. The position loss is computed for the mode $\hat{k} \in \{1, \dots, K_{AC}\}$ whose trajectory $\hat{Q}_{i_{\hat{k}}}$ has the lowest Average Displacement Error compared to the ground truth trajectory Q_i. The formulation of the position loss $\mathscr{L}_{\text{pos}}$ [37] is

$$\mathscr{L}_{\text{pos}} = \frac{1}{T_h} \sum_{t=T_o+1}^{T_o+T_h} L_\delta(\hat{q}^t_{x_{i_{\hat{k}}}} - q^t_{x_i}) + L_\delta(\hat{q}^t_{y_{i_{\hat{k}}}} - q^t_{y_i}), \tag{4.1}$$

where $\hat{q}^t_{i_{\hat{k}}} = [\hat{q}^t_{x_{i_{\hat{k}}}}, \hat{q}^t_{y_{i_{\hat{k}}}}]^T$ and $\hat{q}^t_i = [\hat{q}^t_{x_i}, \hat{q}^t_{y_i}]^T$ respectively represent the 2D position of the focal agent at time t on the predicted and ground-truth trajectory. L_δ is the Huber loss function with $\delta = 1$.

In contrast to HPTR, we opted not to use their negative log-likelihood loss, as it would require propagating probability distributions through the kinematic layer. That is beyond the scope of this thesis but would be an interesting avenue for future work. Additionally, we considered using the L2 loss from DKM [11], but decided on the Huber loss due to its improved robustness to outliers.

Alongside the position loss, a cross-entropy loss trains the network's confidence head. As the ground-truth distribution, we want a predicted confidence of 1 for mode $\hat{k}$ and 0 for all other modes. Here, mode $\hat{k}$ is the mode with the lowest average displacement compared to the ground-truth trajectory. The confidence loss $\mathscr{L}_{\text{conf}}$ is computed as follows [69]:

$$\mathscr{L}_{\text{conf}} = -\log\left(\frac{\exp(\hat{p}_{i_{\hat{k}}})}{\sum_{k=1}^{K_{AC}}\exp(\hat{p}_{i_k})}\right), \tag{4.2}$$

$$\hat{k} = \underset{k\in\{1,\dots,K_{AC}\}}{\operatorname{argmin}} \sum_{t=T_o+1}^{T_o+T_h} \|\hat{Q}_{i_k}^t - Q_i^t\|_2, \tag{4.3}$$

where $\hat{p}_{i_k}$ is the predicted confidence of mode k and $\exp(x) = e^x$ is the exponential function.

Opposed to HPTR, we chose not to incorporate auxiliary losses for speed, velocity, and yaw prediction. While HPTR employs these auxiliary losses to help the network learn additional agent features, we argue that our kinematic layer already forces the network to understand these concepts. Moreover, velocity and yaw data from the dataset do not always align well with the trajectory defined by the agent's position. This results from noise and tracking errors in the dataset's captured agent states per trajectory step. Consequently, employing the auxiliary losses also means the network will probably learn those inconsistencies from the ground truth data. Similarly, in HPTR, the predicted velocity and yaw are in no way constrained to fit the predicted trajectory. Thus, we question the helpfulness of these auxiliary losses. HPTR can potentially learn completely separate behaviors in the predicted trajectory, velocity, and yaw. For example, the trajectory could go forward with a constant velocity, the velocity could be 0 in all steps, and the yaw could turn backward. Further supporting our decision, other state-of-the-art models, like Wayformer [42] and MTR [54], do not use auxiliary losses for these features either.

For networks trained with prior integration via the gating-and-loss method, we introduce the additional attention loss $\mathscr{L}_{\text{attn}}$, as defined in Section 3.3.3. This loss ensures that the agent-to-agent attention learned by the network aligns with the interaction prior. Finally, we form the weighted sum to calculate the loss $\mathscr{L}$ per scene, which we use for training:

$$\mathscr{L} = \tau_{\mathscr{L}_{\text{pos}}}\mathscr{L}_{\text{pos}} + \tau_{\mathscr{L}_{\text{conf}}}\mathscr{L}_{\text{conf}} + \tau_{\mathscr{L}_{\text{attn}}}\mathscr{L}_{\text{attn}}, \tag{4.4}$$

where $\tau_{\mathscr{L}_{\text{pos}}}, \tau_{\mathscr{L}_{\text{conf}}}, \tau_{\mathscr{L}_{\text{attn}}} \in \mathbb{R}$ are the sum weights.

4.2.2 Hyperparameters

When setting up the hyperparameters, we followed HPTR's configuration wherever possible to ensure comparability with existing results. As HPTR, our network is trained with a batch size of 12 and a learning rate of 10^{-4}, halved every 25 training epochs using a learning rate scheduler. After 75% of the training epochs, we apply stochastic weight averaging with the learning rate fixed to the value determined by the scheduler at that point. The Adam optimizer is used for all training, with ReLU activations applied throughout the network.

The network architecture maintains HPTR's multi-head attention with 4 heads and a pre-layer normalization Transformer architecture with a dropout rate of 0.1. The feed-forward hidden dimension of the Transformers is set to 1024, and the base frequency ω of the sinusoidal positional encoding is set to 1000. We train a single network for all agents with class-specific layers as introduced in Sections 3.3.1 and 3.4.2. For map-to-map and agent-to-agent attention, we use KNARPE, introduced in Section 3.2.2, with $K = 36$ nearest neighbors out of $N_{MP} = 1024$ map elements and $N_{AG} = 64$ agents per scene [69]. We use $K = 144$ for map-to-agent attention and $K = 360$ for all-to-anchor attention.

For the loss, we weigh the attention component with a coefficient of 0.1. Initial training runs showed that an unweighted sum of all losses caused the attention loss to diminish quickly to near-zero values. For the other losses, we adopt HPTR's approach of using an unweighted sum. Hence, $\tau_{\mathscr{L}_{\text{pos}}} = 1, \tau_{\mathscr{L}_{\text{conf}}} = 1, \tau_{\mathscr{L}_{\text{attn}}} = 0.1$ are our loss weights.

Considering the model size, our modifications led to a slight increase in parameters. First, removing the traffic light encoder, which is irrelevant for Argoverse 2, because it does not contain traffic light data, reduces the parameter count by 2.8 million. Then, our interpretable interaction encoder adds 970k parameters, which triples to 2.8 million when using class-specific interaction layers for the three agent classes vehicles, pedestrians, and cyclists. The gating-and-loss prior combination method introduces an additional 350k parameters, while the multiply-and-renormalize method does not add parameters. The decoder grows by 1.5 million parameters through class-specific prediction heads for the respective kinematic models. The kinematic layer does not add any parameters as it works fully deterministically. On balance, when all modifications are enabled, this leads us to a final network size of approximately 17.7 million parameters compared to HPTR's 15.2 million.

4.2.3 Interaction Prior Setup

The interaction priors we introduced in Section 3.3.2 require some calibration parameters. We calibrate them based on human intuition about which agents are significant in a traffic scene. This manual tuning approach was necessary because systematically calibrating these parameters via grid-search and ablation studies would have exceeded our resources. Instead, we tuned the parameters according to our understanding of interaction importance.

We apply a softmax temperature of 20 for SKGACN and 12 for DG-SFM. These values were chosen to balance the focus on the most relevant agents while still giving some weight to semi-

relevant agents. Further, the DG-SFM prior requires the SFM social repulsive parameters plus the three additional parameters we introduced. We set these parameters to the following values:

- $V^0_{\text{sfm}} = 1$, controlling the baseline strength of the interaction potential,
- $\tau_{\text{sfm}} = 20$, controlling the strength of the interaction potential's exponential decay,
- $N_{\text{DG}} = 10$, which gives us a discretization step size of 1s due to the $\Delta t = 0.1$s interval,
- $\tau_{\text{dg-sfm}_{\text{sum}}} = 0.15$, the weight for summing the two DG-SFM components,
- and $\tau_{\text{dg-sfm}_{\text{standing}}} = 0.25$, the penalty factor for stationary agents.

4.2.4 Model Variants

To evaluate the effectiveness of our proposed layers and priors, we trained and compared several model variants. For a fair comparison, unless stated otherwise, all model variants are trained using the loss function defined in Section 4.2.1 and trained for 15 epochs on the dataset.

HPTR$_b$ This variant is our baseline. It uses the original HPTR architecture and code from the open-source implementation without any of the modifications we introduced. However, for comparability, we make two changes compared to HPTR as it is defined in the paper [69]. First, we remove HPTR's traffic light encoder layers, since there are no traffic lights in the Argoverse 2 dataset. Second, for comparability, we train the network with our loss functions, as defined in Section 4.2.1. HPTR$_b$ provides a baseline to compare our layers and modifications against, allowing us to assess the impact of our changes.

HPTR$_i$ This variant extends HPTR$_b$ by adding our agent-to-agent interaction layer, as defined in Section 3.3.1. However, it employs only a single shared instance of the agent-to-agent interaction layer for all agent classes. HPTR$_i$ helps us evaluate the effect of adding an agent-to-agent interaction layer but without per-class specialization.

HPTR$_{ci}$ This variant extends HPTR$_i$ by introducing the class-specific instances of the agent-to-agent interaction layer, instantiating a separate layer for vehicles, pedestrians, and cyclists, as defined in Section 3.3.1. HPTR$_{ci}$ allows us to measure the benefit of per-class specialization in the interaction layers.

HPTR$_{i+p}$ Building on HPTR$_i$, this variant incorporates our prior attention integration mechanism into the agent-to-agent attention layer, as defined in Section 3.3.3. We denote the prior integration method used in the index of p, i.e., we can either choose HPTR$_{i+p_{\text{MnR}}}$ for multiply-and-renormalize integration or HPTR$_{i+p_{\text{GnL}}}$ for gating-and-loss integration. Comparing HPTR$_{i+p}$ with HPTR$_i$ allows us to see how adding an interaction prior impacts the network's performance independently of the class-specific interaction layers.

HPTR$_k$ HPTR$_k$ introduces our kinematic layer to HPTR$_b$, enforcing kinematic feasibility for both vehicles and pedestrians, as defined in Section 3.4.2. Note that the kinematic layer does contain class-specific prediction heads. Hence, HPTR$_i$ is not the only variant with class-specific layers. We use this variant to study the isolated impact of enforcing kinematic feasibility on trajectory prediction accuracy and feasibility metrics.

HPTR$_{ci+k}$ This variant combines both our class-specific agent-to-agent interaction layers from HPTR$_{ci}$ and our kinematic layer from HPTR$_k$. HPTR$_{ci+k}$ allows us to assess whether their combination yields further performance improvements.

HPTR$_{ci+p+k}$ Finally, this variant includes all the modifications we propose: class-specific agent-to-agent interaction layers, prior attention integration, and kinematic layer. HPTR$_{ci+p+k}$ serves as the full network and allows us to measure the cumulative impact of all our contributions.

4.3 Training Time and Environment

Like HPTR's training for the Waymo Open Motion Dataset, we trained our networks with 15 epochs on the Argoverse 2 training split for our ablation models in Sections 5.1 to 5.3. Only for reporting our final network's performance in Section 5.4, we trained for 30 dataset epochs. All training experiments were conducted on an NVIDIA RTX6000 GPU with 48GB of VRAM. Training our final model HPTR$_{ci+p+k}$ for 30 epochs took 12 days.

4.4 Metrics

This section details the metrics used to evaluate our proposed method for trajectory prediction. We employ both standard metrics commonly used in the field of trajectory prediction and specialized metrics tailored to evaluating the model's interpretability and kinematic feasibility. Additionally, we consider class-specific metrics to evaluate model performance on the agent classes of vehicles, pedestrians, and cyclists.

4.4.1 Standard Metrics

To evaluate the accuracy of predicted trajectories, we use standard metrics such as *minimum Average Displacement Error (minADE)*, *minimum Final Displacement Error (minFDE)*, *brier-minFDE*, *Miss Rate (MR)*, and *mean Average Precision (mAP)*. These metrics are widely employed in trajectory prediction literature and benchmarks, including Argoverse 2 [63] and the Waymo Open Motion Dataset [15]. Since Argoverse 2 does not define these metrics in its paper, we adopt the definitions from Schuetz et al. [51].

The **minimum Average Displacement Error (minADE)** measures the average Euclidean distance between the ground-truth trajectory and the trajectory for mode k that is closest to it over

the prediction horizon T_h. Formally, for predicted trajectories $\hat{Q}_{i_k}$ and ground-truth trajectories Q_{i_k} for focal agent i, the minADE is defined as

$$\text{minADE}_{K_{AC}}(i) = \frac{1}{T_h} \min_{k \in \{1,\dots,K_{AC}\}} \sum_{t=T_o+1}^{T_o+T_h} \|\hat{Q}_{i_k}^t - Q_i^t\|_2, \tag{4.5}$$

where T_o is the last observation time, T_h is the number of predicted time steps, and $\|\cdot\|_2$ denotes the Euclidean distance. K_{AC} indicates the number of modes out of which the one with minimum distance to the ground truth is picked.

For multiple scenes, we compute the average minADE across all scenes. Formally, given a set of scenes Ω with exactly one focal agent $i \in \Omega_{\text{focal}}$ per scene, minADE on Ω is defined as

$$\text{minADE}_{K_{AC}} = \text{minADE}_{K_{AC}}(\Omega) = \frac{1}{|\Omega_{\text{focal}}|} \sum_{i \in \Omega_{\text{focal}}} \text{minADE}_{K_{AC}}(i). \tag{4.6}$$

We apply a similar definition for minFDE and brier-minFDE to define them over a set of scenes.

The **minimum Final Displacement Error (minFDE)** only measures the distance between the final predicted position at the end of the prediction horizon and the ground-truth final position for the best mode k:

$$\text{minFDE}_{K_{AC}}(i) = \min_{k \in \{1,\dots,K_{AC}\}} \|\hat{Q}_{i_k}^{T_o+T_h} - Q_i^{T_o+T_h}\|_2. \tag{4.7}$$

The **brier-minFDE** is a probabilistic variant of minFDE that incorporates uncertainty into the evaluation. It weights the minFDE with the confidence of the mode $\tilde{k}$ that achieves the minimum final displacement error. With $\hat{p}_{i_k}$ the confidence score for predicted trajectory $\hat{Q}_{i_k}$, the brier-minFDE is defined as

$$\text{brier-minFDE}_{K_{AC}}(i) = \text{minFDE}_{K_{AC}}(i) \cdot (1 - \hat{p}_{i_{\tilde{k}}})^2, \tag{4.8}$$

$$\tilde{k} = \operatorname*{argmin}_{k \in \{1,\dots,K_{AC}\}} \|\hat{Q}_{i_k}^{T_o+T_h} - Q_i^{T_o+T_h}\|_2. \tag{4.9}$$

The **Miss Rate (MR)** evaluates the proportion of predicted trajectories where the final position deviates by more than a given threshold from the ground truth. Given a threshold of δ = 2m, the miss rate is defined as

$$\text{MR}_{K_{AC}} = \frac{1}{|\Omega_{\text{focal}}|} \sum_{i \in \Omega_{\text{focal}}} \mathbb{1}\left(\text{minFDE}_{K_{AC}}(i) > \delta\right), \tag{4.10}$$

where $\mathbb{1}$ is an indicator function that returns 1 if the given condition is true and 0 otherwise. The sum sums over all focal agents in a set of scenes Ω.

The **mean Average Precision (mAP)** is a standard metric recently introduced by the Waymo Open Motion Dataset [15]. It measures how well the predicted confidence scores $\hat{p}_{i_k}$ for the modes $k \in \{1,\dots,K_{AC}\}$ align with the ground truth distribution of multimodality. Inspired by object detection metrics, Waymo applies this metric to trajectory predictions by ranking the

predicted trajectories according to their confidence scores. For each prediction, the mode with the highest confidence that falls within the Miss Rate threshold from the ground truth's final position is considered the true positive.

For defining the mAP, a series of definitions is necessary. First, the precision at a given confidence threshold $P@K$ is defined as follows [44]:

$$P@K(i) = \frac{1}{K} \sum_{k=1}^{K} \text{relevance}(\hat{Q}_{i_{o_k}}), \tag{4.11}$$

where $\text{relevance}(\hat{Q}_{i_k}) = 1$ if $\hat{Q}_{i_k}$ is the true positive mode and 0 otherwise. $[o_1, \ldots, o_{K_{AC}}]^T \in \{1, \ldots, K_{AC}\}^{K_{AC}}$ are the mode indices ordered by descending confidence score $\hat{p}_{i_k}$. Consequently, the index k refers to the top K modes ranked by confidence scores.

The Average Precision (AP) is computed as the area under the precision-recall curve [44]:

$$AP(i) = \frac{\sum_{K=1}^{K_{AC}} P@K(i) \cdot \text{relevance}(\hat{Q}_{i_{o_K}})}{\sum_{k=1}^{K_{AC}} \text{relevance}(\hat{Q}_{i_k})}. \tag{4.12}$$

Finally, the mean Average Precision (mAP) is computed by averaging the AP values over different categories of shapes of the underlying ground truth trajectory [15]:

$$mAP = \frac{1}{|\mathscr{C}|} \sum_{c \in \mathscr{C}} AP_c. \tag{4.13}$$

Here, $\mathscr{C}$ is the set of eight categories of ground truth trajectory shapes: straight, straight-left, straight-right, left, right, left u-turn, right u-turn, and stationary [15]. AP_c represents the mean of the Average Precision $AP(i)$ over all focal agent predictions i in a category $c \subset \Omega_{\text{focal}}$. Note that, as opposed to all other metrics presented here, a higher mAP indicates better behavior.

4.4.2 Metrics for Interpretability and Feasibility

Moving beyond standard metrics, we introduce specialized metrics that assess the model's interpretability and feasibility. These metrics evaluate how well the model's predictions align with prior knowledge and physical constraints, as motivated in Section 1.2.

Interpretability Metrics

For interpretability, we measure the correlation of minADE with the difference between the prior's interaction importance score β_{ij} and the network's attention scores. Due to the different mechanics of the the prior integration methods, we select either the predicted scores $\alpha_{ij}^{\text{predicted}}$ or the combined scores $\alpha_{ij}^{\text{combined}}$ as the attention scores, choosing the option that yields the stronger correlation. We define the **Prior-to-Attention Difference** $\Delta\alpha(i)$ as the average difference over all neighbors $j \in J_i$ of focal agent i:

$$\Delta\alpha(i) = \max_{\alpha \in \{\alpha^{\text{predicted}}, \alpha^{\text{combined}}\}} \frac{1}{|J_i|} \sum_{j \in J_i} |\alpha_{ij} - \beta_{ij}|. \tag{4.14}$$

The **Pearson correlation coefficient** ρ is then computed as follows [44]:

$$\rho(\text{minADE}, \Delta\alpha) = \frac{\sum_{i \in \Omega_{\text{focal}}} (\text{minADE}(i) - \overline{\text{minADE}})(\Delta\alpha(i) - \overline{\Delta\alpha})}{\sqrt{\sum_{i \in \Omega_{\text{focal}}} (\text{minADE}(i) - \overline{\text{minADE}})^2 \sum_{i \in \Omega_{\text{focal}}} (\Delta\alpha(i) - \overline{\Delta\alpha})^2}}, \quad [4.15]$$

where minADE is short for $\text{minADE}_{K_{AC}}$ with K_{AC} predicted modes. $\overline{\text{minADE}}$ and $\overline{\Delta\alpha}$ denote the respective mean values of $\text{minADE}(i)$ and $\Delta\alpha(i)$ over all predicted focal agents $i \in \Omega_{\text{focal}}$. We also compute the **p-value** to assess the statistical significance of this correlation [44].

Feasibility Metrics

For feasibility, we introduce metrics that evaluate the physical realism of the predicted trajectories. These metrics ensure that the predicted trajectories adhere to kinematic constraints and feasibility limits, as discussed in Section 3.4.2.

The **Infeasible Acceleration Step Rate** measures the proportion of steps in which the predicted acceleration a_i^t is physically impossible. As introduced in Section 3.4.2, for vehicles, cyclists, and pedestrians, accelerations are considered infeasible if they fall outside the range $[-8, 8]$ m/s^2. This metric is defined as

$$\text{Infeasible Acceleration Step Rate} = \frac{1}{|\Omega_{\text{focal}}| \cdot T_h} \sum_{i \in \Omega_{\text{focal}}} \sum_{t=T_o+1}^{T_o+T_h} \mathbb{1}(|a_i^t| > 8). \quad [4.16]$$

Analogously, the **Infeasible Curvature Step Rate** measures the proportion of steps with infeasible curvature κ_i^t. As introduced in Section 3.4.2, for vehicles and cyclists, curvatures outside the range $[-0.3, 0.3]$ 1/m are considered infeasible. For pedestrians, curvature is unlimited. The metric is defined as

$$\text{Infeasible Curvature Step Rate} = \frac{1}{|\Omega_{\text{focal}}| \cdot T_h} \sum_{i \in \Omega_{\text{focal}}} \sum_{t=T_o+1}^{T_o+T_h} \mathbb{1}(|\kappa_i^t| > 0.3). \quad [4.17]$$

The **Infeasible Speed Step Rate** evaluates the proportion of steps with physically impossible speed $||v_i^t||$. As introduced in Section 3.4.2, for vehicles and cyclists, speeds are considered infeasible if they are outside the range $[0, 36]$ m/s, while for pedestrians the range is $[0, 10]$ m/s. This metric is defined as

$$\text{Infeasible Speed Step Rate} = \frac{1}{|\Omega_{\text{focal}}| \cdot T_h} \sum_{i \in \Omega_{\text{focal}}} \sum_{t=T_o+1}^{T_o+T_h} \mathbb{1}(\|v_i^t\| > v_{\max_i}). \quad [4.18]$$

The **Infeasible Step Rate** captures the overall proportion of steps that exhibit infeasible acceleration, curvature, or speed:

$$\text{Infeasible Step Rate} = \frac{1}{|\Omega_{\text{focal}}| \cdot T_h} \sum_{i \in \Omega_{\text{focal}}} \sum_{t=T_o+1}^{T_o+T_h} \mathbb{1}(|a_i^t| > 8 \vee |\kappa_i^t| > 0.3 \vee \|v_i^t\| > v_{\max_i}). \quad [4.19]$$

The **Infeasible Prediction Rate** is the proportion of predicted trajectories that contain at least one infeasible step as defined by the Infeasible Step Rate:

$$\text{Infeasible Prediction Rate} = \frac{1}{|\Omega_{\text{focal}}|} \sum_{i \in \Omega_{\text{focal}}} \mathbb{1}\left(\exists t \in \{T_o+1, \ldots, T_o+T_h\}.\ |a_i^t| > 8 \vee |\kappa_i^t| > 0.3 \vee \|v_i^t\| > v_{\max_i}\right). \quad [4.20]$$

4.4.3 Class-Specific Metrics

All the metrics mentioned above are calculated across all agents, regardless of their class. To evaluate the model's performance on specific classes of agents, we also calculate class-specific metrics. These metrics are computed by restricting the evaluation to agents belonging to a particular class. For example, the pedestrian-specific minimum Average Displacement Error (ped-minADE) is computed as

$$\text{ped-minADE}_{K_{AC}} = \text{ped-minADE}_{K_{AC}}(\Omega) = \frac{1}{|\Omega_{\text{ped}}|} \sum_{i \in \Omega_{\text{ped}}} \text{minADE}_{K_{AC}}(i), \quad [4.21]$$

where $\Omega_{\text{ped}} = \{i \in \Omega_{\text{focal}} \,|\, \text{agent\_class}(i) = ped\}$ are only the focal pedestrians from the scenes Ω. Analogous metrics are defined for vehicles and cyclists. This allows for a more granular analysis of the model's strengths and weaknesses across different agent types, ensuring that the proposed method performs well for both vehicles and pedestrians.

5 Evaluation

In this chapter, we systematically evaluate our proposed modifications to the network with a focus on quantifying their impact on prediction accuracy, interpretability, and kinematic feasibility. Unless stated otherwise, all models are trained for 15 dataset epochs on the training split of Argoverse 2, as described in Section 4.2.4. We perform all evaluations on the Argoverse 2 validation split with the same preprocessing as introduced in Section 4.1, except for disabling the randomized coordinate origin per scene. This evaluation is structured to progressively analyze each contribution individually and in combination, moving from the effects of class-specific interaction layers via interaction prior integration to the kinematic layer. The following sections will each first explain the setup of the evaluation, then present the results, and finally discuss the findings.

5.1 Impact of Class-Specific Interaction Layers

Our first experiment evaluates the effect of the class-specific agent-to-agent Transformer layer, as introduced in Section 3.3.1. We compare the network performance when adding the layer to HPTR_b and HPTR_k, as introduced in Section 4.2.4. We measure the metrics for all agents and per agent class. We obtain the per-class metrics by accumulating and averaging the metrics only over predictions for the given class, as defined in Section 4.4.3. We present the results in Table 5.1.

We observe that the introduction of class-specific interaction layers does not harm HPTR's performance. When added to the plain HPTR_b model, the networks perform on par. The new layers even significantly improve performance when integrated into HPTR together with a kinematic layer, represented by HPTR_{ci+k}, which contains class-specific prediction heads. This indicates that the class-specific layers in the agent encoder are particularly effective when the decoder also has class-specific layers that facilitate the prediction of distinct behaviors for each agent class.

Table 5.1: Ablation on class-specific interaction layers based on HPTR_b and HPTR_k.

Model	Kinematic Layer	Class-Specific Interaction Layers	All Agents				Vehicle		Pedestrian		Cyclist	
			brier-minFDE$_6$	mAP	minADE$_6$	minFDE$_6$	minADE$_6$	minFDE$_6$	minADE$_6$	minFDE$_6$	minADE$_6$	minFDE$_6$
HPTR_b			**2.28**	0.275	**0.82**	1.49	**0.96**	1.73	**0.44**	**0.80**	1.07	1.93
HPTR_{ci}		✓	**2.28**	**0.288**	0.84	**1.48**	**0.96**	**1.72**	0.49	0.90	**1.06**	**1.82**
HPTR_k	✓		2.58	0.231	1.05	1.80	1.21	2.04	0.49	0.88	1.46	2.48
HPTR_{ci+k}	✓	✓	2.50	0.179	0.99	1.70	1.18	1.95	0.49	0.89	1.31	2.27

Table 5.2: Correlation ρ of minADE with the difference between the network's attention and the prior model's interaction importance score. The analysis was conducted on the predicted agent-to-agent attention of HPTR$_{ci}$'s class-specific agent-to-agent interaction layers.

Prior Model	$\rho(\text{minADE}_6, \Delta\alpha)$	p
DG-SFM	0.132	$< 10^{-12}$
SKGACN	0.025	$< 10^{-12}$

The most substantial improvements in both comparisons are seen for cyclists. This suggests that isolating their unique behavior, such as weaving through traffic, addresses the class imbalance issue and enhances the prediction quality.

For pedestrians, the class-specific layers yield the least improvement. This hints at a greater complexity in learning pedestrian interactions compared to other agents. As described in Section 4.1, pedestrians are underrepresented in the dataset, which may require more training epochs. The significant drop in mAP when using class-specific layers with kinematic layers supports this hypothesis. The mAP measures the fit between the confidence predictions and the true multimodal distribution. A low score in the mAP, while the minADE and minFDE improve, suggests that the confidence heads have not yet converged and could thus need more training.

Vehicles show small gains from the class-specific interaction layers. This is expected, as the dataset is dominated by vehicles. With only one set of layers for all classes, vehicles largely determine the interaction encoding learned by the network. Consequently, the additional benefit is limited for this class.

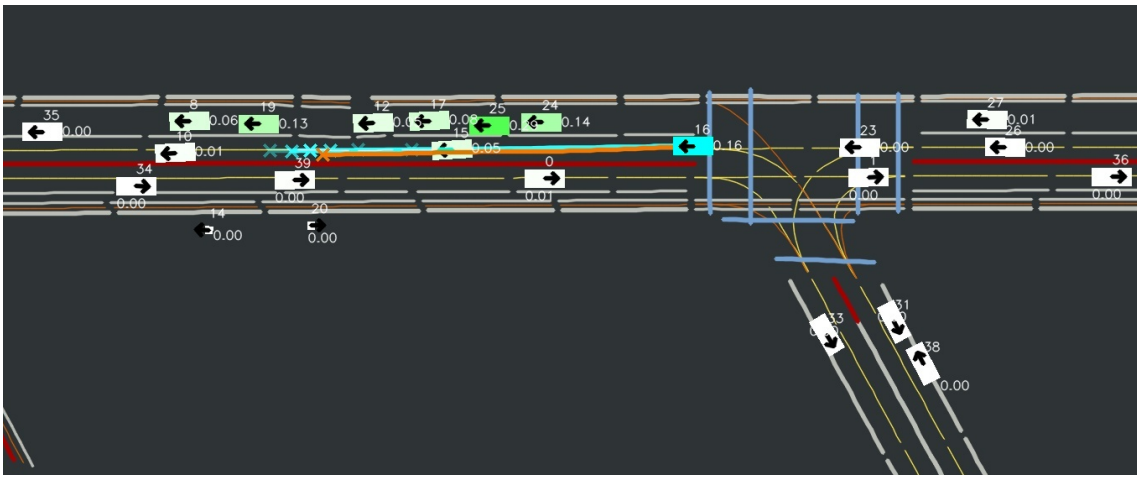
5.2 Interpretability Evaluation

In this section, we evaluate whether our proposed method can improve the interpretability of the network's interaction encoding by integrating an interaction prior into the agent-to-agent layer. First, we study the attention observed in the agent-to-agent layer without providing any prior. Based on the results of this study, we evaluate different approaches for incorporating an interaction prior into the agent-to-agent layer. Finally, we compare two prior models to determine which is the better choice to incorporate.

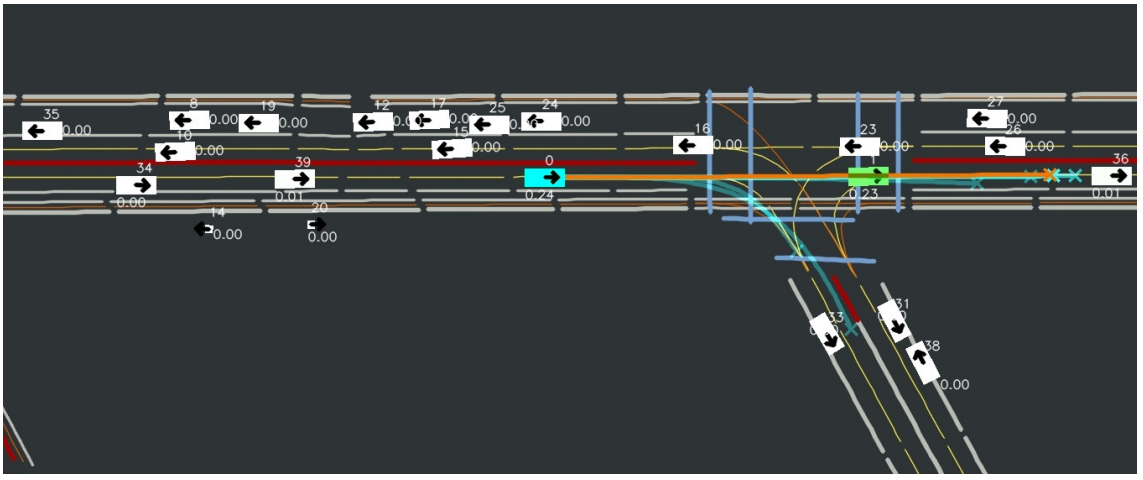
5.2.1 Reasonability of Agent-to-Agent Attention in HPTR

Firstly, we investigate whether the agent-to-agent attention fits human intuition solely based on learning from the data. Therefore, we add our class-specific agent-to-agent Transformer layer to plain HPTR$_b$ for capturing interactions, as defined in section 3.3.1 as HPTR$_{ci}$. After training the network, we analyze the attention scores in the agent-to-agent layer. We measure the correlation of minADE with the Prior-to-Attention Difference in Table 5.2. Additionally, we provide qualitative examples of HPTR$_{ci}$'s agent-to-agent attention in Figure 5.1.

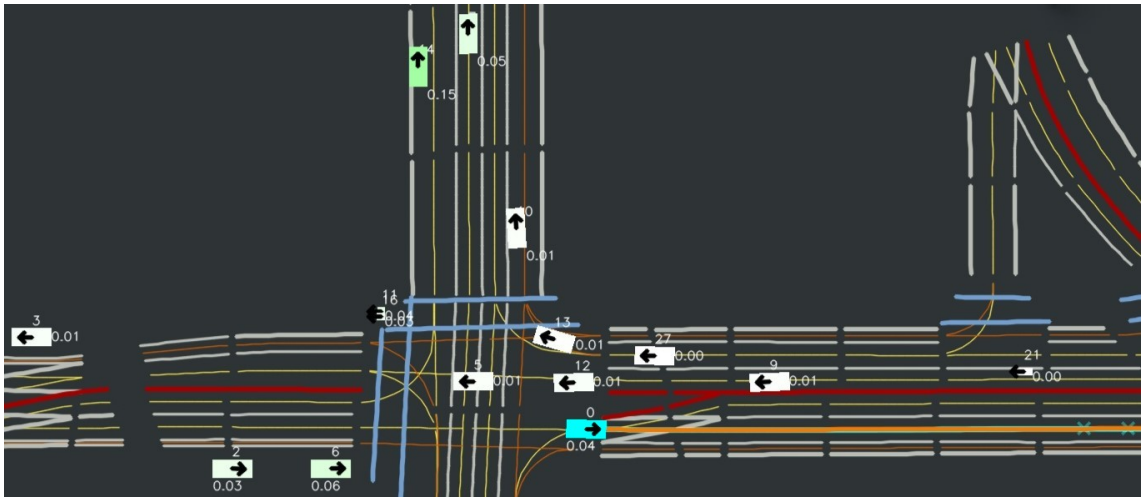
From Table 5.2, we observe a weak but statistically significant correlation between minADE and the Prior-to-Attention Difference, particularly with the DG-SFM prior. The extremely low p-values confirm that the results are statistically significant. This suggests that there is a tendency



(a) HPTR's attention matches human intuition regarding which agents are relevant to the focal agent's prediction. It focuses on upcoming vehicles parking on the roadside, which require enough space while passing.



(b) HPTR's attention does not sum to one when considering only valid agents. In this scene, the network assigned an attention score of 0.47 to an invalid agent.



(c) HPTR's attention focuses on a vehicle that is intuitively irrelevant to the focal agent's trajectory prediction because it is on a different road and moving away from the focal agent.

Figure 5.1: Qualitative examples of HPTR$_{ci}$'s agent-to-agent attention without any prior. For visualization, we adopt the same scheme as HPTR [69]. The focal agent is marked in cyan, and the ground truth trajectory is drawn in orange. The network's predicted trajectories are drawn in cyan, where a higher opacity represents a higher predicted probability for that trajectory. A black arrow indicates each agent's heading and increases in length for higher speeds. Neighboring agents are labeled with their index and attention score in the agent-to-agent attention layer. More attention is reflected by a deeper green color. The road's sidelines are colored grey and the middle line is colored red. Crosswalks are depicted in blue.

for the network's prediction to be incorrect if the network's predicted attention is very different from the prior model's interaction importance score. This tendency substantiates that aligning the encoder's reasoning with human reasoning does not only provide us with more reasonable predictions but also has the potential to improve the prediction.

One possible explanation for the correlation is that HPTR's attention is heavily misaligned for some scenes. While the attention does match the human intuition of important agents quite well for some scenes, e.g., in Figure 5.1a, there are other scenes where the network seems to have learned noise, e.g., in Figures 5.1b and 5.1c. We deem some of the attention weights to be noise because they focus on agents that are clearly irrelevant to the focal agent's prediction, such as invalid agents or agents unrelated to the focal agent.

Thus, we conclude that minimizing the discrepancy between the model's attention and the prior is a worthwhile direction. Since larger differences tend to coincide with incorrect predictions, reducing this gap might enhance prediction accuracy.

5.2.2 Comparison of Integration Methods for Interaction Priors

Motivated by the correlation we have found in the agent-to-agent layer, we continue our evaluation in search of a way to make the attention more aligned with our prior interaction scores. Accordingly, in this experiment, we compare the two prior integration methods, multiply-and-renormalize and gating-and-loss, as proposed in Section 3.3.3. We compare the performance of our baseline HPTR_b with the networks $\text{HPTR}_{i+p_{\text{MnR}}}$ and $\text{HPTR}_{i+p_{\text{GnL}}}$ that have an agent-to-agent interaction layer into which they integrate interaction importance scores. For comparison, we use the SKGACN prior for both networks in this experiment. We choose SKGACN here because it is an already established interaction prior. We base the ablation on HPTR_i, which uses an agent-to-agent layer without class-specific instances, to separate this ablation from the one in Section 5.1. We evaluate the two methods in three groups of metrics: In Table 5.3, we measure the two methods' effect on accuracy. In Table 5.4 and Figure 5.2, we analyze the correlation between the minADE and the Prior-to-Attention Difference on the networks' predicted attention scores. Lastly, in Table 5.5, we investigate the methods' ability to handle incorrect priors. Therefore, we inject counterfactual priors and compare the effects on accuracy. We draw multiple conclusions from the results.

Integrating prior interaction scores is a valuable addition. From the accuracy evaluation in Table 5.3, we see that the model with the gating-and-loss integration performs similarly to the baseline. The model deteriorates the minADE by 1 cm and improves the minFDE by 2 cm, which is a negligible discrepancy. The performance even improves significantly for cyclists. The remaining difference in performance seems to be mostly due to less accurate confidence predictions, which would explain the poorer performance in brier-minFDE and mAP. The increased model size of approximately 1 million parameters compared to the baseline may be a reason why the models with prior integration converge more slowly in the confidence prediction. Extending training time could thus eliminate the performance gap in brier-minFDE and mAP.

Table 5.3: Comparison of the multiply-and-renormalize ($\text{HPTR}_{i+p_{\text{MnR}}}$) and gating-and-loss ($\text{HPTR}_{i+p_{\text{GnL}}}$) methods for interaction prior integration.

Model	All Agents				Vehicle		Pedestrian		Cyclist	
	brier-minFDE$_6$	mAP	minADE$_6$	minFDE$_6$	minADE$_6$	minFDE$_6$	minADE$_6$	minFDE$_6$	minADE$_6$	minFDE$_6$
HPTR_b	2.28	0.275	0.82	1.49	0.96	1.73	0.44	0.80	1.07	1.93
$\text{HPTR}_{i+p_{\text{MnR}}}$	2.39	0.237	0.85	1.53	1.03	1.84	0.46	0.86	1.09	1.94
$\text{HPTR}_{i+p_{\text{GnL}}}$	2.33	0.261	0.83	1.47	0.98	1.77	0.47	0.86	1.03	1.77

Table 5.4: Correlation ρ of minADE with the difference between prior interaction scores and the network's predicted attention scores. We compare this correlation for the two interaction prior integration methods, multiply-and-renormalize and gating-and-loss, using the SKGACN prior model. HPTR with agent-to-agent interaction layers serves as the baseline.

Model	$\rho(\text{minADE}_6, \Delta\alpha)$	p
HPTR_{ci}	0.025	$< 10^{-12}$
$\text{HPTR}_{i+p_{\text{MnR}}}$	0.14	$< 10^{-12}$
$\text{HPTR}_{i+p_{\text{GnL}}}$	0.22	$< 10^{-12}$

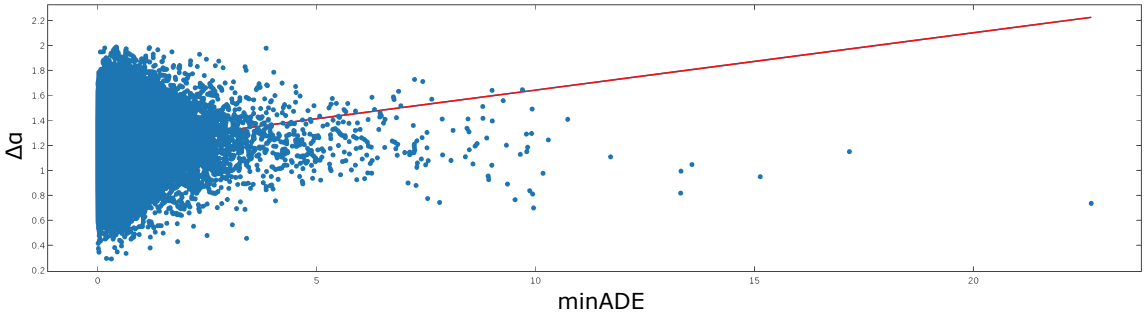


Figure 5.2: Correlation scatter plot for $\text{HPTR}_{i+p_{\text{MnR}}}$ with SKGACN containing all 25,000 scenes from the validation set. Each scene is represented by one blue point. The y-axis shows the Prior-to-Attention Difference $\Delta\alpha$ measured while computing the prediction for that scene. The x-axis represents the minADE resulting from the trajectories predicted for that scene. The red line is the linear regression line for the relationship between minADE and $\Delta\alpha$.

Table 5.5: Comparison of the multiply-and-renormalize and gating-and-loss methods when injecting counterfactual priors into the prediction process. To inject a prior, we replace the prior interaction scores given to the network in the agent-to-agent layer. We compare the effect of three counterfactual priors: *one-wins-all*, in which one randomly picked agent receives a score of one while all other agents receive a score of zero; *random*, where scores are randomly distributed across all agents following a uniform distribution; and *equal*, where all agents receive the same score.

Model	Prior as in Training			One-Wins-All Prior			Random Prior			Equal Prior		
	brier-minFDE$_6$	minADE$_6$	minFDE$_6$	brier-minFDE$_6$	minADE$_6$	minFDE$_6$	brier-minFDE$_6$	minADE$_6$	minFDE$_6$	brier-minFDE$_6$	minADE$_6$	minFDE$_6$
$\text{HPTR}_{i+p_{\text{MnR}}}$	2.39	0.85	1.53	2.46	0.88	1.59	2.43	0.86	1.55	2.42	0.86	1.55
$\text{HPTR}_{i+p_{\text{GnL}}}$	2.33	0.83	1.47	2.37	0.84	1.47	2.33	0.82	1.46	2.33	0.87	1.55

While not affecting the model performance negatively, the prior-integrated models offer the important advantage of improved encoder interpretability. The correlation shown in Figure 5.2 could be the foundation of a monitoring system for the model. If the model's attention varies strongly from the prior, this may indicate that the model is about to make an unreasonable and incorrect prediction. Such an indication could then, for example, trigger the supervision of a safety driver in an SAE level 4 autonomous system.

Gating-and-loss is superior to multiply-and-renormalize. The results of the counterfactual prior injection in Table 5.5 demonstrate that gating-and-loss can cope better with incorrect priors. In all three cases, the multiply-and-renormalize model deteriorates the brier-minFDE. This could be because it is less certain about the goal of the focal agent when gathering incorrect interaction information due to a flawed prior. It fits the math behind the integration to see it struggle more in cases where the prior does not suit the scene. Gating-and-loss can rely more on the predicted attention in such cases by setting the gating scalars accordingly. On the other hand, for multiply-and-renormalize, it is hard to recover from a high prior score on irrelevant agents as it would need to predict exceptionally low attention on these agents to balance the resulting combined attention scores. Hence, it is expected that multiply-and-renormalize performs worst with the one-wins-all prior. On balance, priors that do not capture the scene's interactions well affect gating-and-loss less than multiply-and-renormalize.

Moreover, gating-and-loss also offers superior accuracy, as shown in Table 5.3, and provides better interpretability, as its Prior-to-Attention Difference correlates more strongly with prediction errors, as illustrated in Figure 5.2. Due to the two advantages of robustness and accuracy, we choose gating-and-loss over multiply-and-renormalize for our final model.

The influence of the interaction embedding is limited. The correlations shown in Table 5.4 and Figure 5.2 are present but weak for both the prior-integrated models. Additionally, both prior integration methods can cope with flawed priors without a large effect on their accuracy. One reason for this might be that some predictions do not require high-quality interaction information from neighboring agents as much as others. Scholler et al. [50] showed that modern trajectory prediction benchmarks contain a major number of scenes in which the focal agent moves straight. For such scenes, interaction information may be less relevant than for scenes in which the focal agent takes a turn, changes the lane, or crosses a crosswalk. Another factor for the limited influence of the interaction embedding might be the Transformer architecture. In the agent-to-agent layer, the Transformer's residual-add layer could learn to forward only a small portion of the information gathered from the neighbors' embeddings. Thus, it is worth exploring the integration of interaction priors more. Our results do not clearly identify the circumstances that benefit a strong correlation between minADE and Prior-to-Attention Difference, which could help to improve interpretability further.

Table 5.6: Comparison of the SKGACN and DG-SFM prior models for interaction importance scores. As a baseline, the two models are compared to the L2 prior, which assigns each agent an interaction importance score inversely proportional to its Euclidean distance from the focal agent.

Model	Prior	All Agents				Vehicle		Pedestrian		Cyclist	
		brier-minFDE$_6$	mAP	minADE$_6$	minFDE$_6$	minADE$_6$	minFDE$_6$	minADE$_6$	minFDE$_6$	minADE$_6$	minFDE$_6$
HPTR$_b$	-	**2.28**	0.275	**0.82**	1.49	**0.96**	**1.73**	**0.44**	**0.80**	1.07	1.93
HPTR$_{i+p_{\text{GnL}}}$	L2	2.36	**0.279**	0.83	1.48	1.00	1.80	0.47	0.83	1.04	1.83
	SKGACN	2.33	0.261	0.83	1.47	0.98	1.77	0.47	0.86	1.03	1.77
	DG-SFM	2.36	0.273	**0.82**	**1.46**	1.00	1.79	0.47	0.84	**1.00**	**1.73**

Table 5.7: Correlation ρ of minADE with the difference between prior interaction scores and the network's predicted attention scores before they are combined with the prior. We compare this correlation when computing the prior with the three different prior models: SKGACN, DG-SFM, and L2.

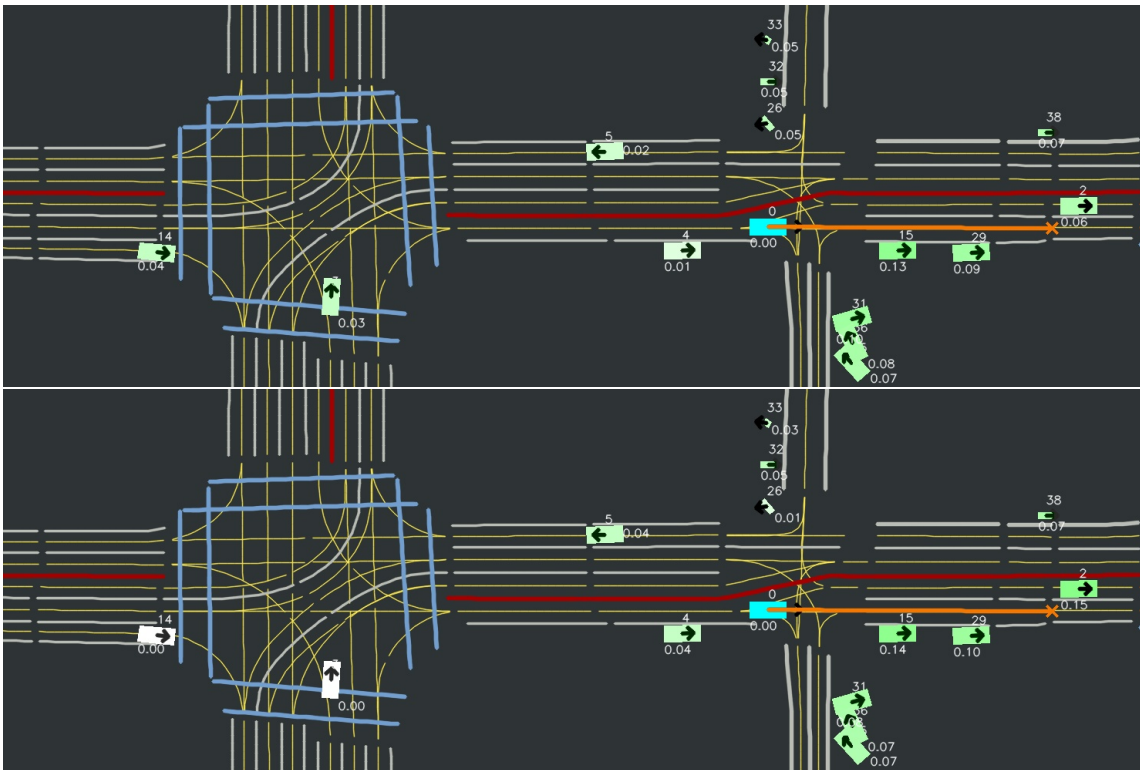
Model	Prior	ρ(minADE$_6$, $\Delta\alpha$)	p
HPTR$_{i+p_{\text{GnL}}}$	L2	-0.13	$< 10^{-12}$
	SKGACN	0.22	$< 10^{-12}$
	DG-SFM	0.23	$< 10^{-12}$

5.2.3 Comparison of Interaction Priors

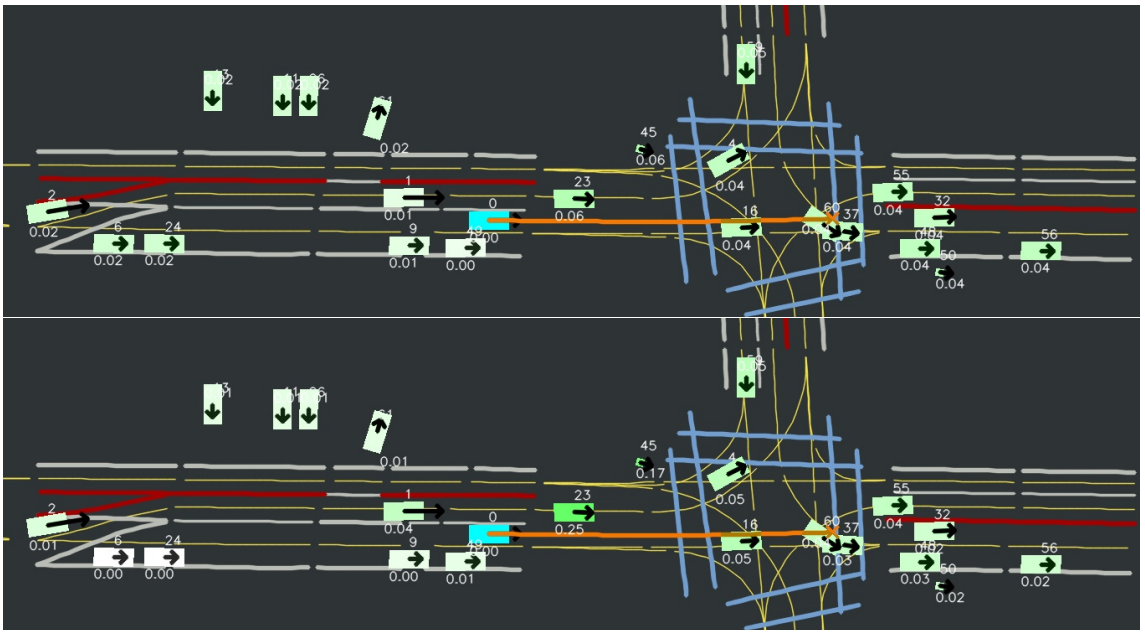
After analyzing the prior integration methods, we study the impact of the underlying prior on the prediction. Therefore, we compare the two prior models, SKGACN and DG-SFM, as introduced in Section 3.3.2, to determine which works best. For comparison, we train one network with the HPTR$_{i+p_{\text{GnL}}}$ architecture for each prior model. As a baseline, we also compare SKGACN and DG-SFM to the naive L2 prior. It assigns each agent an interaction importance score inversely proportional to its Euclidean distance from the focal agent. For reference, we also provide the metrics for the baseline network HPTR$_b$, which has no prior integration. In Tables 5.6 and 5.7, we provide the accuracy and correlation metrics used for evaluating the prior integration methods in Section 5.2.2. Furthermore, we provide qualitative examples illustrating the differences between the priors in three scenes in Figure 5.3.

The accuracy comparison in Table 5.6 shows that the more advanced SKGACN and DG-SFM priors outperform the naive L2 prior in terms of minADE and minFDE. DG-SFM performs similarly to the baseline HPTR$_b$, with only marginal improvements in most metrics, except for cyclists, where it shows a significant advantage. SKGACN demonstrates slightly lower accuracy for vehicles and pedestrians compared to DG-SFM. Interestingly, DG-SFM and SKGACN deteriorate the brier-minFDE compared to the network with the L2 prior, while improving in all other metrics. One possible explanation is that, while the L2 network's predictions may be less accurate, it is easier for the confidence head to predict which mode will be best because the more consistent prior led to more consistent behavior in the modes.

In terms of correlation, as presented in Table 5.7, both the SKGACN and DG-SFM priors yield significantly higher results than the network without priors shown in Table 5.2. The L2 prior, by contrast, fails to produce any meaningful correlation. This substantiates our argument that the correlation arises from unaligned interaction attention. Since the L2 prior does not adequately



(a) SKGACN (top) assigns the highest score to the parking cars ahead of the focal agent. Even agents far away, like agent 14, receive a substantial score. In contrast, DG-SFM (bottom) assigns the highest score to the leading vehicle with id 2, as the focal agent catches up with it. Agents too far away to interact with the focal agent receive barely any score.



(b) SKGACN (top) assigns higher scores to the agents in the intersection, while agents behind the focal agent receive a lower score. Vehicle 1, which is about to overtake the focal agent in the left lane, receives a score close to zero. Similar to SKGACN, DG-SFM (bottom) assigns even lower scores to the stationary agents behind the focal agent. Nevertheless, DG-SFM gives the overtaking vehicle 1 a score comparable to the score of the agents in the intersection. Additionally, DG-SFM highlights pedestrian 45, who is approaching the intersection, and vehicle 23, which the focal agent is overtaking.

Figure 5.3: Qualitative examples of the interaction importance scores calculated by SKGACN (top) and DG-SFM (bottom). The illustration scheme is the same as in Figure 5.1, with a higher score reflected by a deeper green color. In contrast to HPTR, the prior gives no attention to the focal agent since the agent-to-agent layer shall model the importance of the neighboring agents, not that of the focal agent.

represent reasonable interaction importance, it does not facilitate the same correlation as the SKGACN and DG-SFM priors.

The qualitative examples reinforce that DG-SFM has an edge over SKGACN. DG-SFM produces more realistic and reasonable attention patterns than SKGACN and the HPTR model without any prior. The examples show that DG-SFM better aligns with human intuition, assigning higher importance to relevant agents, such as those directly interacting with the focal agent, as in Figure 5.3b, while ignoring distant or irrelevant agents, as in Figure 5.3a. These observations fit DG-SFM's theoretical advantages over SKGACN from Section 3.3.2. DG-SFM should be more capable in complex traffic scenarios and more suited for mixed use with vehicles, pedestrians, and cyclists. In Figure 5.3b, we can see the advantage of DG-SFM in considering agents that are behind the focal agent. Just as in our theoretical considerations in Figure 3.5, DG-SFM can pay attention to an overtaking vehicle while SKGACN struggles with that situation. Moreover, Figure 5.3a confirms that DG-SFM depends less on the distance to the focal agent for determining the interaction scores, as discussed in Figure 3.6. While SKGACN assigns the highest scores to nearby parking vehicles, DG-SFM can additionally focus on other agents relevant for interaction.

In conclusion, DG-SFM demonstrates superior performance in key metrics, good correlation between prior difference and prediction error, as well as more realistic qualitative attention patterns. The mathematically substantiated advantages of its modeling make it the more robust choice for complex scenarios. Hence, we choose DG-SFM over SKGACN for our final model.

5.3 Kinematic Feasibility Evaluation

This section assesses the effect of adding our kinematic layer. Additionally, we examine the side effects that arise from enforcing kinematic feasibility through kinematic models. Therefore, we start by measuring the extent of infeasible motions in the dataset's ground-truth trajectories. Next, we evaluate the error that results from the kinematic models' inability to reproduce these infeasible motions in the dataset. Lastly, we study the effect on the network's accuracy when forcing it to adhere to the agents' kinematics by adding our kinematic layer.

5.3.1 Infeasible Motions in the Ground-Truth Trajectories

The goal of this analysis is to evaluate the physical feasibility of the ground-truth trajectories in the dataset to determine whether enforcing physical feasibility could prevent the network from reproducing the ground truths accurately. For this purpose, we analyze the Argoverse 2 dataset's trajectories in both the training and validation splits. Table 5.8 assesses the infeasibility of the dataset based on the physical limits defined. Specifically, we measure the proportion of infeasible steps in the trajectories, as well as the proportion of scenes containing at least one trajectory with an infeasible step. A step is considered infeasible if its acceleration or curvature exceeds the defined feasible limits in Section 3.4.2.

The analysis reveals that the dataset contains a significant number of infeasible steps. In particular, the curvature measurements show a high rate of infeasibility, where over 30% of the

Table 5.8: Statistics of measured acceleration and curvature in the dataset's ground-truth trajectories. Each trajectory consists of 110 steps observed with a sampling time of 0.1 s between the steps. We determine the share of infeasible steps in all trajectories in the dataset. Additionally, we determine the share of scenes, which contain the trajectories of up to $N_{AG} = 64$ agents, that have at least one trajectory with at least one infeasible step. As defined in Section 3.4.2, an acceleration step is considered infeasible if it is not in the range $[-8, 8]$ m/s^2 and a curvature step is considered infeasible if it is not in the range $[-0.3, 0.3]$ 1/m.

Agent Class	Dataset Split	Acceleration [m/s^2]				Curvature [1/m]				Velocity [m/s]	
		min	max	Infeasible Step Rate	Scenes with at least one Infeasible Step	min	max	Infeasible Step Rate	Scenes with at least one Infeasible Step	max	Infeasible Step Rate
Vehicles	Training	−241	202	0.33%	16.4%	$< -10^9$	$> 10^9$	32.6%	60.4%	34.9	0%
	Validation	−163	153	0.33%	16.4%			32.7%	60.6%	29.6	
Pedestrians	Training	−58.4	31.1	0.008%	0.22%			-		9.4	0%
	Validation	−27.1	19.4	0.014%	0.32%					8.9	
Cyclists	Training	−58.1	34.6	0.034%	1.02%	$< -10^9$	$> 10^9$	41.6%	72.7%	26.2	0%
	Validation	−17.4	18.0	0.023%	0.64%			41.9%	72.0%	20.3	

steps for vehicles and over 40% for cyclists are labeled infeasible. This can likely be attributed to sensor noise. On the small scale of 0.1 s between steps, a small error in the sensed position can quickly lead to extreme curvatures that exceed the kinematically feasible limits. Additionally, infeasible step rates as high as 40% raise the question of whether the used curvature limit of 0.3 1/m is set too low for cyclists. However, fine-tuning the feasibility limits for cyclists goes beyond the scope of this thesis.

Either way, these findings support the argument for incorporating a kinematic layer, as it highlights the prevalence of kinematic infeasibility in the dataset. Without a kinematic layer, the network would likely learn to reproduce these infeasible trajectories. That would be undesirable as it would essentially mean learning the distribution of the sensor noise to minimize the distance to the ground-truth future trajectories. Enforcing kinematic constraints, therefore, ensures that the network produces realistic, kinematically feasible predictions. Besides being more aligned, this should also benefit the generalizability of the model across different datasets, since the characteristics of the sensor noise are specific to the dataset.

Additionally, the results substantiate the velocity and acceleration limits chosen for this thesis. The limits selected in Section 3.4.2 are both realistic and grounded in the dataset's observed statistics. The velocity limits are never reached, while the acceleration limits are only exceeded by rare extreme cases of up to $|a| = 241$ m/s^2.

5.3.2 Limitations of the Trajectories Produced by Kinematic Models

Having established that the ground-truth trajectories contain infeasible motions, this analysis evaluates how closely our kinematic models can reproduce the ground-truth trajectories. The aim is to quantify the reproduction error introduced by the kinematic model. If such an error exists, it suggests that even a network making perfect predictions will not be able to perform better than this error, as the kinematic constraints prevent exact reproduction.

To measure the reproduction error, we use a greedy algorithm to find the best possible control inputs for the kinematic model given the future trajectory. At each step, the algorithm selects the control inputs necessary to reach the position and heading of the next step in the ground-

Table 5.9: Reproduction error introduced by kinematic models. Since the ground-truth trajectories are not feasible in all steps, it is not possible to reproduce every ground-truth trajectory perfectly when adhering to the limits of physics represented by the kinematic models. For infeasible steps, our greedy reproduction algorithm picks the best control inputs possible to get as close as possible to the ground-truth trajectory's next position.

Agent Class	Kinematic Model	minADE_6	minFDE_6	MR_6
Pedestrian	Single Integrator	0	0	0
	Double Integrator	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	0
	Unicycle	0.521	1.14	6.2%
Vehicle	Unicycle	0.207	0.587	2.2%
Cyclist	Unicycle	0.861	1.92	8.9%

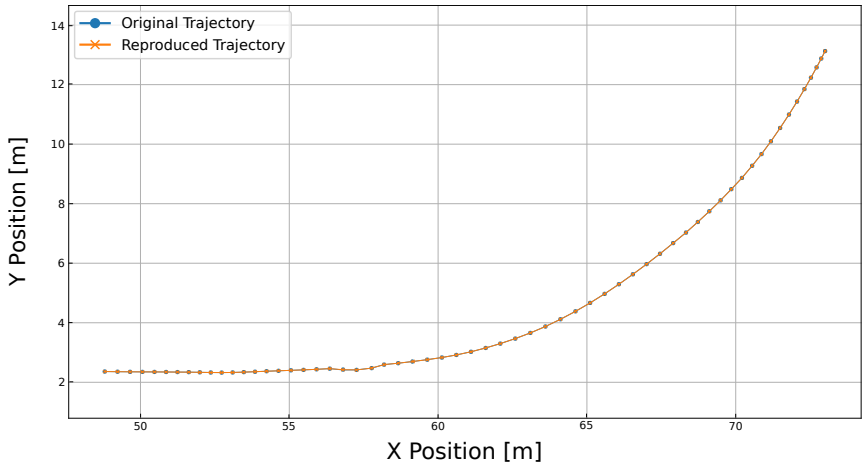
truth trajectory. However, when the feasibility limits of the kinematic model prevent reaching the next position, the algorithm selects the control inputs that bring the reproduced trajectory as close as possible to the ground-truth trajectory. Note that this algorithm does not always find the mathematically ideal solution. Nevertheless, it is sufficient for our application.

In Table 5.9, we report the minADE, minFDE, and MR between the ground truth trajectories and the trajectories reproduced by the algorithm steering the kinematic model. The experiment was conducted on the validation dataset. To offer a better understanding of the algorithm, we illustrate an example of a trajectory reproduction in Figure 5.4. We draw multiple conclusions from the result.

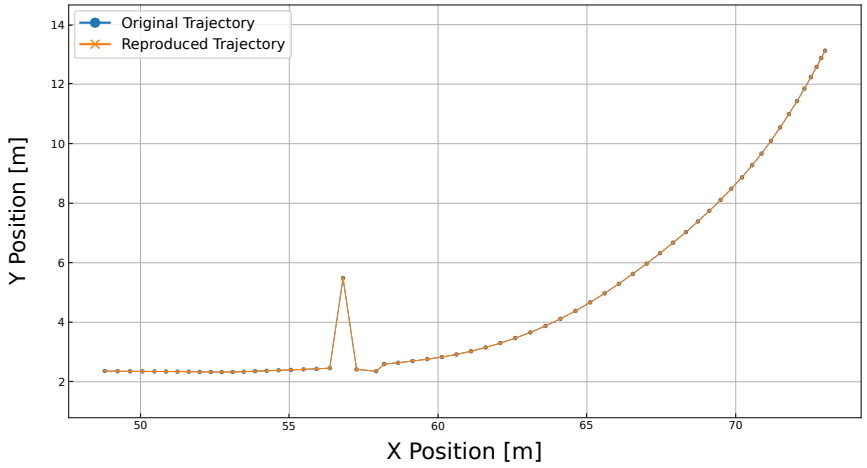
Kinematic models lead to a significant reproduction error. This is expected since the dataset contains a significant number of infeasible motions. For pedestrians, there is almost no reproduction error when using the single or double integrator, as both models do not limit the turning capabilities and acceleration violations are rare. However, we observe higher errors for vehicles and cyclists, especially in minFDE. We believe this error arises from the reproduced trajectories lagging behind the ground truth. This occurs because the kinematic model is unable to accelerate or turn as rapidly as the ground truth in certain instances, causing the reproduced trajectory to fall behind and fail to reach the ground truth's final position in time.

Cyclist trajectories exhibit particularly high errors since we do not fine-tune the feasibility limits for cyclists. We focus only on optimizing our network for pedestrians and vehicles to demonstrate our method. Here, the unicycle model with the same limits as for vehicles seems too restrictive for cyclists.

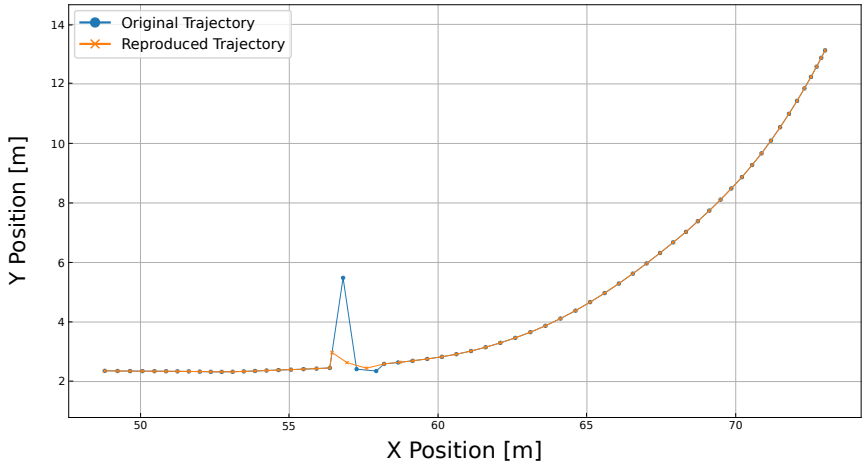
When regarding the high errors, it is also important to note that the greedy algorithm does not always find the best reproduction possible. For instance, in Figure 5.4c, the algorithm's greedy logic drives it to make an inefficient step toward an infeasible outlier, whereas a straight trajectory might have resulted in a smaller minADE. This issue becomes more pronounced with the unicycle model, which is constrained not only by acceleration limits but also by curvature limits.



(a) The algorithm reproduces feasible trajectories perfectly as every step in the trajectory can be produced by giving the kinematic model the corresponding control inputs.



(b) If there were no limits on the control inputs, the algorithm could also reproduce infeasible trajectories perfectly.



(c) When the feasibility limits prevent the algorithm from reaching the ground truth's next position, it is not clear which control inputs to pick. In each step, our reproduction algorithm greedily picks the control inputs that get the reproduced trajectory closest to the ground-truth trajectory's position at that step.

Figure 5.4: Reproduction of an exemplary trajectory employing our reproduction algorithm with the double integrator as the kinematic model.

The double integrator is a better kinematic model for pedestrians than the single integrator and the unicycle. The double integrator strikes the best balance. It is more physically feasible than the single integrator, which only enforces a maximum velocity. Yet, as opposed to the unicycle, it is versatile enough to reproduce all of the dataset's trajectories. As a result, we select the double integrator as the kinematic model for pedestrians in our subsequent experiments.

Reproduction errors will hurt the performance of networks with a kinematic layer. Kinematic models can reproduce any trajectory if there are no control input limits. However, once feasibility limits are enforced, there will be samples in the dataset that cannot be reproduced precisely due to their infeasibility. Consequently, the reproduction error we report represents the lower bound of the best possible performance a network with a kinematic layer can achieve when using the respective kinematic models. Any network performing better than this error is likely overfitting to noise in the dataset. For example, the high reproduction error for vehicles using the unicycle model suggests that the dataset contains some rare, noisy samples that violate physics. Networks without feasibility guarantees can easily learn these noisy and infeasible behaviors, leading to better metric scores, but undesirable and unrealistic predictions. In contrast, our kinematic layer prevents the model from learning such noise, which is a significant advantage. It ensures that our predictions are more physically realistic and less prone to overfitting on noise present in the training data.

5.3.3 Impact of Kinematic Layers

Aware of the infeasibilities in the dataset, we aim to determine the impact of kinematic layers on the model's predictions. To achieve this, we compare the baseline HPTR_b model, which does not include any kinematic constraints, with HPTR_k, which incorporates the kinematic layers, as defined in Section 3.4.2. Furthermore, we evaluate the effect of kinematic layers combined with our class-specific agent-to-agent layer by comparing HPTR_{ci} and HPTR_{ci+k}. The evaluation is carried out using two types of metrics. First, we measure the predictive performance of the models in Table 5.10. Most of these results were presented before in Table 5.1 but are reorganized here to highlight the effect of kinematic layers. Second, we measure the infeasibility of the official HPTR model's predictions in Table 5.11. There, we analyze the rate of infeasible steps and trajectories and how they are distributed over the agent classes.

Looking at the results in Table 5.10, it is clear that the introduction of kinematic layers reduces the prediction accuracy. For instance, HPTR_k and HPTR_{ci+k} show a higher minADE_6 and minFDE_6 compared to their counterparts without kinematic layers. This performance decrease is particularly pronounced for cyclists, which can be attributed to the lack of a more suitable kinematic model for this agent class in our setup. In contrast, the performance of pedestrians remains relatively close to the baseline, indicating that the kinematic constraints have less of an impact on this class. Furthermore, we see that our class-specific interaction layers remarkably improve the prediction accuracy obtained with the kinematic layer. HPTR_{ci+k} significantly

Table 5.10: Ablation on kinematic layers based on HPTR_b and HPTR_{ci}. We measure the rate of infeasible steps and trajectories among all trajectories output by the networks as predictions. A step of a predicted trajectory is considered infeasible if it exceeds the defined limits from Section 3.4.2. A predicted trajectory is considered infeasible if it contains at least one infeasible step.

Model	Class-Specific Interaction Layers	Kinematic Layer	All Agents						Vehicle		Pedestrian		Cyclist	
			brier-minFDE_6	mAP	minADE_6	minFDE_6	Infeasible Step Rate	Infeasible Prediction Rate	minADE_6	minFDE_6	minADE_6	minFDE_6	minADE_6	minFDE_6
HPTR_b			2.28	0.275	0.82	1.49	26.7%	99.5%	0.96	1.73	0.44	0.80	1.07	1.93
HPTR_k		✓	2.58	0.231	1.05	1.80	0%	0%	1.21	2.04	0.49	0.88	1.46	2.48
HPTR_{ci}	✓		2.28	0.288	0.84	1.48	25.9%	99.3%	0.96	1.72	0.49	0.90	1.06	1.82
HPTR_{ci+k}	✓	✓	2.50	0.179	0.99	1.70	0%	0%	1.18	1.95	0.49	0.89	1.31	2.27

Table 5.11: Feasibility statistics of the trajectories predicted by the official HPTR model trained by Zhang et al. [69], which we loaded from their publicly available checkpoint. The *Any* columns give the rate of steps and trajectories that violate the acceleration, curvature, or speed limit.

Agent Class	Infeasible Step Rates [%]				Predictions with at least one Infeasible Step [%]			
	Acceleration	Curvature	Speed	Any	Acceleration	Curvature	Speed	Any
Vehicles	3.7	12	0	16	60	37	0	89
Pedestrians	0.008	-	0	0.08	0.3	-	0	0.3
Cyclists	1.5	22	0	23	33	64	0	89
All	3.4	14	0	17	56	42	0	88

outperforms HPTR_k, even though its foundation HPTR_{ci} performs slightly worse than HPTR_b. This underpins our argument of synergy effects between our contributions.

Despite the reduction in performance, the primary advantage of incorporating kinematic layers is the guarantee of kinematic feasibility. As demonstrated in Table 5.11, the official HPTR model, trained for 75 instead of 15 dataset epochs, still exhibits a high rate of infeasibility. 56% of all trajectories predicted by HPTR contain accelerations exceeding 8 m/s^2, which is physically impossible in reality. This high rate of infeasibility is concerning because it means that over half of the predicted trajectories can not occur in real-world scenarios. Interestingly, the infeasible steps are mostly either due to excessive acceleration or curvature, but not both. This can be observed from the percentages of infeasible acceleration and curvature steps almost summing to the percentage of steps with any infeasibility.

In stark contrast, our models with kinematic layers never produce an infeasible trajectory. The high infeasibility in HPTR substantiates the need for kinematic constraints. By enforcing physical constraints, we ensure that our network predicts physically realistic trajectories and does not reproduce any noisy behavior from the dataset, as discussed in Section 5.3.2. This leads to more reliable, aligned, and realistic trajectory predictions. Even though this comes at the cost of worse accuracy scores compared to the baseline, this trade-off is ultimately well worth it.

Table 5.12: Comparison of our proposed model with our ablation baseline HPTR_b and the official HPTR model trained by Zhang et al. [69] on the Argoverse 2 validation split. The correlation of minADE and Prior-To-Attention Difference $\rho(\text{minADE}_6, \Delta\alpha)$ is not measurable for the two baselines without an agent-to-agent layer.

Model	Trained Epochs	brier-minFDE$_6$	mAP	minFDE$_6$	minFDE$_1$	MR$_2$	minADE$_6$	minADE$_1$	Infeasible Step Rate	Infeasible Prediction Rate	$\rho(\text{minADE}_6, \Delta\alpha)$	p
Official HPTR	75	**2.02**	**0.302**	**1.28**	**6.17**	**14.3%**	**0.69**	**2.44**	17.3%	87.7%	-	-
HPTR w/ Huber loss (HPTR_b)	15	2.28	0.275	1.49	6.57	17.4%	0.82	2.64	26.7%	99.6%	-	-
Ours ($\text{HPTR}_{ci+p_{\text{dg-sfm}}+k}$)	30	2.27	0.278	1.51	6.67	18.2%	0.90	2.69	**0%**	**0%**	**0.12**	$\mathbf{< 10^{-12}}$

Table 5.13: Comparison of the per-class accuracy of the combinations of our proposed changes. All metrics, also those for $\text{HPTR}_{ci+p_{\text{dg-sfm}}+k}$, were measured after training for 15 dataset epochs.

Model	Kinematic Layer	Class-Specific Interaction Layers	Interaction Prior Integration	All Agents				Vehicle		Pedestrian		Cyclist	
				brier-minFDE$_6$	mAP	minADE$_6$	minFDE$_6$	minADE$_6$	minFDE$_6$	minADE$_6$	minFDE$_6$	minADE$_6$	minFDE$_6$
HPTR_b				2.28	0.275	0.82	1.49	0.96	1.73	0.44	0.80	1.07	1.93
HPTR_k	✓			2.58	0.231	1.05	1.80	1.21	2.04	0.49	0.88	1.46	2.48
HPTR_{ci+k}	✓	✓		2.50	0.179	0.99	1.70	1.18	1.95	0.49	0.89	1.31	2.27
$\text{HPTR}_{ci+p_{\text{dg-sfm}}+k}$	✓	✓	✓	2.40	0.230	0.97	1.67	1.03	1.85	0.47	0.85	1.42	2.27

5.4 Comparison with the State-of-the-Art

Finally, having conducted the individual studies on the effects of class-specific interaction layers, prior knowledge within these interaction layers, and kinematic constraints, we assess the efficacy of all the proposed changes combined. We compare our proposed model with the official HPTR model trained by Zhang et al. [69], which we loaded from their publicly available checkpoint. In Tables 5.12 and 5.13, our evaluation aims to determine how our modifications affect the model's performance. Therefore, our model, referred to as $\text{HPTR}_{ci+p_{\text{dg-sfm}}+k}$, combines all our proposed changes with the options that proved to be best suited for interpretability and feasibility in our ablation studies. Specifically, our model integrates the DG-SFM interaction prior via the gating-and-loss method, as well as a unicycle model for vehicles and cyclists and a double integrator model for pedestrians via the kinematic layer. Apart from our position-based Huber loss function, all other hyperparameters and settings align with the original HPTR model for a consistent baseline comparison, as described in Section 4.2.

In Table 5.12 the official HPTR model shows better accuracy, but the comparison is distorted by differences in training loss functions and epoch counts. Firstly, the official HPTR uses a negative log-likelihood loss, which has an edge over our Huber loss in learning multimodal distributions through Gaussian position representation [54, 9]. Additionally, its 75-epoch training period, compared to our model's 30 epochs, gives the official HPTR another distorting advantage. Training our model for 75 epochs would have taken us 30 days, which was not viable for this thesis. Our ablation results for class-specific layers and prior integration closely matched HPTR_b, suggesting that the remaining performance gap is almost exclusively due to the kinematic layer. While HPTR can learn any infeasible noise from the dataset to reduce the minADE and minFDE, our model is kinematically constrained. Thus, it can only reach a certain lower

bound of minADE and minFDE, as discussed in Table 5.9. Consequently, our model is similarly close to its lower bound performance as the official HPTR.

Despite the accuracy trade-off, our model's kinematic constraints provide critical benefits in feasibility, preventing the learning of unrealistic patterns and enhancing interpretability. These constraints lead to more realistic predictions, which are crucial in safety-critical autonomous driving contexts. Additionally, the use of prior-driven, class-specific interaction layers fosters interpretability, as the correlation in prior difference and prediction error could aid in monitoring the network for incorrect and unaligned predictions. The step-wise integration of our changes in Table 5.13 shows that each layer enhances the model performance after an initial deterioration due to the kinematic layer. In sum, while our model sacrifices a small amount of accuracy, it more than compensates with feasible and interpretable predictions better suited for practical applications where aligned predictions are crucial.

6 Conclusion and Outlook

Trajectory prediction is an essential problem in making roads safer with autonomous driving. It enables collision-free planning based on the anticipated movements. Despite significant advances through deep learning, current state-of-the-art trajectory prediction does not always align with human expectations due to deep learning's data-driven nature.

Previous approaches to addressing this issue have often relied on hybrid prediction methods that integrate expert knowledge into deep neural networks. This expert knowledge can be derived from rule-based models like the Social Force Model for pedestrian interactions or kinematic models for the physics of motion. However, previous works do not leverage this expert knowledge to interpret the network's predictions and often focus specifically on either vehicles or pedestrians.

To address these limitations, this thesis posed the following research question: *Can integrating prior knowledge improve the alignment of trajectory predictions with human expectations by enhancing feasibility and interpretability?* In response, we propose integrating expert knowledge of interactions and kinematics to align the predictions with human reasoning. For interpretability, we introduce an agent-to-agent interaction embedding layer guided by a rule-based prior for interaction importance. For kinematic feasibility, we add a kinematic layer at the end of the network. It takes predicted control inputs from the network to compute trajectories using kinematic models, which guarantee that the trajectories are kinematically feasible by enforcing realistic limits on the control inputs.

To evaluate our approach, we divided our research question into two parts. First, we investigated whether integrating prior knowledge improves the interpretability of learned interactions. We found that HPTR learned to depend on interactions that do not align with human reasoning. Further analysis showed that these misaligned interactions correlate with erroneous predictions. However, integrating our proposed DG-SFM prior resulted in interaction attentions better aligned with human intuition. Importantly, integrating the prior did not decrease model accuracy and established a stronger correlation between prediction errors and deviations in the network's attention weights from the prior-based importance scores. Compared to the SKGACN and L2 priors, our DG-SFM prior better suited human intuition and improved model accuracy.

Second, we examined whether enforcing kinematic feasibility individually for each agent class is worthwhile. Analysis of the dataset revealed infeasible noise and showed that 88% of HPTR's predictions are physically infeasible. The analysis substantiated the need for kinematically feasible prediction, as otherwise, models will learn infeasible noise. This improves their metrics but leads to misaligned predictions. Our kinematic layer addressed this issue by enforcing feasibility, leading to better-aligned predictions for all agent classes through tailored kinematic models. Particularly for pedestrians, we enhanced alignment with our double integrator model beyond what has been achieved in the state-of-the-art. The reproducibility analysis showed that the double integrator model strikes a better trade-off between being able to reproduce all pedestrian movements and restricting trajectories to physically feasible ones.

While our model advances the alignment of trajectory prediction, there are many directions one could take for future improvement. First, although we achieved better alignment and feasibility guarantees, our modifications do come at an accuracy cost compared to the original HPTR, as studied in Section 5.4. Future work could address this by propagating HPTR's Gaussian position representation through the kinematic layer, similar to Trajectron++ [48]. This adaptation would enable training our model with HPTR's negative log-likelihood loss, facilitating improved accuracy and a level comparison between our model and HPTR.

Another limitation concerns the potentially limited influence of the agent-to-agent layer on the prediction. As suspected in Section 5.2.2, our model might not always rely equally on high-quality interaction information. This complicates the correlation analysis between prediction errors and the Prior-To-Attention Difference. Future studies could explore positioning the agent-to-agent interaction layer as the final embedding layer before the prediction head. In our model, the anchor-embedding layer following our agent-to-agent layer impeded our interpretability analysis.

Finally, limited by the time and scope constraints in this thesis, our evaluation of the interaction prior's effect opened interesting research directions. Further studies could explore what concrete advantages result from better-aligned interaction modeling. For example, improved generalization or more robust handling of rare scenarios are plausible consequences. Therefore, future research could employ adversarial evaluation techniques, such as provoking socially unacceptable predictions or predictions that misinterpret map context, following methods in [47] and [5]. Additionally, evaluating the accuracy in distinct categories of maneuvers would provide a more nuanced performance assessment in challenging real-world scenarios.

A List of Figures

B List of Tables

C Bibliography

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.

[2] M. Antonello, M. Dobre, S. V. Albrecht, J. Redford, and S. Ramamoorthy. Flash: Fast and Light Motion Prediction for Autonomous Driving with Bayesian Inverse Planning and Learned Motion Profiles. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9829–9836, 2022.

[3] A. Antonucci, G. P. R. Papini, L. Palopoli, and D. Fontanelli. Generating Reliable and Efficient Predictions of Human Motion: A Promising Encounter between Physics and Neural Networks. *IEEE Access*, 10:144–157, 2022.

[4] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz. On the nonholonomic nature of human locomotion. *Autonomous Robots*, 25(1-2):25–35, 2008.

[5] M. Bahari, S. Saadatnejad, A. Rahimi, M. Shaverdikondori, A. H. Shahidzadeh, S.-M. Moosavi-Dezfooli, and A. Alahi. Vehicle Trajectory Prediction Works, but Not Everywhere. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17123–17133, 2022.

[6] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-Body Collision Avoidance. In *Robotics Research*, pages 3–19. Springer, 2011.

[7] F. Camara, N. Bellotto, S. Cosar, F. Weber, D. Nathanael, M. Althoff, J. Wu, J. Ruenz, A. Dietrich, G. Markkula, A. Schieben, F. Tango, N. Merat, and C. Fox. Pedestrian Models for Autonomous Driving Part II: High-Level Models of Human Behavior. *IEEE Transactions on Intelligent Transportation Systems*, 22(9):5453–5472, 2021.

[8] S. Carrasco Limeros, S. Majchrowska, J. Johnander, C. Petersson, and D. Fernández Llorca. Towards explainable motion prediction using heterogeneous graph representations. *Transportation Research Part C: Emerging Technologies*, 157:104405, 2023.

[9] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019.

[10] X. Chen, F. Luo, F. Zhao, and Q. Ye. Goal-Guided and Interaction-Aware State Refinement Graph Attention Network for Multi-Agent Trajectory Prediction. *IEEE Robotics and Automation Letters*, 9(1):57–64, 2024.

[11] H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, J. Schneider, D. Bradley, and N. Djuric. Deep Kinematic Models for Kinematically Feasible Vehicle Trajectory Predictions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10563–10569, 2020.

[12] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric. Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096, 2019.

[13] Z. Ding and H. Zhao. Incorporating Driving Knowledge in Deep Learning Based Vehicle Trajectory Prediction: A Survey. *IEEE Transactions on Intelligent Vehicles*, 8(8):3996–4015, 2023.

[14] Q. Du, X. Wang, S. Yin, L. Li, and H. Ning. Social Force Embedded Mixed Graph Convolutional Network for Multi-class Trajectory Prediction. *IEEE Transactions on Intelligent Vehicles*, pages 1–11, 2024.

[15] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov. Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9710–9719, 2021.

[16] X. Fan, S. Zhang, B. Chen, and M. Zhou. Bayesian Attention Modules. In *Advances in Neural Information Processing Systems*, volume 33, pages 16362–16376. Curran Associates, Inc., 2020.

[17] L. Feng, M. Bahari, K. M. B. Amor, É. Zablocki, M. Cord, and A. Alahi. UniTraj: A unified framework for scalable vehicle trajectory prediction. *arXiv preprint arXiv:2403.15098*, 2024.

[18] K. Fuerstenberg and J. Scholz. Reliable pedestrian protection using laserscanners. In *IEEE Proceedings. Intelligent Vehicles Symposium*, pages 142–146, 2005.

[19] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020.

[20] Ó. Gil and A. Sanfeliu. Human Motion Trajectory Prediction Using the Social Force Model for Real-Time and Low Computational Cost Applications. In *Robot 2023: Sixth Iberian Robotics Conference*, pages 235–247. Springer Nature Switzerland, 2024.

[21] H. Girase, J. Hoang, S. Yalamanchi, and M. Marchetti-Bowick. Physically feasible vehicle trajectory prediction. *arXiv preprint arXiv:2104.14679*, 2021.

[22] M. Golchoubian, M. Ghafurian, K. Dautenhahn, and N. L. Azad. Pedestrian Trajectory Prediction in Pedestrian-Vehicle Mixed Environments: A Systematic Review. *IEEE Transactions on Intelligent Transportation Systems*, 24(11):11544–11567, 2023.

[23] R. Graubner and E. Nixdorf. Biomechanical analysis of the sprint and hurdles events at the 2009 IAAF world championships in athletics. *Positions*, 1(10), 2009.

[24] D. Grimm, P. Schörner, M. Dreßler, and J.-M. Zöllner. Holistic Graph-based Motion Prediction. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2965–2972, 2023.

[25] D. Grimm, M. Zipfl, F. Hertlein, A. Naumann, J. Luettin, S. Thoma, S. Schmid, L. Halilaj, A. Rettinger, and J. M. Zöllner. Heterogeneous Graph-based Trajectory Prediction using Local Map Context and Social Interactions. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2901–2907, 2023.

[26] J. Gu, C. Sun, and H. Zhao. DenseTNT: End-to-end Trajectory Prediction from Dense Goal Sets. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15283–15292, 2021.

[27] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, 1995.

[28] S. Hossain, F. T. Johora, J. P. Müller, S. Hartmann, and A. Reinhardt. SFMGNet: A physics-based neural network to predict pedestrian trajectories. *arXiv preprint arXiv:2202.02791*, 2022.

[29] W. Huang, M. Fellendorf, and R. Schönauer. Social force based vehicle model for 2-dimensional spaces. In *91st Annual Meeting of the Transportation Research Board. Washington, DC, USA*, 2011.

[30] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen. A Survey on Trajectory-Prediction Methods for Autonomous Driving. *IEEE Transactions on Intelligent Vehicles*, 7(3):652–674, 2022.

[31] F. Janjoš, M. Dolgov, and J. M. Zöllner. Self-Supervised Action-Space Prediction for Automated Driving. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 200–207, 2021.

[32] R. Korbmacher and A. Tordeux. Review of Pedestrian Trajectory Prediction Methods: Comparing Deep Learning and Knowledge-Based Approaches. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):24126–24144, 2022.

[33] S. M. LaValle. Better unicycle models. *Planning Algorithms*, pages 743–743, 2006.

[34] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by Example. *Computer Graphics Forum*, 26(3):655–664, 2007.

[35] H. Li, Z. Liao, Y. Rui, L. Li, and B. Ran. A Physical Law Constrained Deep Learning Model for Vehicle Trajectory Prediction. *IEEE Internet of Things Journal*, 10(24):22775–22790, 2023.

[36] J. Li, H. Ma, Z. Zhang, J. Li, and M. Tomizuka. Spatio-Temporal Graph Dual-Attention Network for Multi-Agent Prediction and Tracking. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):10556–10569, 2022.

[37] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun. Learning Lane Graph Representations for Motion Forecasting. In *Computer Vision – ECCV 2020*, pages 541–556. Springer International Publishing, 2020.

[38] W. Liu, P. Zhou, Z. Zhao, Z. Wang, Q. Ju, H. Deng, and P. Wang. K-BERT: Enabling Language Representation with Knowledge Graph. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(03):2901–2908, 2020.

[39] K. Lv and L. Yuan. SKGACN: Social Knowledge-Guided Graph Attention Convolutional Network for Human Trajectory Prediction. *IEEE Transactions on Instrumentation and Measurement*, 72:1–11, 2023.

[40] K. Margatina, C. Baziotis, and A. Potamianos. Attention-based Conditioning Methods for External Knowledge Integration. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3944–3951. Association for Computational Linguistics, 2019.

[41] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[42] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp. Wayformer: Motion Forecasting via Simple & Efficient Attention Networks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2980–2987. IEEE, 2023.

[43] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268, 2009.

[44] O. Rainio, J. Teuho, and R. Klén. Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, 14(1):6086, 2024.

[45] E. Rehder and H. Kloeden. Goal-Directed Pedestrian Prediction. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 139–147. IEEE, 2015.

[46] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras. Human Motion Trajectory Prediction: A Survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

[47] S. Saadatnejad, M. Bahari, P. Khorsandi, M. Saneian, S.-M. Moosavi-Dezfooli, and A. Alahi. Are socially-aware trajectory prediction models really socially-aware? *Transportation Research Part C: Emerging Technologies*, 141:103705, 2022.

[48] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Dynamically-Feasible Trajectory Forecasting with Heterogeneous Data. In *Computer Vision – ECCV 2020*, pages 683–700. Springer International Publishing, 2020.

[49] J. Schmidt, P. Huissel, J. Wiederer, J. Jordan, V. Belagiannis, and K. Dietmayer. RESET: Revisiting Trajectory Sets for Conditional Behavior Prediction. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–8, 2023.

[50] C. Scholler, V. Aravantinos, F. Lay, and A. Knoll. What the Constant Velocity Model Can Teach Us About Pedestrian Motion Prediction. *IEEE Robotics and Automation Letters*, 5(2):1696–1703, 2020.

[51] E. Schuetz and F. B. Flohr. A Review of Trajectory Prediction Methods for the Vulnerable Road User. *Robotics*, 13(1):1, 2024.

[52] J. Schulz, C. Hubmann, N. Morin, J. Lochner, and D. Burschka. Learning Interaction-Aware Probabilistic Driver Behavior Models from Urban Scenarios. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1326–1333. IEEE, 2019.

[53] P. Schwab, D. Miladinovic, and W. Karlen. Granger-Causal Attentive Mixtures of Experts: Learning Important Features with Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4846–4853, 2019.

[54] S. Shi, L. Jiang, D. Dai, and B. Schiele. Motion Transformer with Global Intention Localization and Local Movement Refinement. In *Advances in Neural Information Processing Systems*, volume 35, pages 6531–6543. Curran Associates, Inc., 2022.

[55] 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*, 46(5):3955–3971, 2024.

[56] H. Song, D. Luan, W. Ding, M. Y. Wang, and Q. Chen. Learning to Predict Vehicle Trajectories with Model-based Planning. In *Conference on Robot Learning*, pages 1035–1045. PMLR, 2022.

[57] B. Tang, Y. Zhong, U. Neumann, G. Wang, S. Chen, and Y. Zhang. Collaborative Uncertainty in Multi-Agent Trajectory Forecasting. In *Advances in Neural Information Processing Systems*, volume 34, pages 6328–6340. Curran Associates, Inc., 2021.

[58] A. Tato and R. Nkambou. Infusing Expert Knowledge Into a Deep Neural Network Using Attention Mechanism for Personalized Learning Environments. *Frontiers in Artificial Intelligence*, 5, 2022.

[59] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov, and B. Sapp. MultiPath++: Efficient Information Fusion and Trajectory Aggregation for Behavior Prediction. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7814–7821, 2022.

[60] A. Vaswani. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[61] M. Vitelli, Y. Chang, Y. Ye, A. Ferreira, M. Wolczyk, B. Osinski, M. Niendorf, H. Grimmett, Q. Huang, A. Jain, and P. Ondruska. SafetyNet: Safe Planning for Real-World Self-Driving Vehicles Using Machine-Learned Policies. *2022 International Conference on Robotics and Automation (ICRA)*, pages 897–904, 2022.

[62] W. Wang, L. Wang, C. Zhang, C. Liu, and L. Sun. Social Interactions for Autonomous Driving: A Review and Perspectives. *Foundations and Trends® in Robotics*, 10(3-4):198–376, 2022.

[63] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays. Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting. *arXiv preprint arXiv:2301.00493*, 2023.

[64] World Health Organization. *Global Status Report on Road Safety 2023*. World Health Organization, 2023.

[65] T. Xia, Y. Wang, Y. Tian, and Y. Chang. Using Prior Knowledge to Guide BERT's Attention in Semantic Textual Matching Tasks. In *Proceedings of the Web Conference 2021*, WWW '21, pages 2466–2475. Association for Computing Machinery, 2021.

[66] J. Zębala, P. Ciępka, and A. Reza. Pedestrian acceleration and speeds. *Problems of Forensic Sciences*, 91:227–234, 2012.

[67] P. Zhang, J. Xue, P. Zhang, N. Zheng, and W. Ouyang. Social-Aware Pedestrian Trajectory Prediction via States Refinement LSTM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2742–2759, 2022.

[68] W. Zhang, H. Cheng, F. T. Johora, and M. Sester. ForceFormer: Exploring Social Force and Transformer for Pedestrian Trajectory Prediction. *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7, 2023.

[69] Z. Zhang, A. Liniger, C. Sakaridis, F. Yu, and L. V. Gool. Real-Time Motion Prediction via Heterogeneous Polyline Transformer with Relative Pose Encoding. *Advances in Neural Information Processing Systems*, 36:57481–57499, 2023.

[70] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, C. Li, and D. Anguelov. TNT: Target-driven Trajectory Prediction. In *Proceedings of the 2020 Conference on Robot Learning*, pages 895–904. PMLR, 2021.