

REVIEW

Deep-learning-based vehicle trajectory prediction: A review

Chenhui Yin¹ | Marco Cecotti²  | Daniel J. Auger² | Abbas Fotouhi² | Haobin Jiang³

¹School of Automotive and Traffic Engineering,
Jiangsu University, Zhenjiang, China

²Advanced Vehicle Engineering Centre, Cranfield
University, Cranfield, UK

³Automotive Engineering Research Institute, Jiangsu
University, Zhenjiang, China

Correspondence

Marco Cecotti, Advanced Vehicle Engineering
Centre, Cranfield University, Cranfield, UK.
Email: m.cecotti@cranfield.ac.uk

Funding information

Graduate Research and Innovation Projects of
Jiangsu Province, Grant/Award Number:
KYCX21_3334; China Scholarship Council,
Grant/Award Number: 202108690001

Abstract

Vehicle trajectory prediction enables autonomous vehicles to better reason about fast-changing driving scenarios and thus perform well-informed decision-making tasks. Among different prediction approaches, deep learning-based (DL-based) methodologies stand out because of their capabilities to efficiently summarise historical data, infer nonlinear behavioural patterns from human driving data, and perform long-horizon prediction. This work reviews the DL-based methods that have shown promising results, organising them in terms of usage of the input data, separating the encodings of the target vehicle's historical data, surrounding vehicle's historical data, and road layout data. In particular, this paper explores the relationships between the scope of the prediction components and the input data formats, as well as the connections with other elements in the same prediction framework, including vehicle interaction and road scene mining. This information is crucial to understand complex architectural decisions and to provide guidance for the design of improved solutions. This work also compares the performance of the most successful prediction models, establishing that appropriate encodings of vehicle interactions and road scenes improve trajectory prediction accuracy, with the best performance achieved by attention mechanism and Transformer-based models. Finally, this work discusses future research directions, including considerations for real-time applications.

1 | INTRODUCTION

Autonomous driving has the potential to improve road safety and traffic efficiency. To match the performance of experienced human drivers, autonomous vehicles (AVs) have to evaluate fast-changing driving scenarios, and predict the trajectories of surrounding vehicles on the road.

A simple approach for vehicle trajectory prediction is to rely solely on the current states of the target vehicles, based on the assumption that the vehicle's state transition is constant in the near future and the transition process, initialised by the vehicle's current states, is governed by the laws of physics. Models like Constant Velocity (CV) and Constant Turn Rate and Velocity (CTRV) are commonly used in this case [1]. Different types of estimators, like Bayes filters, are usually applied to deal with the uncertainties in the observation of vehicle current states [2]. These methods—based on the physics of expected and simplified vehicle dynamics—are computationally efficient but have poor long-term accuracy due to a lack of ability to consider the effects of the road layout, the interactions with other road

actors, and driving behaviours related to comfort, kindness, aggressiveness etc. Hence, they are often suitable in simple scenarios with short-term prediction horizons (less than 1 s) [3, 4].

To provide additional contexts about the vehicle manoeuvres, many methods take into consideration their historical states. One option is to infer the most likely motion pattern of the target vehicle and select corresponding prototype trajectories [5]. The set of motion patterns includes possible manoeuvres at a given location. For a known road structure, motion patterns and prototype trajectories can be identified before the prediction tasks, based on road geometry or recorded driving data [6]. In the prediction stage, historical trajectories are matched with the predefined motion patterns and associated prototype trajectories are selected to represent the future trajectories of the target vehicle [7, 8]. One limitation of this method is the difficulty of generalising prototype trajectories for different road layouts because they are heavily dependent on a specific road geometry. Also, this approach cannot capture dynamic interactions with surrounding vehicles, leading to a loss of crucial context.

This is an open access article under the terms of the [Creative Commons Attribution](https://creativecommons.org/licenses/by/4.0/) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2025 The Author(s). *IET Intelligent Transport Systems* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

In the past few years, DL-based methods have gained attention for their ability to process complex traffic scene contexts, model interactions with surrounding vehicles and between vehicles and the road infrastructure. This contributes to effective long-term predictions. This article focuses on DL-based vehicle trajectory prediction methods and seeks to provide a critical overview. In this work, the term prediction backbone is used to refer to the prediction component that focuses on the historical data of the target vehicle, although it could include additional data when they are inherently associated together at the sensor level, like when processing bird-eye view photos. This element is present in the vast majority of prediction methods, distinguishing it from optional architectural components like the explicit encoding of vehicle interactions or road layout.

This topic has previously been reviewed in several publications. Mozaffari et al. [9] have summarised DL-based methods and have discussed their superior capabilities in traffic scene understanding and trajectory prediction compared to conventional methods based on physics or classical machine learning [6], especially for a long prediction horizon. In their work, DL-based methods are classified according to input representation, output types and prediction methods. More specifically, Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs) and their combinations are considered in detail. However, there is limited coverage of research works covering the application of Graph Neural Networks (GNNs), despite the fact that they have been proven to be powerful in both vehicle interaction and map structure encodings in recent years. In [3], DL-based and reinforcement learning-based (RL-based) methods are discussed, along with the physical-based and classical machine learning-based methods that have been mentioned in [6]. In these works, significant attention has been paid to the classification of DL-based prediction methods in terms of Sequential Networks, GNNs, and Generative Models. Details of the deployment of specific NNs are well summarised. These methods are also discussed by Gomes et al. [10]. In each classification, the discussion is carried out in terms of how vehicles' intentions and interactions are represented and used in trajectory prediction. Huang et al. [11] have reviewed the last 5 years' works on trajectory prediction and have summarised their improvements on the backbone prediction methods following the general representation of vehicles and map elements, integration of dynamic and static scene information and enhancement of prediction interpretability. The general framework of prediction methods is first introduced, including scene information, the scene context encoder, and the motion predictor. The commonly used input representations, like arranging scene information in grids or graphs, are also well illustrated in their work. However, there is no strong exploration of the relationship between the input representation format and the model chosen for input encoding.

It is a fact that the same NN class could be used in multiple parts of one trajectory prediction architecture, but with different performances and purposes. For example, CNNs could be used to extract temporal features of the vehicles. They could also be applied in the encoding of vehicle interactions and the extraction of map features, with minor modifications between

these two applications. These applications of CNNs rely heavily on the format of input data and the type of features, temporal or spatial, that are expected to be extracted. In order to inform the reader of the performances of the best DL models and the latest research trends, this work aims to analyse the relations among the architectures of the DL-based models for use in vehicle trajectory prediction, the prediction goals, the input information, and the corresponding data formats. The analysis is organised in terms of trajectory prediction backbones, vehicle interaction mining and road scene mining.

This paper contributes to the body of literature with:

1. the review of popular DL-based vehicle trajectory prediction methods, focusing on the evolution of the methodologies according to the enhanced importance of vehicle historical states.
2. the analysis of the relations between the representation formats of driving scenario data and the chosen architectures in terms of trajectory prediction backbones, vehicle interaction mining and road scene mining, as shown in Figure 1.
3. the comparison of the performance of the state-of-the-art DL-based trajectory prediction models, leading to some discussion on future research directions.

In the rest of this paper, Section 2 introduces the basic concepts of DL-based vehicle trajectory prediction. Section 3 summarises the fundamental architectures of NNs used in the trajectory prediction tasks. Section 4 reviews DL methods used in trajectory prediction in terms of trajectory prediction backbones, vehicle interaction mining, road scene mining and vehicle-road fusion. The evaluation metrics and performance of state-of-the-art prediction models are compared in Section 5. Finally, the conclusion and the potential research gap are discussed in Section 6.

Based on this structure, Section 3 introduces various NN architectures including FFs, CNNs, RNNs, AMs, TFs, GFs, and GNNs. The characteristics, strengths and weaknesses of these methods are discussed with respect to trajectory prediction tasks.

Section 4 categorises commonly used NNs for DL-based trajectory prediction into trajectory prediction backbones, vehicle interaction mining, road scene mining and their fusion. For trajectory prediction backbones, NNs are introduced with progressively enhanced capability for summarising sequential data. Subsequently, the reformatted traffic scene representations, including surrounding vehicles and map data, are discussed alongside suitable NNs for vehicle interaction mining, road scene mining and their comprehensive fusion. The strengths and limitations of each NN in specific applications are analysed and summarised in tabular form.

Section 5 evaluates state-of-the-art prediction models using evaluation metrics such as RMSE, minADE and minFDE, while also analysing the computational complexity of certain methods. Before delving into detailed comparisons, multi-modal prediction is introduced as many of the reviewed approaches provide multi-modal results. The comparison emphasises the advantages of explicitly mining vehicle interactions and

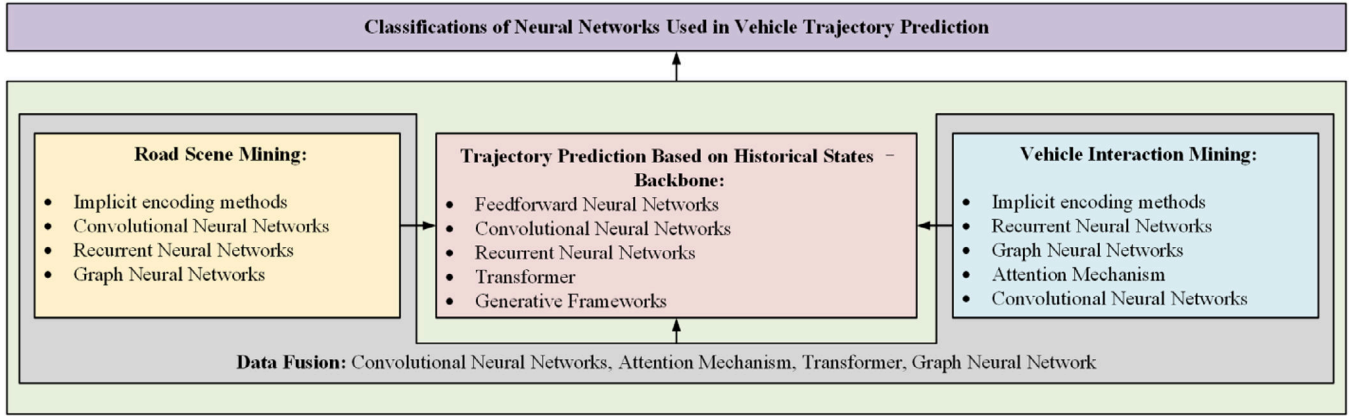


FIGURE 1 An overview of classification in this research.

map information. Notably, attention-based methods like the attention mechanism and transformer demonstrate superior trajectory prediction accuracy, flexible structural designs and relatively low computational complexity.

2 | PROBLEM FORMULATION OF VEHICLE TRAJECTORY PREDICTION

The prediction of vehicle trajectories aims to forecast the future motion states of target vehicles over a fixed-length horizon, based on the available information of the driving scenario acquired by the autonomous ego vehicle (EV). More specifically, the motion states are usually coordinates of the target vehicles (TVs) at consecutive time steps.

In the driving scenario, the EV perceives the surrounding environment including other vehicles and road scenes. Among other vehicles, those vehicles whose trajectories are to be predicted are defined as TVs. Surrounding vehicles (SVs) are defined as vehicles around the TVs which might have an influence on their moving motions. The remaining vehicles are noted as non-effective vehicles (NVs) which are assumed to have no influence on the TVs' behaviour.

With a fixed historical horizon, the observation of vehicles (i.e. the road objects) is represented by O_{obs} , and other traffic scene information regarding road layouts and traffic rules (i.e. the road scene) are defined as S_{obs} . The vehicle trajectory prediction aims to obtain the motion states Y_{TVs} of the TVs over the prediction horizon based on O_{obs} and S_{obs} :

$$Y_{TVs} = \left\{ Y_{t+1}^i, Y_{t+2}^i, \dots, Y_{t+p}^i \right\}_{i=1}^N, \quad (1)$$

where Y_t^i is the predicted motion state of the i th TV at time step t , N is the number of TVs and p is the length of the prediction horizon. Typically, Y_t^i represents the 2D coordinates (x_t^i, y_t^i) of the TV, with respect to the EV or a static map. Considering the prediction uncertainty, the prediction task can be expressed in a conditional probability form: $P(Y_{TVs} | O_{obs}, S_{obs})$.

Due to the finite number of allowed vehicle manoeuvres at a given location, a multimodal prediction model can be used to inform the EV of the evolution and likelihood of different trajectories. A multimodal prediction produces several trajectory predictions at one time, indicating multiple possible sets of future vehicle motion states. In this case, the multimodal motion states of each TV could be represented by:

$$Y_{TV} = \{ Y_{TV}^1, Y_{TV}^2, \dots, Y_{TV}^K \}, \quad (2)$$

where K denotes the total number of modalities.

3 | FUNDAMENTAL DEEPING LEARNING ALGORITHMS FOR TRAJECTORY PREDICTION

In order to realise the trajectory prediction, as well as vehicle interaction mining and road scene mining, the following deep learning methods are often used: Feedforward Neural Networks (FFs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Attention Mechanisms (AMs), Transformers (TFs), Generative Frameworks (GFs), and Graph Neural Networks (GNNs).

FFs are basic NNs and are also known as Multilayer Perceptrons (MLP) or Back-Propagation (BP) neural networks [12]. They are generally used in vehicle trajectory prediction for the extraction of latent features in high-dimensional space and the conversion of high-dimensional data into outputs with physical significance. Their application as trajectory prediction backbone is summarised in Section 4.1.1.

The processing of sequential data, like historical trajectories, presents however unique challenges. RNNs specialise in processing this type of data and capturing long-term dependencies that FFs are not designed to handle. In practice, gated RNNs, like Long Short-Term Memory (LSTM) [13] and Gated Recurrent Unit (GRU) [14], are commonly deployed in historical trajectory encoding and future trajectory generation, considering their superior performance among RNNs against

gradient vanishing or gradient explosion in the backpropagation of training. The application of RNNs in trajectory prediction is summarised in Sections 4.1.3 and 4.2.2.

While RNNs are efficient at capturing dependencies of sequential data, CNNs excel in extracting spatial features [15]. They are able to explore spatial interrelationships by processing image-like measurements or bird-eye-view (BEV) representations of road scenes, resulting in direct trajectory prediction or improving the prediction in cooperation with RNNs by exploring spatial relationships between vehicles and road layouts and among traffic agents. The detailed application cases are summarised in Sections 4.1.2, 4.2.5, 4.3.2 and 4.4.1.

Currently, AMs are popular in the application of trajectory prediction. They were originally proposed to improve the performance of the encoder-decoder translation model. The original encoder compresses the sequential inputs iteratively with a forgetting mechanism for the decoder as a fixed-length vector [16], which inevitably causes information loss due to its expressive limitation. The AMs resolve this issue by enabling the decoder to visit each element of the input sequence and focus on particular parts [17]. The AMs can be further classified into self-attention and cross-attention, which learn self-presentation and allow knowledge integration separately. The TF is based on these two attention methods [18], within an encoder-decoder structure. Multi-head attention, a stacked of multiple self-attentions, is applied to extract a weighted sequence representation and cross-attention is used to transfer information from the encoder to the decoder. The application of AMs and TFs is presented in Sections 4.1.4, 4.2.4 and 4.4.2.

In trajectory prediction, GFs are usually used to describe the underlying ambiguity of future behaviours of vehicles by introducing a stochastic latent variable while generating the prediction. The most commonly applied GFs in trajectory prediction are Generative Adversarial Network (GAN) [19] and Conditional Variational Auto Encoder (CVAE) [20]. GAN consists of a generator and a discriminator. The generator is used to generate data with a combination of inputs and latent variables. These generated data are then mixed with the ground truth and the role of the discriminator is to estimate the probability for a sample of mixed data to originate from the ground truth set rather than the generated set. The performance of the generator and discriminator can be iteratively improved and the probability of the discriminator making a mistake will be maximized at last [21]. Though a latent variable is also used in CVAE, its distribution is learnable. CVAE usually consists of an encoder and a decoder. In the encoder, a recognition network is used to produce the latent variable space, based on the input during model training. In the decoder, latent variables are selected randomly from the produced latent variable space and are then combined with the input to generate the stochastic predictions. Details in the application can be found in Section 4.1.5.

Other useful NNs for trajectory prediction are the GNNs. GNNs are usually used when traffic elements are organised by non-Euclidean metrics, like the interactions among vehicles and the connections within the road layouts. In a graph, traffic elements serve as graph nodes and node edges are created by their (topological, semantic, legal, emotional etc.) interconnections.

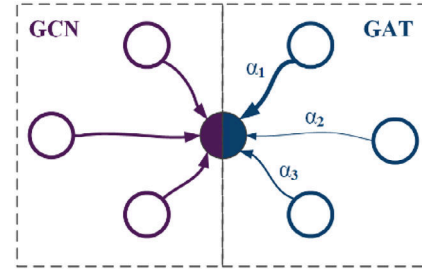


FIGURE 2 Illustration of message passing in GCN and GAT separately. The same structure is used in GCN and GAT, a host node aggregating information from its three neighbour nodes without self-connection. In GAT, the aggregated node features from neighbours are assigned different weights. The higher the weights are, the more important the neighbour nodes are. In this illustration, α is the weight and $\alpha_1 > \alpha_3 > \alpha_2$.

GNNs update the node features by message passing. Graph Convolution Network (GCN) and Graph Attention Network (GAT) are two significant branches of message passing. GCNs are inspired by traditional CNNs and the nodes update themselves by collecting features of their neighbours and performing aggregations like averaging and pooling [22], which are similar to the operations inside the CNN kernels. On the other hand, GATs assign different weights to the neighbours of a node by computing attention coefficients based on the features of each pair of nodes [18]. These coefficients are then used to update node features by a weighted aggregation, allowing the model to focus more on the most relevant information. An illustration of message passing in GCN and GAT is shown in Figure 2. Details of relevant applications can be found in Sections 4.2.3, 4.3.3 and 4.4.3.

4 | CLASSIFICATION OF DL-BASED TRAJECTORY PREDICTION METHODS

In this section, the trajectory prediction backbone is first reviewed in Section 4.1 to show a general framework of NN methods. Then NN methods for vehicle interaction mining and road scene mining are discussed in Sections 4.2 and 4.3 separately. At last, methods fusing vehicle interaction and road scene information are introduced in Section 4.4.

4.1 | Basic neural networks for trajectory prediction backbone

4.1.1 | Feedforward neural networks

FFs project the historical states of the vehicles into a high-dimension hidden space for feature extraction and then infer future trajectories based on these hidden states [23, 24] as shown in Figure 3.

The historical observations of the TVs usually include speeds, steering angles and positions in the historical horizon. In [23], these data are fed into an FF directly, consisting of one input layer, one hidden layer and one output layer. The output of the

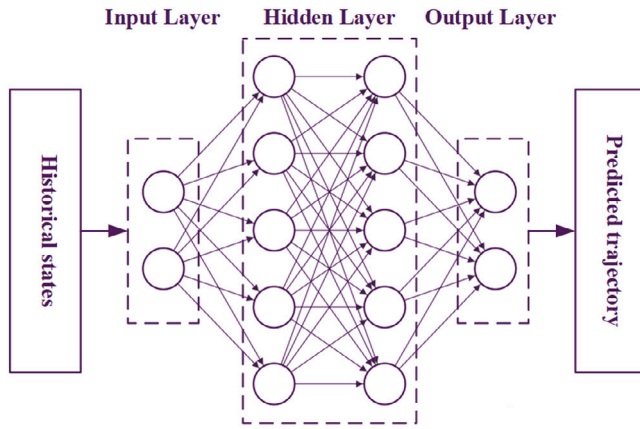


FIGURE 3 Illustration of vehicle trajectory prediction methods based on FF.

FF is as simple as a lateral position point at a specific horizon, for example, the lateral position of the TV at a given time (e.g. 1 s) in the future. Every time step, the FF processes a new set of historical samples and thus the predicted trajectory is continuously updated while the prediction model is in operation.

Though the FF is simple in structure and efficient in execution, the prediction horizon is constrained to a short horizon (e.g. 1 s) for its limited performance in exploring time-series dependencies of vehicle historical states.

4.1.2 | Convolutional neural networks

In trajectory prediction, CNNs are often associated with data inputs in the form of point clouds or RGB images. These data are common outputs of detection sensors and could be further processed to create a BEV scenario describing both the static and dynamic scenes. The CNNs are then applied to extract spatio-temporal features and predict the trajectory in a short-term horizon [25, 26].

To facilitate the processing of temporal features from spatial data, the raw data from the sensors are usually concatenated with each other across historical time frames. FaF [25] uses a bunch of convolution layers to predict the future tracklets, in the form of consecutive bounding boxes along the prediction horizon. 3D point cloud data are arranged in voxel representation (as shown in Figure 4) and grouped with an extended temporal dimension, ending up with a 4D tensor encoding an occupancy grid of the 3D space over several time frames. To process temporal features of the data, 1D convolution could be used to do data aggregation across the temporal dimension. After compression in the temporal dimension, the height dimension is treated as the channel dimension and 2D convolutions are applied across the X and Y dimensions to extract spatial features of the whole scenario.

IntentNet [26] modified FaF's architecture by converting the voxel representation into an elevation map and concatenating the time dimension to the height information, ending up with

2D grids. In this case, 3D convolutions are replaced by 2D convolutions to extract temporal and spatial features. A similar 2D convolution is applied to exploit semantic information from the map of the road layout. And the outputs of these two convolution backbones are concatenated together and fed to the fusion subnetwork for intention estimation and final trajectory prediction.

Following this work, Casas et al. [27] extend the single-branch network of PIXOR [28] to a two-stream network, such that the first stream processes LiDAR point clouds and the other processes road-layout maps, and their outputs are then concatenated along the channel dimension. This concatenation is further used to produce the bounding boxes of vehicles and to predict vehicle trajectories.

These researches unify the tasks of object detection, tracking, and short-term motion forecasting within an end-to-end fully convolutional approach. One limitation is that they are only performing well with a short prediction horizon, typically around 1 s.

CNNs are valued in vehicle behaviour prediction for their capabilities of taking raw sensor data, like point clouds, as input and exploring their short-term temporal dependencies and spatial relationships. Though RGB images from cameras are also common inputs to AVs, they are usually processed through CNNs to perform object detection instead of trajectory prediction because pure CNNs are not efficient at extracting TV's temporal states and encoding them for trajectory prediction purposes. Thus, CNNs are seldom used as prediction backbones with RGB image inputs.

4.1.3 | Recurrent neural networks

RNNs are often used to extract features that represent the motion of the TV across a historical horizon. The inputs are usually historical state sequences, including positions and velocities.

A single LSTM is used in [29] to predict the future position of the TV as a regression task. The purpose of the LSTM layer is to abstract a meaningful representation of the input time series, while its output is fed into dense layers to extract the final prediction: the future longitudinal velocity and lateral position. Additionally, the present states of the TV are also fed directly to the dense output layer, bypassing the LSTM, to allow the model to focus on variations from the current states. In [30], in addition to the trajectory prediction LSTM module, another LSTM module is used to extract the lateral driving intentions of the TV. This intention encoding is then concatenated with the sequential longitudinal features of the TV and fed to the trajectory prediction LSTM module, which predicts the lateral deviation and longitudinal acceleration across the prediction horizon. In [31], an encoder-decoder structure is used, as shown in Figure 5. The encoder works as an observation stage, extracting features from historical data that are fed to the decoder to generate the prediction trajectory.

To improve the encoding of the historical data, LSTMs could be stacked together. Park et al. [32] use an encoder-decoder

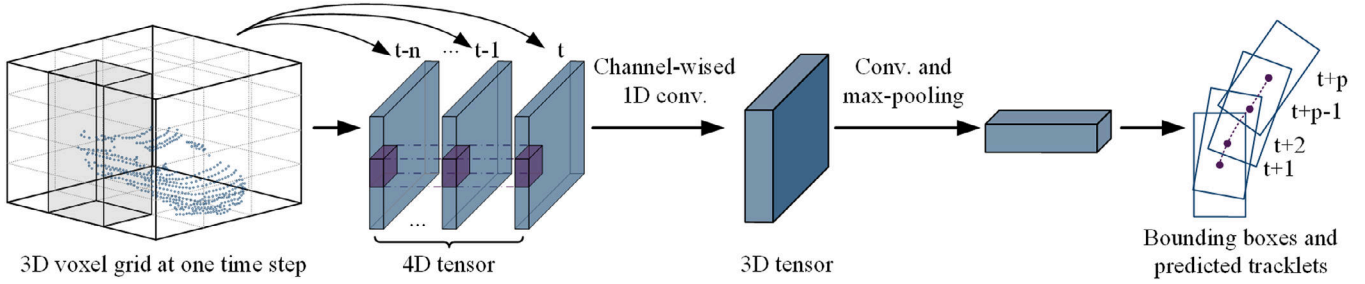


FIGURE 4 Illustration of trajectory prediction pipeline based on historical point cloud input and CNN methods. Point clouds at each time step are arranged in the 3D voxel grid, and historical voxel grids are concatenated in temporal sequence to form a 4D tensor. To collapse the temporal dimension, height information in the voxel grid is treated as channels in the 2D tensor, and 1D convolution is deployed to process the spatial data across the temporal dimension, equivalent to producing a single point cloud from all frames by weighting their contribution [25].

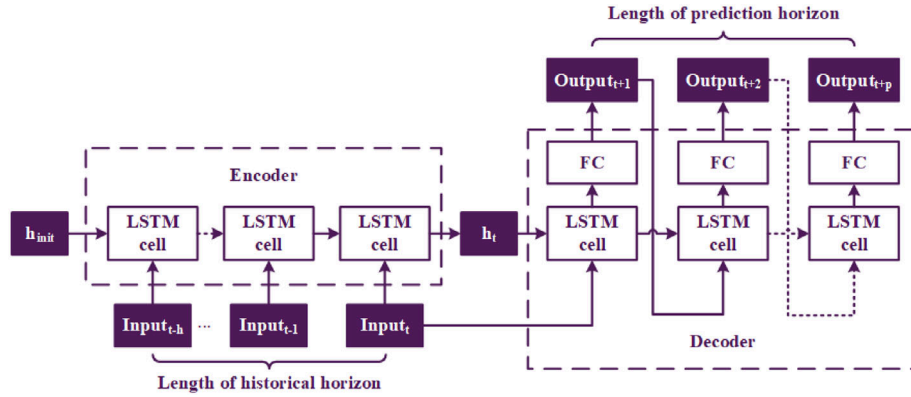


FIGURE 5 Illustration of LSTM prediction model with an encoder-decoder structure. The input to the encoder is a sequence of historical vehicle state data and the output is a fixed-length vector summarizing of the whole input. This vector is used as the initial state of the LSTM in the decoder, along with the current state of the vehicle. The decoder runs recursively to create the output sequence step by step.

LSTM to predict the probability of occupancy on a grid map and apply a beam search algorithm to select the most probable future trajectory candidates. The LSTM encoder consists of two stacked LSTM layers and takes the latest trajectory samples from the SVs, as well as the state information of the EV, to produce a fixed-length vector which captures the temporal structure of the historical trajectory. Based on this fixed-length vector, the LSTM decoder, also a 2-layer stacked LSTM, generates the future trajectory on the occupancy grid map (OGM).

Similar to the method in [30], Deo and Trivedi [33] proposed a separate manoeuvre classification module based on a single LSTM to predict manoeuvre class probabilities based on the backbone of the encoder-decoder structure. These manoeuvre encodings are appended with the vehicle trajectory encodings from the trajectory encoder. Then the decoders, different for each manoeuvre, use this information to predict the final position of the TV.

Although RNNs are some of the main NNs associated with data time series analysis and prediction, they have some deficiencies in modelling spatial relationships such as vehicles' spatial interaction and processing image-like data such as road scene contexts. This explains why sophisticated solutions using RNNs usually exploit additional methods to compensate for the weakness of a single RNN.

4.1.4 | Transformer

Another neural network structure that is skilled at learning historical data is the Transformer (TF). One difference in processing sequential data between TF and LSTM is that TF needs additional positional encodings in the input because it is permutation-invariant and cannot distinguish the time flow [18].

The TF is generally made up of an encoder and a decoder, as shown in Figure 6. In trajectory prediction, the encoder creates a summary of the complete time-series input, for example, the historical states, by constructing vectors of queries (Q), keys (K) and values (V) through self-attention. The decoder runs recursively to predict future positions by focusing on appropriate information in the input as well as previously generated predictions [34]. Alternatively, in the decoder, the historical states of the target agent could serve as the input as well [35].

In [36], a structure of dual TFs is used to predict future trajectories of the TV. The inputs of the first TF are the historical positions of the TV and the interaction information, consisting of relative states of the SVs, whereas its output is the manoeuvre intention. This intention information is concatenated with the input of the historical trajectory of the TV and fed to the encoder and decoder of the second TF to produce future predictions.

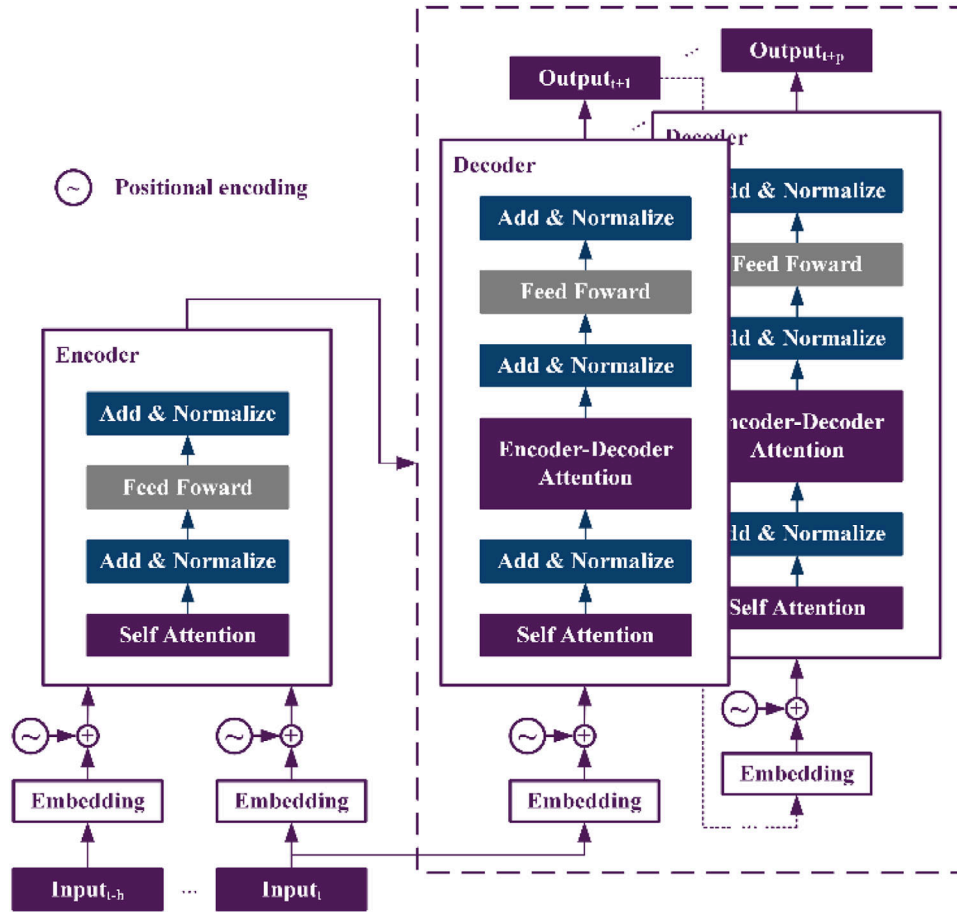


FIGURE 6 Illustration of vehicle trajectory prediction model based on TF.

LSTM may undergo gradient vanishing or information loss while back-propagating the signal sequentially from the current to the earliest time step, but TF could learn from every single input of the historical data because it keeps the complete time-series encoding. Results have shown that the ability of TF has outperformed LSTM in pedestrian trajectory prediction [34].

4.1.5 | Generative frameworks

When treating trajectory prediction as a regression work, one of the limitations of pure RNNs is that they cannot consider the multi-modal character of human drivers. Future predictions can be inherently ambiguous and have uncertainties as multiple plausible scenarios can be explained under the same past situation (e.g. a vehicle heading toward an intersection with decreasing velocities can make turns or just perform giving way). Thus, learning a deterministic function that directly maps historical data to deterministic output will under-represent the potential prediction space and easily overfit training data. Moreover, a naively trained network with a simple loss will produce predictions that average out all possible outcomes, for example, the model tends to do a prediction which is an average perfor-

mance of right-turning and going-through trajectories while the vehicle is driving in a right-turn and straight lane.

GFs solve these issues by introducing stochastic latent variables in the prediction models. The most commonly applied GFs in trajectory prediction are Generative Adversarial Network (GAN) and Conditional Variational Auto Encoder (CVAE). Their structures are shown in Figure 7.

Lee et al. [37] have created a CVAE-based trajectory prediction method based on GRU. In the training phase, the CVAE encoder encodes the past and future trajectories of an agent through two separate GRUs. Then these two encodings are concatenated to produce the mean and standard deviation values in the latent variable space with FFs. In the decoder, the latent variable produced by the encoder is reshaped to be combined with the encoding of the past trajectory through an element-wise multiplication and is fed into a GRU to generate the candidate prediction trajectory. In the inference phase, the encoding of the past trajectory is combined with randomly sampled latent variables to produce multiple candidate prediction trajectories. In [38], a similar method is proposed. The difference is that LSTM layers are used and, in the CVAE encoder, the trajectories of SVs are added to the trajectory of the TV to consider potential interactions.

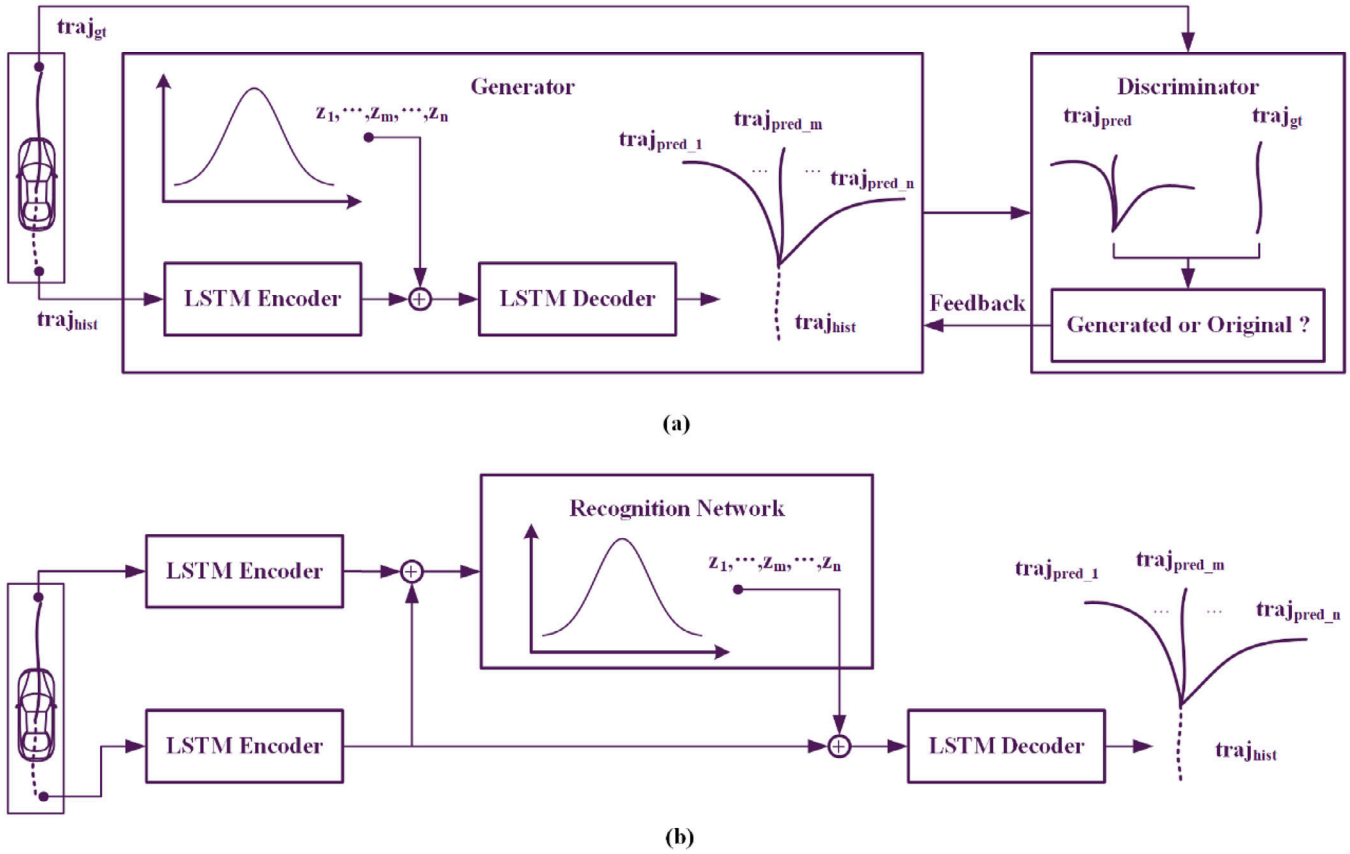


FIGURE 7 Illustration of GAN and CVAE-based trajectory prediction in (a) and (b) separately. (a) The GAN-based method randomly samples the latent variable with a known distribution, for example, Gaussian distribution. (b) The CVAE-based method creates latent variable space through training. In the testing stage, latent variables are sampled from this learned distribution of the latent variable space.

In terms of combinations of CVAE and CNN, Dulian et al. [39] have used a CNN-based encoder to extract features of road scenario data, including separate layers of drivable areas, lanes and road actors. The latent variable space is then created based on these layer encodings, along with the agent state encoding. In [40], the encoder is LSTM-based and similar to that in [38], and a U-Net-like CNN [41] is used to process the road scene context, working as part of the CVAE decoder.

A summary of each backbone NN can be found in Table 1.

4.2 | Neural networks for interaction mining

Vehicles move on the road according to the actions of SVs, adjusting the heading angles and velocities to avoid potential crashes and negotiating the space to improve traffic efficiency. To this end, if the interactions between the TV and its SVs could be modelled effectively, the trajectory prediction accuracy will likely improve.

SVs are usually identified as having a potential influence on the motion of the TV. Adjacent vehicles in the eight directions around the TV are usually identified as direct neighbours and considered SVs [33, 38]. Often, the vehicle in front of the forward vehicle is also identified as SV [29]. See the vehicle ff in Figure 8a.

Taking into account the moving speeds, a rectangle field centred at the TV is considered a region of interest (RoI) and all moving objects in this field are considered SVs [44, 45]. In this case, the RoI is converted into a grid and vehicles inside each grid cell are identified as SVs and their information is assigned to the corresponding grid cells, as shown in Figure 8c. The above definitions of SVs are usually used in simple scenarios, where the SVs move in the same direction as the TV, like highways.

However, this neighbour topology does not cater to the urban scenario or unstructured roads, where SVs might move transversally in the direction of the TVs and the interaction between the TV and SVs is generally more complicated. In [46, 47], the SVs are identified in a polar coordinate system. More generally, a graph-like neighbour topology could be established as shown in Figure 8b.

4.2.1 | Implicit encoding

After identifying the SVs and their dynamic information, the interaction between the SVs and the TV could be explored. One solution is to concatenate the information of the SVs with the information of the TV as a comprehensive input to the prediction model. This method fuses the SVs' interaction with the TV in an implicit way.

TABLE 1 A summary of backbone architectures.

NNs	Inputs	Works	Pros	Cons
FF	historical state sequences	[23, 24]	<ul style="list-style-type: none"> Simple in structure and efficient in execution Useful to project mined traffic scene context into final trajectory prediction 	<ul style="list-style-type: none"> Limited performance in exploring time-series dependencies of agents' historical states
CNN	Point clouds	[25–28]	<ul style="list-style-type: none"> Capable of keeping the spatial relationship of the traffic scene input data, enabling modelling of the interactions between agents and the road scene context 	<ul style="list-style-type: none"> Only performing well with a short prediction horizon, typically around 1 s, because of its limitation on extracting temporal features
RNN	historical state sequences	[29–33]	<ul style="list-style-type: none"> Powerful in data time series analysis and prediction 	<ul style="list-style-type: none"> Deficiencies in modelling spatial relationships such as agents' spatial interaction and road scene contexts Undergoing gradient vanishing or information loss while back-propagating the signal sequentially from the current to the earliest time step
TF	historical state sequences	[34–36, 42]	<ul style="list-style-type: none"> Skilled at learning historical data from every single input of the historical data Easy to realise multi-modal prediction 	<ul style="list-style-type: none"> Needing additional positional encodings in the input because it is permutation-invariant and cannot distinguish the time flow Having more trainable parameters and achieving limited performance with a small training dataset
GF	historical state sequences	[37–41, 43]	<ul style="list-style-type: none"> Introducing stochastic latent variables in the prediction models to ease the production of multi-modal predictions The distribution of the latent variables can be predefined or learned 	<ul style="list-style-type: none"> The selection of latent variables in the inference stage might be arbitrary, and not able to model the uncertainty in driving intentions, which might produce unrealistic trajectories latent variables are assumed to have a specific distribution which might not aligned with the traffic scene

For the SVs, the most used information is the position. Both global coordinates [24, 33] and coordinates relative to the TV [29, 32] are used. The relative coordinates are easier to be recognised by human drivers and sensors on the vehicle. Meanwhile, the speeds of SVs are also used as input for the prediction model [24, 29, 32]. Other information is also sometimes used in the inputs, such as the heading angle of the vehicles [31], yaw rate [32], acceleration [24] and time to collision [29].

Since the speed and acceleration are the first-order and second-order derivatives of the position, NNs could extract these features from the historical position data. But providing them directly allows for a reduction of the model size and the training requirements. There is some research using the control inputs of the TV like the steering wheel angle [23], but this information is not easily measured for SVs.

In the implicit fusion of data about the TV and SVs, no explicit process of computing the interaction feature exists, despite using states that are relative to the TV as direct inputs. While this approach is easy to implement, it has certain limitations. NNs may struggle to effectively capture the interactions among the TV and SVs from the complex input arrays, which include the original states of both the TV and SVs. This challenge becomes more pronounced when dealing with high-dimensional input arrays that incorporate variables like speed,

position and heading, especially when a large number of SVs are involved.

Thus, analysis of the interaction between vehicles as a separate procedure and producing a high-dimensional intermediate encoding can be combined with the TV states to facilitate trajectory prediction [48]. Such high-dimensional intermediate encoding is produced by analysing the spatial relationship between the TV and its SVs by models like CNNs and GNNs. This is more explicit than simply appending the original states of SVs to the model input. Apart from the spatial relationship among SVs, the historical features are also extracted by pre-processing the historical states with models like CNNs and RNNs.

4.2.2 | Recurrent neural networks

With identified SVs, the TV could be paired with each one of them to model the respective interaction. In [49], this pair-wise interaction is encoded by LSTM. Firstly, two different LSTM layers are used to separately encode the historical trajectories of the TV and its SVs. Then the historical encoding of the TV is paired with that of the SVs. Further, this concatenation is processed by a stacked LSTM to further encode the impact of

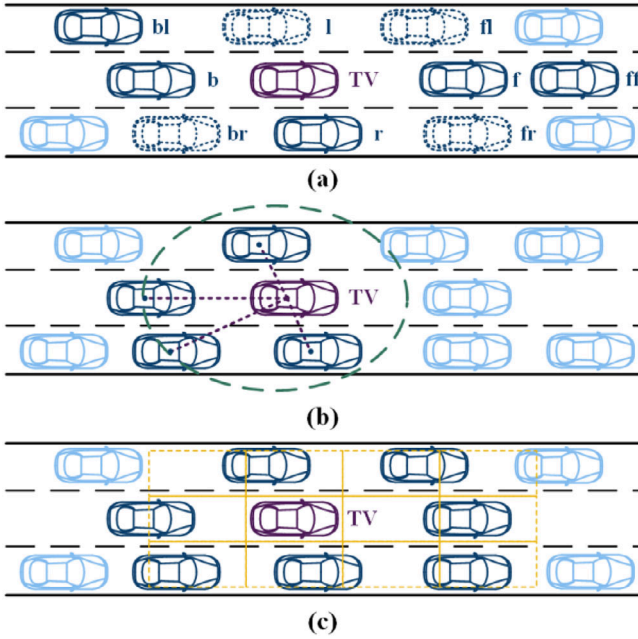


FIGURE 8 Illustration of the identification methods of SVs. The TV is in purple; the identified SVs are in dark blue and the irrelevant road actors are in light blue. In (a), eight SVs are identified based on whether they are directly adjacent to the target vehicle. The SVs are marked according to their relative positions to the target vehicle including behind (b), forward (f), left (l) and right (r). Some positions might not be covered by any SV, as expressed by the dashed contours. In (b), SVs are identified based on their Euclidean distances relative to the TV. In (c), a space grid is created and vehicles inside the grid cells are identified as SVs.

spatial interaction between the TV and its SVs. In [50], GRU encoders are used to model the pairwise interaction between the TV and each of the SVs. In each pair, two GRUs with shared parameters are used to encode the individual manoeuvre histories and dynamics. The two GRU encodings are then concatenated to represent the pair-wise interaction.

4.2.3 | Graph neural networks

If the SVs are identified with a graph-like structure, GNN could be used to propagate the information between the TV and its SVs. The TV and SVs in this structure could form a graph, where each of them serves as graph nodes and their potential interactions form the graph edges. The connection could be bidirectional or unidirectional. An illustration of graph-like vehicle interaction is shown in Figure 8b.

In [27], a fully connected directed graph is created based on potential interactions between the TV and its SVs, as well as the interactions among the SVs. The road actors are represented by bounding boxes and act as nodes in the graph. The node states are the convolution results of the fixed-size feature map of vehicle nodes. During message passing, the concatenation of two connected nodes is first created and then processed by MLP to produce the message. Then the node updates its states by aggregating messages from all of its local neighbour nodes with an ordering-invariant, feature-wise max operator. By this feature-

wise max operation, the model could figure out the interplay for each pair of vehicles in a bidirectional fashion. In [51], GAT is used to characterise the interdependences of heterogeneous traffic agents, including vehicles, bicycles and pedestrians. Traffic agents are treated as graph nodes, and within an identified TV and SV pair, the attention coefficient is calculated by matrix product based on the feature encodings of the nodes. Then the node feature is updated by the weighted sum of its neighbours. GAT is also used in [52] while the attention coefficient is calculated by normalized attention operation among all connected neighbour nodes [53]. In [54], convolutional layers are used to extract node features and GCN is used to aggregate node features from neighbours for the purpose of encoding interactions. In [55], the node features from the neighbours are weighted by the reciprocal of the sum of the distance between the two nodes. This operation is designed to emphasize the influence of the SVs by their relative distance to the TV. The adjacency matrices of most GNN-based interaction mining methods are established through prior knowledge, like setting matrix values to be reciprocals of geographic distances. This static matrix lacks the adaptive learning capability to depict the evolution of agent interactions through time. In [56], a learnable adaptive graph is used to resolve this deficiency. More specifically, an adjacency matrix is initialised by the element-wise multiplication of the learnable embedding array of all agent nodes and their transpose. Then softmax is used to determine if a connection exists.

With these graph-like structures, the spatial relationships among the TVs and SVs can be explored. However, this structure only establishes spatial connections among vehicles at the current time frame. In [57, 58], to integrate the influence of the neighbours both in spatial and temporary dimensions, a spatio-temporal graph is created where edges across the historical time frames and through the spatial space are established. More specifically, at each time step, spatial edges are created among spatial-adjacent traffic agents; for the same traffic agent, temporal edges are produced to connect its states in adjacent frames. Based on these temporal and spatial edges, an instance layer is created for the message passing and the interaction capture among individual nodes. The interaction among traffic agents is modelled by a weighted sum of neighbour features, which is further concatenated with the temporal feature to update the node feature. Additionally, a category layer is established to identify traffic agents of the same type and a super node is used to summarise the shared behaviours of the same category by averaging node features in the whole layer. This feature is then transmitted back to the instance layer to form final interaction encoding, by connecting individual nodes to its corresponding super node. A simplified spatio-temporal graph can be established by creating a separate spatial graph at all temporal time steps [59, 60]. The spatial graph describes the interaction among vehicles and graphs across the time domain enable a dynamic description of interaction variation. To better evaluate the diverse influence of each SV across the time domain, an adaptive adjacency matrix is used in [59]. In the adjacency matrix, the edge weights between one node and its neighbours are calculated by normalized attention operation based on node-neighbour encoding

concatenations. This adaptive adjacency matrix has the ability to capture interactions among vehicles that are not explicitly covered by intuitive manual linkages [53].

4.2.4 | Attention mechanism

The influence of the SVs on the TV is not identical. For example, the SV following the TV might have less influence on the decision-making of the TV than the SV in front or the SV that wants to merge into the lane of the TV. To identify the SVs that have more influence on the TV and better encode interactions, attention-based methods could be used. These methods often work in a kind of one-directional information transfer way, in which the interactive features of SVs are aggregated and linked to the TV.

In [61], the historical states of all agents in the scene are encoded with a sequential combination of 1D convolution and a variation of GRU. Then these encodings are used to create the Q, K and V vectors for the AM. The Q vector is generated based on the TV's encoding, while the K and V vectors are based on the SVs' encodings. The normalized dot product of Q and K vectors creates an attention map and then this attention map is used to pool their value features into a context vector, which encodes the influence of the SVs on the TV. A similar structure is used in [62]. GRU is used to encode historical states of the vehicles in the scene and the final results of SVs are assembled into a global embedding matrix. Then attention weights among the TV and SVs are calculated by dot production of the TV's hidden states and the generated global embedding matrix, which is further used to mine the interaction among vehicles. In [63], the historical states, future trajectories and reference path waypoints are encoded with a temporal convolutional layer and LSTM. Those encodings are fed into the linear layers for mapping Q, K and V vectors in the attention. A self-attention structure is then used to capture the interactions between agents in the historical time domain. A similar strategy is also used in [64]. Considering that the SVs are arranged in the occupancy grid, K and V vectors are mapped from the grid map instead of individual SVs in [65].

4.2.5 | Convolutional neural networks

Another popular solution to deal with interactions from multiple neighbours is to use social pooling [48, 66]. For this operation, a grid map is created, which is centred on the TV. Then each cell in the grid map is filled with states of SVs. The vehicle states could be their original dynamic information or high-dimensional states, processed by NNs like LSTMs. If multiple SVs occupy the same grid cell, the grid cell will be populated with aggregation of their states by an operation like adding. This social pooling preserves the spatial information of SVs and enables spatially proximal road actors to share information with the TV.

These grid cell states are concatenated end to end before being fed into the prediction model, therefore losing their

spatial organisation. To overcome this issue, CNN is used to capture the spatial inter-dependencies of the social pooling tensor. In [45, 67], convolutional and pooling layers are used to learn the locally useful features within the spatial grid of the social tensor. Two convolutional layers and one pooling layer are used to obtain the social context encoding. In [68], an additional MLP is used to process the output of the pooling layer to obtain the final interaction vector. Multi-Agent Tensor Fusion (MATF) encoding and decoding is introduced in [69]. In the encoding part, a social tensor, augmented with convolutional encoded scene context channels, is fed to a U-net-like [41] fully convolutional network to fuse interaction among agents and between agents and scene context while keeping spatial locality. This combined method is also used in [44], but the AM is used to select the most noteworthy SVs at each time step. In [46], the SVs around the TV are identified in the polar coordinate. The relative position and velocity are concatenated with each vehicle's LSTM hidden state and processed independently by an MLP. Finally, the MLP output is max-pooled element-wisely to compute the pooling vector of the TV. In this way, the inter-dependencies of the motion of SVs are captured without being restricted to a specific grid size.

A summary of NNs for agent interaction mining is illustrated in Table 2.

4.3 | Neural networks for road scene mining

Vehicle motion is constrained by map-relevant information such as road structures and traffic rules. For example, the vehicles will follow the geometry of driving lanes and manoeuvre according to the lane connectivity and priority rules. More and more prediction models consider the fusion of map information to improve prediction accuracy.

4.3.1 | Implicit encoding

One way to explore the influence of road geometry on trajectory prediction is to represent the vehicle states relative to the road elements. In [70], the index of the occupied lane and the deviation from the lane centerline are considered part of the model input. In [30], the relative displacement to the lane markings and lane centre lines, as well as the first and second-order derivatives, are used as inputs of an LSTM model.

4.3.2 | Convolutional neural networks

One popular method to process road scene contexts, including road geometries and traffic dynamics, is to form an image-like representation of traffic scenarios and deploy CNNs for spatial and temporal feature extraction. Basic ideas for creating image-like representations are to discretise the road scene around the TV into regular grids in BEV and further digitise the road scene with numerical values or RGB values to form multiple-channel grid maps or rasterisation images [61, 71–73]. In [72, 74], the

TABLE 2 A Summary of interaction mining architectures.

NNs	Inputs	Works	Pros	Cons
Implicit	Matrices to contain both the TV's and SVs' states	[24, 29, 31–33]	<ul style="list-style-type: none"> No extra NNs needed to process interactions Easy to arrange data and simple in structure 	<ul style="list-style-type: none"> Challenging to capture interactions because the input vector might be too large for the NNs to effectively extract and summarise the relevant features
RNN	historical state sequences	[49, 50]	<ul style="list-style-type: none"> Simple to implement 	<ul style="list-style-type: none"> Restricted to mine pair-wise interaction
GNN	Agent node feature matrix and adjacency matrix	[27, 51–60]	<ul style="list-style-type: none"> Flexible to define connections among vehicles Able to build connections across both the spatial and temporal dimensions Flexible to choose message passing technologies, such as multi-step aggregation, bi-directional or unidirectional aggregation, attention-based aggregation 	<ul style="list-style-type: none"> Inefficient storage because the number of SVs might vary and sometimes this ends up with a very sparse matrix
AM	Matrices of TV's and SVs' states	[61–64]	<ul style="list-style-type: none"> Able to mine the interaction according to the learned importance of SVs Ordering-independent arrangement of SVs 	<ul style="list-style-type: none"> Additional NNs needed to extract features before being taken as AM input
CNN	Grid maps with SV's states	[41, 44–46, 48, 66–68]	<ul style="list-style-type: none"> Efficient in the operation of feature extraction Able to preserve the spatial information of SVs and enables spatially proximal road actors to share information with the TV and capture the spatial inter-dependencies of the social pooling tensor 	<ul style="list-style-type: none"> Requirements of representing SVs in a grid-like structure Computational intensity in converting SVs' information to a fine-grained image

RGB images are used and they are created through rasterisation, where different static contexts of the vector road-layout map are assigned distinct RGB colours.

In [72], a multiple-layer rasterisation image is created to represent the traffic scenario around the TV. The traffic scenario is represented by pieces of vector layers, representing traffic elements like driving lane polygons, road boundary polylines and directional lane centreline polylines. The generated layer is discretised into grids and grid cells overlapping with the traffic elements are assigned distinct RGB values to make them prominent, as shown in Figure 9b. At last, a rasterisation image could be obtained by stacking RGB images on top of each other starting from layers that represent macro areas such as road polygons to layers that represent finer structures like lane centrelines, where the subsequent images will overlay the content of the previous one. A similar technology is also used in [61]. Different from overwriting subsequent images, a multi-channel image is created to store the rasterised road scenes without loss of spatial features. For example, the completeness of the lane markers will not be affected when they are obstructed by agent boxes since these two features are stored in different channels.

The main difference between multi-channel grid maps and RGB images is that the grid cells in grid maps are populated with numerical values instead of RGB values. This mechanism allows grid cells to store dynamic traffic contexts like traffic light states, which is not easy for RGB representation. Also, predefining orderings while stacking RGB images could be avoided.

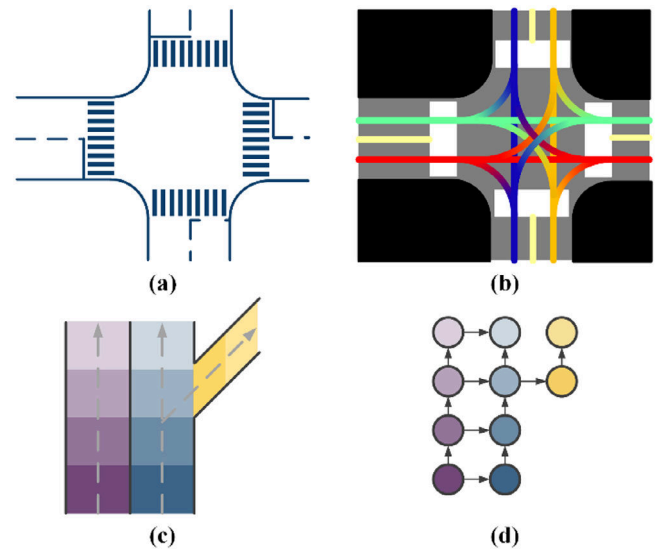


FIGURE 9 Road layout maps could be represented in rasterized images or map graphs, as shown in (b) and (d) separately. The rasterized map in (b) is created from (a), where the drivable areas (in grey), lane markings (lane markings in light yellow, pedestrian crossings in white) and lane centrelines are filled with identical colours. The map graph in (d) is created from (c), showing the traversability of lane segments.

In [26], road geometries and traffic rules, including traffic lights and lane types, are encoded by 17 binary maps in which the spatial grid cells are assigned with 1 or −1 if they are crossed

by lane markings or ruled by traffic rules. This parametrisation of the map allows the model to extract rich scene information from the geometric details together with traffic rule information. In [75], to enable variable receptive fields in the grid maps, single-channel grid maps with three different discretisation resolutions are created to represent the scene. The three grid sizes are classified into ‘local’, ‘medium’ and ‘global’ to describe the scenario from micro trajectory features to the whole scenario context. In a grid map with a specific resolution type, grid cells are filled with values representing the severity of collision with the occupying traffic agents, and the constraint intensity of traffic rules on vehicles. Cases of high severity of potential collisions and highly constraining traffic rules are assigned with high numerical values. For example, motorcycles are given higher values than cars because they are more vulnerable when involved in an accident. And solid line markings have bigger values than dashed line markings because the solid ones are more restrictive.

The advantage of using image-like representation is that CNNs can efficiently and effectively abstract RL map features [56] in a broad spectrum of road scenes, ranging from simple highways to complex urban environments. Nevertheless, some limitations lie in road connectivity representation, topology-guided feature exploration and real-time application. First, image-like representation is not able to semantically describe lane connection types of lane pairs [76], for example, it is not easy to distinguish adjacent lane pairs in the same or opposite directions. Additionally, 2D convolution may not be able to capture global topology features among road lanes because of the limited receptive fields of the convolution kernel. This could be partly resolved by generating images with larger grid sizes [75], which might cover more global connectivity information in the convolution kernel. However, the efficiency of this solution is still limited. Second, topology-guided feature exploration cannot be realised, like the feature extraction along a specific lane. Expansion on the reception field could cover more intended areas along a target lane but will inevitably contain more redundant regions since the kernel treats the image pixels homogeneously. Lastly, the continuous discretisation of elements around the TV poses a challenge for real-time applications [27, 77] because it is time-consuming and should be conducted at each time step.

4.3.3 | Graph neural networks

In autonomous driving, road-layout maps are usually served as prior. They are arranged with a set of lane segments and their connectivity in a vectorised way, providing structured representations of the scenarios with geometric and semantic pairs or with geometry entities alone. To better use this geographic and semantic information and preserve the map element connectivity, these vectorised road-layout maps are usually converted to a directed graph for further feature extraction, as shown in Figure 9d.

A primary map graph could be constructed based on actors’ traversability across the connected lane nodes. These lane nodes usually consist of lane segments with similar lengths. A lane seg-

ment length of 10–20 m could represent the roads on the map at a macro scale, while a length of 1 m or even smaller represents the roads at a micro scale [64]. These lane segments often contain a sequence of ordered lane centrepoin- [78]. Since these sequential centrepoin- capture both the direction of the traffic flow and the legal routes that each driver can follow, their physical characteristics are often used as the lane node features [79].

Graph nodes

The basic lane node is described by features inherited from lane centrepoin- [76, 79, 80]. Semantic information like traffic control elements could also be added to the node features as a supplement to indicate whether the centrepoin- lie on a stop line or crosswalk [79], to distinguish turning directions [80] and to show traffic light statuses [81].

Further, lane node features sometimes include road actors’ information. To show the accessibility of a lane node, the spatio-temporal occupancy status of the lane segments is incorporated into the lane node features based on the historical trajectories of road actors [80]. In [81], apart from the above-mentioned geometric and semantic features, a lane node is matched with its nearby road actors, and these road actors’ positions and orientations relative to the matched lane segment centres over their historical horizon are also added to the node feature. In this way, the lane node encodes the road actors’ manoeuvre intention into its features, like approaching or leaving the matched lane node. In [79], lane node encodings are updated with nearby agent encodings using multi-head attention. In [76], spatial attention is used to capture actor-map interactions. Road actors within a spatial range of the selected lane node are extracted. The lane node features and road actor features are concatenated together and then projected into the new lane node feature. With this spatial attention, real-time traffic information is introduced into lane nodes, modelling road blockages and availability of the lanes.

Apart from the lane centrepoin-, vectors of polyline elements in road-layout maps could also be treated as graph nodes. In [77], a node is a polyline vector consisting of sequential points from lane boundaries (or motion trajectories). The node features are the start and end locations of each polyline, along with other attributes such as polyline ID and semantic labels.

Graph edges

Individual lane nodes are linked by edges in the graph, representing the lane segment connectivity in road-layout maps [82]. According to the adjacency of the lane segments in the map, edges have types of predecessors, successors, and left and right neighbours. Given a lane segment, its predecessor and successor are other lane segments that can directly travel to and from the selected segment. Left and right neighbours refer to the lane segments which can be directly reached without violating traffic rules in the traversal directions [76]. In [79], an idea of proximal edges is also used, which accounts for lane-changing manoeuvres between neighbouring lane segments. In this

application, a distance threshold is used to identify the neighbour lane segments and their orientations are checked to be within a yaw threshold of the ego lane segment to avoid erroneous assignment of nodes when multiple lanes overlap.

The edges are often stored in the form of an adjacency matrix. A value of one in the p row and q column represents that node p and node q have linkage, and vice versa. In [76, 81], four edge matrices instead of one edge matrix are used to store linkage information. Storing linkage information of all types of connections in separate matrices enables non-interfering exploration of individual connection types.

Graph

A map graph could be constructed in a global way by connecting all lane nodes in the scenario according to the map topology.

A fully connected graph is simple to construct but has high computational cost and it ignores specific semantic connections between elements.

To better explore local scenes and global topology, graph layers of both micro and macro levels are created simultaneously to encode information on a different scale. The global graph preserves the map structure and the local graph captures the more fine-grained information within a small region. In [80], two separate map graphs are established to focus on local traversability and global topological information separately. The nodes in the local graph are sampled points on the drivable area with a resolution of 1 m around the target road actor, noted as the drivable area (DA) layer. And the global graph is created based on the lane segment centreline with a resolution of 2 m, noted as the lane segments (LS) layer. The DA and LS layers are from the same map data and thus always spatially overlapped. To realise better local-global feature fusion, inter-layer edges between nodes of these two layers are established by matching nodes in the LS layer with their Euclidean neighbours in the DA layer. This connection enables message propagation forward and backward between these two layers. In [77], a hierarchical graph network is established. Based on the idea of constraining the connectivity of the graph based on the spatial and semantic proximity of the nodes, a subgraph is established with nodes in the same polyline sharing the same semantic labels. Then, all local graphs are fully connected with each other to form a global graph for information exchange. In [81], a local graph named LaneRoI is built by nearby lane nodes of the TV and extensions of these nearby lane nodes along the graph topology. Multiple LaneRoIs are then connected to form a global graph. This topology could better represent the local scenario context while maintaining awareness of global topology.

Message propagation

The lane node feature above only captures the local information of a lane segment. To aggregate the topology information of the lane graph at a larger scale, the encoded lane nodes exchange information by message propagation through the graph, updating lane node encodings by collecting encoding info from its surroundings.

In some applications, message propagation among lane nodes is processed by pooling methods. In [77], message propagation is done by concatenating the target node encoding with the max-pooling results of its neighbour node encodings. In [81], different local graphs (LaneRoI) can include the same lane node because of their spatial overlapping. To update the lane node feature, node features of all its neighbours, inside both its current local graph and other local graphs containing this node, are pooled based on MLPs. In [80], the features of the neighbour nodes are arranged in a stacked grid and a max-pooling operation is used to get the global feature inside the local DA graph. This max-pooled feature is then concatenated with the target node's feature and further processed by a fully connected layer.

Pooling methods usually lead to a one-step aggregation from the node's direct neighbours. For a multi-step aggregation from indirect neighbours with a given exploration depth, GCN could be used [64]. In [76], a graph convolution method is used to realise multi-scale message propagation along different directions. Because road actors usually move farther in the lane longitudinal direction than the lateral direction over the prediction horizon, the graph convolution operates within four separate adjacency matrices to enable exploration in predecessors, successors, left neighbours, and right neighbours with diverse depths. To accommodate the long-range dependency of lane node information, the graph convolution is further set to enable exploration with customised depth along the lane edges, mimicking the long-range exploration ability of the dilated 2D convolution operator by enlarging the receptive field. In [81], a similar multi-scale anisotropic graph convolution method is built but it operates inside each LaneRoI, a local graph embedding both actor and map information.

In [80], GAT is used to realise message propagation according to the importance of the neighbours. The attention weights are calculated based on encodings of node features and the node aggregates neighbour encodings through weighted sum.

A summary of NNs for road scene mining is illustrated in Table 3.

4.4 | Neural networks for fusion

Vehicles drive on the road, guided by the road geometries and layouts. These vehicles also have an influence on the dynamic states of the road scene, like the real-time traffic density. To form a comprehensive encoding vector which includes both vehicle and map information, the map encodings and agent encodings are commonly concatenated along a specific dimension [83, 84]. However, this simple concatenation does not well explore the internal relation between vehicles and maps, like the guidance of the road to the vehicle motion and the evolution of traffic densities.

To provide a map-aware agent encoding or an agent-aware map encoding, these two heterogeneous encodings could be fused by exchanging features with each other, as shown in [76]. Based on the literature, the most commonly used methods include CNNs, AM, TF and GNNs.

TABLE 3 A summary of road scene mining architectures.

NNs	Inputs	Works	Pros	Cons
Implicit	Vehicles' states relative to road elements	[30, 70]	<ul style="list-style-type: none"> • Simple in input data arrangement • No extra NNs needed to process road scene data • Efficient in simple road scenes like highways 	<ul style="list-style-type: none"> • Lack of awareness of the global road scene
GNN	Lane node features and adjacent matrices	[64, 76–82]	<ul style="list-style-type: none"> • Able to construct a heterogenous graph to contain both geometry and semantic information • Able to realise topology-guided feature exploration, like the feature extraction along a specific lane • Reduce time consumption because static road layout can be pre-processed and reused in the stage of training • Enable multi-step message passing with customised depth • Flexible to explore both local and global scenarios 	<ul style="list-style-type: none"> • Time-consuming and storage-inefficient to construct and store a dynamic graph at each time step
CNN	Rasterised images	[61, 72, 75, 74]	<ul style="list-style-type: none"> • Efficient and effective abstraction of map features • Uniform representation of traffic scene contexts like surrounding vehicles and road lane markings • Highly adaptive to a broad spectrum of road scenes, ranging from simple highways to complex urban environments 	<ul style="list-style-type: none"> • Time-consuming in scenario rasterisation • Have information loss because of spatial occlusion of different elements • Inefficient in capturing global topology features because of the limited receptive fields of the convolution kernel • Not able to perform topology-guided feature exploration
	Grid maps	[26, 75]	<ul style="list-style-type: none"> • Efficient and effective abstraction of map features • Allow storing traffic contexts like traffic light states • Avoid predefining orderings of different traffic scene context • More informative because channel layers can be customised • Highly adaptive to a broad spectrum of road scenes, ranging from simple highways to complex urban environments 	<ul style="list-style-type: none"> • Time-consuming in scenario discretisation • Inefficient in capturing global topology features because of the limited receptive fields of the convolution kernel • Not able to perform topology-guided feature exploration • Occupy more storage than RGB images

4.4.1 | Convolutional neural networks

CNN does the fusion based on a combination of representation of road scene and agent motion. With the rasterised road scene layers that have described the map elements [61, 72], traffic agents' dynamics are stacked additionally on top of the rasterised images. Each traffic agent is assigned a polygon to represent its pose at the current time step and multiple polygons of the same colour but with reduced brightness levels to represent its historical states. Then CNNs are used to process these images uniformly [74, 85] to encode the spatial correlation between the agent and the map elements. The limitations of these fusion methods are related to the difficulties of CNN in encoding temporal and semantic relationships, as discussed in Sections 4.2.5 and 4.3.2.

4.4.2 | Attention mechanism and transformer

In some sources, the encodings of traffic agents and road layouts are fused for a map-aware agent encoding or agent intention estimation.

Both AMs and TFs explore the interrelationships among vehicles and road layout elements by cross-attention, in which a map-aware agent encoding or agent intention estimation is generated. For example, Ngiam et al. [86] have used a TF-based method to enable the agent features to be updated by attending to the road graph. In this method, the Q vectors are generated from the agent encodings and K and V vectors are from the road graph embeddings. The map-aware agent encoding is calculated by cross-attention based on these Q, K and V vectors. Conversely, in other research works [64, 87, 88], vehicle information is projected into the lane node feature with similar approaches. For example, Guo et al. [87] map the Q vector from the centreline node feature encoding while obtaining K and V vectors from the vehicles' encodings. And then multi-head attention is used to calculate the weights of influence of different vehicles on each lane node, after which vehicle encodings are allocated according to their weights. In that way, the traffic flow density could be inferred in the model. A stacked structure is used in [42] to incorporate multiple channels of agent motion and traffic scene data. More specifically, three TFs are arranged in a hierarchy to produce an initial prediction trajectory proposal and fuse map and interaction information step by step.

AMs and TFs can also identify potential succeeding lane segments in the prediction horizon. For lane selection, Luo et al. [89] have fused the embeddings of the historical trajectory of the TV and centrelines of potential proceeding lanes to predict the probability of each potential lane that the vehicle might head for. The historical trajectory is encoded by LSTM while the centrelines are encoded by 1D convolution for efficiency. Then dot-product and a Softmax layer are used to calculate the final probabilities of each potential lane.

4.4.3 | Graph neural networks

When GNN techniques are used, map elements are usually arranged as nodes in the graph structure. The information exchange between vehicles and map elements could be operated by integrating vehicle information into the map nodes first and then propagating through the whole map graph.

Liang et al. [76] aggregate vehicle encodings to the lane node features to capture the real-time traffic information and then GCN is used to propagate this fused information over the lane graph. This approach is also applied in [87] and [80].

Alternatively, vehicles could also act as nodes in the graph to engage in message propagation directly with lane nodes.

Mo et al. [90] have used GAT to fuse vehicle and map information to model the effects of a TV's neighbours and centrelines on its future motion. A heterogeneous hierarchical graph containing both traffic agents and lane centrelines is created. These two types of node features are fused step by step. Three separate GATs are used to gather information from centrelines to the SVs, from the SVs to the TV, and finally from centrelines to the TV. By this sequential fusion, the TV obtains information both from SVs and road layouts.

Tang et al. [59] accumulate information from the TV's neighbourhood to the TV by GCN at each historical time stamp. The TV's neighbourhood information includes both SVs' historical trajectories and lane border segments. By message passing through both nodes of SVs and map elements, the TV generates a comprehensive neighbourhood context encoding. This dynamic graph is also used in [91]. A comprehensive adjacency matrix is created by extending the vehicle-vehicle adjacency matrix with road network connection information through a weighting factor, which determines the influence of the road network topology on the vehicle interaction graph.

A summary of NNs for traffic scene context fusion is illustrated in Table 4.

5 | PERFORMANCE COMPARISON

This section first introduces some commonly used quantitative evaluation metrics. Then the effectiveness of some state-of-the-art prediction models is compared.

5.1 | Evaluation metric

Commonly used displacement error metrics for single-modal trajectory prediction include Final Displacement Error, Aver-

age Displacement Error, Root Mean Square Error and Mean Absolute Error.

Final displacement error (FDE): This metric calculates the L2 distance between the predicted trajectory endpoint and the ground truth endpoint:

$$FDE = \sqrt{\left(x_{t+p} - x_{t+p}^{gt}\right)^2 + \left(y_{t+p} - y_{t+p}^{gt}\right)^2}, \quad (3)$$

where x_{t+p} and y_{t+p} represent the predicted 2D position coordinates of a TV at the last time step ($t + p$) of the prediction horizon. The superscript gt represents that the data are from the ground truth. The final trajectory prediction point is related to the TV's driving intention, so it is important to evaluate its accuracy.

Average displacement error (ADE): It is used to measure the overall distance deviation of the predicted trajectory to the ground truth:

$$ADE = \frac{1}{p} \sum_{i=1}^p \sqrt{\left(x_{t+i} - x_{t+i}^{gt}\right)^2 + \left(y_{t+i} - y_{t+i}^{gt}\right)^2}, \quad (4)$$

where x_t, y_t represent the predicted trajectory point at the timestep $t + i$. The same variables are used in later formulas.

Root mean square error (RMSE): This metric is sensitive to large prediction errors and is one of the commonly used metrics for trajectory prediction. It is the average sum of squared distances of total timesteps from a single predicted trajectory to the ground truth:

$$RMSE = \sqrt{\frac{1}{p} \sum_{i=1}^p \left[\left(x_{t+i} - x_{t+i}^{gt}\right)^2 + \left(y_{t+i} - y_{t+i}^{gt}\right)^2 \right]}. \quad (5)$$

Mean absolute error (MAE): It is the average absolute error between the predicted trajectory and the ground truth t , and is generally used in the horizontal and vertical directions:

$$MAE_x = \frac{1}{T_{pred}} \sum_{t=1}^{T_{pred}} \left| x_t - x_t^{gt} \right|, \quad (6)$$

$$MAE_y = \frac{1}{T_{pred}} \sum_{t=1}^{T_{pred}} \left| y_t - y_t^{gt} \right|. \quad (7)$$

The evaluation metrics commonly used for multimodal trajectory prediction include minFDE-K, and minADE-K, where K denotes the total number of modalities of the predicted trajectories. minFDE-K means the minimum FDE among K predicted trajectories. In Argoverse [92], minADE-K corresponds to the ADE of the predicted trajectory with the minimum FDE value, which aims to ensure the consistency of the optimal predicted trajectory.

Miss rate (MR): If the predicted trajectory's endpoint deviation from the true endpoint exceeds a threshold d for a sample, that sample is called a miss. The ratio of the total number of misses m to all samples M is called the miss rate. MR reflects

TABLE 4 A summary of fusion architectures.

NNs	Works	Pros	Cons
CNN	[61, 72, 74, 85]	<ul style="list-style-type: none"> • Efficient in feature extraction operation • Well explore the spatial relation between map elements and agents 	<ul style="list-style-type: none"> • Time-consuming and storage-intensive in scenario rasterisation • Hard to extract semantic constraints of the map elements on the vehicle motions
AM & TF	[64, 86–89]	<ul style="list-style-type: none"> • Allow information fusion among heterogeneous scenario encodings parallelly • Powerful in interaction extraction • Aggregate interaction information according to importance 	<ul style="list-style-type: none"> • Burden in cached storage while the sizes of the attended elements and their features are large
GNN	[59, 76, 80, 87, 90, 91]	<ul style="list-style-type: none"> • Enable direct information exchange by building a heterogenous graph • Provide customised connections among traffic scene elements, such as priority-aware and distance-aware connections 	<ul style="list-style-type: none"> • Time-consuming and storage-intensive to construct a dynamic graph at each time step

the overall closeness of the model's prediction of the trajectory endpoint. In Argoverse [92], a predefined fixed threshold d of 2 m is used.

To account for the influence of probability in the multi-modal prediction, minFDE and minADE can be further extended to p_minFDE and p_minADE , which add the probability of the best-predicted trajectory into evaluation:

$$p_minFDE = minFDE + \min(-\log(p), -\log(p_{thold})), \quad (8)$$

$$p_minADE = minADE + \min(-\log(p), -\log(p_{thold})), \quad (9)$$

where p_{thold} is set to be 0.05 in Argoverse [92].

Two other probability-based metrics are also commonly used:

$$brier_minFDE = minFDE + (1 - p)^2, \quad (10)$$

$$brier_minADE = minADE + (1 - p)^2. \quad (11)$$

In (8)–(11), p refers to the probability of the best-forecasted trajectory and thus accounts for uncertainty information in the evaluation.

In some research, predictions are represented as a mean value with Gaussian uncertainties. Negative Log Likelihood (NLL) is used to evaluate the prediction accuracy at the time step of $t + i$:

$$NLL = \frac{1}{2} (Y_{t+i} - Y_{t+i}^{\#})^T \Sigma^{-1} (Y_{t+i} - Y_{t+i}^{\#}) + \frac{1}{2} \ln \det(\Sigma) - \ln(2\pi). \quad (12)$$

In (12), Σ is the covariance matrix of the prediction:

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}, \quad (13)$$

where σ_x and σ_y are the variances of the predicted values x_{t+i} and y_{t+i} , respectively, and ρ is their correlation coefficient.

5.2 | Multi-modal prediction

Future trajectory predictions are inherently ambiguous and uncertain due to the diverse, unobservable intentions of each agent and the complex interactions within traffic scenarios. Even under similar conditions, agents' trajectories can vary significantly. To account for these uncertainties and enhance the accuracy of trajectory predictions, it is beneficial to predict a set of plausible trajectories rather than relying on a single deterministic outcome. This approach, aiming to infer multiple potential trajectories, is known as multi-modal trajectory prediction.

Multi-modal prediction methods are generally categorised into two types: end-to-end prediction and two-stage prediction, depending on whether to produce intermediate outputs regarding driving manoeuvres.

In the end-to-end prediction approach, K modalities of future trajectories can be generated simultaneously, either by assigning K prediction headers to decode the same traffic scene encoding [76, 78] or by generating K traffic scene encodings through the addition of unique latent variables before feeding them into a single prediction header [69, 93]. These two methods are referred to as multi-header-based and probability-based predictions, respectively.

The multi-header-based prediction can be seen as an extension of the uni-header prediction, commonly used for generating uni-modal trajectories, which makes this approach relatively straightforward to implement. However, it may suffer from modal collapse since all prediction headers share the same context encoding. On the other hand, probability-based prediction employs GFs like CVAEs [37]. Here, latent variables are randomly sampled from a predefined or learned distribution representing the modality distribution. This approach tends to generate statistically significant trajectory patterns observed in the training data, which might lead to modal collapse. Furthermore, the lack of physical interpretability in the latent variables can result in physically implausible predictions.

A more structured approach divides the prediction process into two stages. The first stage involves predicting intermediate outputs that represent potential driving manoeuvres, such as driving intention proposals, anchor trajectories, or final goals. The second stage uses these clues to generate or refine the trajectory predictions. These methods are known as proposal-based, anchor-based, and goal-based predictions, respectively.

Driving intention proposals typically include potential lateral (e.g. going straight, turning left or right) and longitudinal (e.g. accelerating, braking) motion patterns [45, 47, 94]. In proposal-based methods, the probabilities for each motion pattern are predicted first, and then these intermediate outputs are used to generate predictions for each pattern. Similarly, anchor-based methods use a set of possible trajectory segments, or “anchors”, to represent potential manoeuvres [85, 95]. These anchor trajectories, clustered from datasets, serve as priors. During prediction, the most likely anchor trajectories are selected and refined to produce the final predictions. Both approaches align well with physical manoeuvres and narrow the solution space. However, these predefined patterns may not suffice in more complex traffic scenarios. For example, stop-and-go motions may not be accurately captured by predefined proposals or anchor trajectories. This limitation can be partially addressed by predicting final goals, which are the positions an agent aims to reach by the end of the prediction horizon and reflect the agent’s driving intentions. Using these predicted goals in combination with traffic scene context encodings, complete trajectories can be generated to achieve multi-modal prediction [61, 81, 96].

The expressive ability of NNs in manoeuvre representation is constrained by the predefined patterns in proposals and anchor trajectories. With advancements in TF models, multi-modal predictions have become more feasible through the use of learnable proposals, which are represented in latent space. These proposals are initialised with either learnable position encodings or random vectors and are iteratively updated to integrate traffic scene context [38, 97, 98]. The updated proposals are then treated as query vectors within the TF framework and used to predict trajectories via a unified prediction header. This method is advantageous because it allows for easy scaling of the number of queries, potentially reducing the risk of mode collapse.

An alternative to predefined anchors is the generation of adaptive anchors based on the dynamic traffic scene context, allowing for better accommodation of varying scenarios [99–101]. This approach offers greater flexibility and adaptability, making it more suitable for complex and evolving environments.

A summary of multi-modal prediction methods and relevant sources can be found in Table 5.

5.3 | Comparison of state-of-the-art methods

The training and validation of DL-based prediction algorithms are dataset-dependent. Different datasets may have distinct evaluation metrics, making it difficult to directly compare all

methods. This subsection will compare some state-of-the-art methods based on two frequently used datasets: NGSIM [103] and Argoverse [92] as presented in Tables 6 and 7. A more detailed summary of the characteristics of these methods is provided in Tables A1 and A2 of the Appendix. The data in the tables are all from the publications and model parameters are fine-tuned by models’ creators to achieve competitive performance.

Table 6 includes the RMSE of trajectory prediction across various prediction horizon values. The results are sorted by ascending the final prediction error (at 5 s). It can be seen that the larger the prediction horizon, the lower the prediction accuracy (lower RMSE values, better the prediction performance). Regarding the prediction backbones, LSTMs are popular. They are commonly used in encoding historical states of the TV, as well as those of SVs. The produced encodings of SVs are then permuted and used in interaction mining. Though the pipeline is similar across different interaction mining methods, the details of how SVs are identified and arranged vary. CS-LSTM [45] selects most adjacent SVs and arranges them in a grid centred on the TV. This TV-centric interaction mining narrows down the field of interest (FOI) to explore potential interaction and turns out to be more effective than identifying all vehicles in the current scene, as used in MATF [69]. Prediction accuracy is further improved when SVs are arranged in non-Euclidean ways, like a graph [54] or in polar coordinates [46, 47]. These topologies better identify SVs and are more adaptive to different road geometry like, for example, a curve. Advanced NNs like GNNs [54] and AMs [47, 94] can be used for interaction mining. In HLTP [94], a combination of NNs is used to mine the agent interaction. An AM-based method is used to explore the short-term interaction with neighbours in the forward vision while another CNN-based method is used to mine the interaction of all SVs. Then these two encodings are fused by TFs to generate a comprehensive encoding of the traffic context. Through Table 6, it can be seen that explicit interaction mining has proven to provide more accurate trajectory prediction results than implicit mining, which is used in S-LSTM [29] and D-LSTM [30], because NNs have limited ability to mine interaction directly from tedious input matrices consisting of original states of both the TV and SVs.

Despite the advancements in the organisation of SVs, the development of the prediction backbone is crucial to improve the prediction accuracy. In the applications of LSTM, the sequential structure of the encoder-decoder helps reduce prediction errors. Table 6 shows that, though implicit interaction mining is used, M-LSTM [33] has an encoder-decoder-based model and achieves better results than S-LSTM [29] and D-LSTM [30] which do not have an equivalent LSTM decoder. Further improvements are achieved by applying AM and TF in the prediction backbone. In IH-Net [68], the prediction backbone is still an LSTM-based encoder-decoder. Differently from the widely applied encoder architecture that uses the hidden state of the last LSTM cell as the encoder’s final output, hidden states at all time steps are cached in a feature matrix. In the decoding stage, additional high-dimension features are obtained by attending the hidden state of the decoder LSTM cell to

TABLE 5 A summary of multi-modal prediction methods.

Multi-modal solutions		Subclass	Works	Summary
Multi-modal	Proposal-based	Learnable proposals	[38, 97, 98]	<ul style="list-style-type: none"> These proposals can be initialised by learnable positional encodings [38] or random latent vectors [97]. To have a comprehensive understanding of traffic scenarios, these proposals will be updated with the traffic scene encodings that encompass agent historical state encoding, road scene encoding and corresponding interaction encoding. This updating process can be carried out either hierarchically or uniformly. MLPs are usually used to generate final trajectory predictions based on these updated proposals. Pros: easy to implement in TF structures and flexible to set modality numbers
		Predefined proposals	[33, 45, 47, 46, 94, 90]	<ul style="list-style-type: none"> Proposals are predefined according to the potential longitudinal and lateral manoeuvres of agents. The probabilities of all proposals are predicted according to the traffic scene context encodings and are then used to produce manoeuvre encodings, which will be concatenated with traffic scene context encodings to predict the trajectory for each proposal. Pros: good interpretability and reduced exploring scope Cons: limit the expressive ability in driving manoeuvre identification of NNs
	Multi-header-based	-	[48, 59, 78, 76, 82, 99, 102]	<ul style="list-style-type: none"> All modalities will have unique prediction headers, which share the same NN structure but have different parameters, to predict the trajectories based on the shared traffic scene encoding. MLPs are usually used as the prediction headers. Pros: simple to implement Cons: susceptible to modal collapse because of the shared traffic scene context across headers
	Anchor-based	Adaptive anchors	[99–101]	<ul style="list-style-type: none"> Compared to pre-defined anchors, adaptive anchors are generated with flexibility according to the varying traffic scenarios. They can be initialised by a learnable vector and be updated with traffic scene encodings [100, 101], or generated directly from the traffic scene encodings [99]. Pros: flexible to set modality numbers and adaptive to scenarios Cons: hard to train the model in the two-stage structure
		Predefined anchors	[85, 95]	<ul style="list-style-type: none"> A set of fixed anchor trajectories can be obtained in advance by clustering trajectory samples from the datasets. Future trajectory points are predicted as offsets to the anchor trajectory points. Pros: Avoid the modal collapse and gain Cons: Require a large amount of anchors to cover all potential modalities
	Goal-based	-	[61, 79–81, 96]	<ul style="list-style-type: none"> Goals are possible final positions of an agent and they are predicted before the trajectory prediction. With the predicted goals, complete trajectories could be generated by NNs like MLPs. Pros: good interpretability and adaptive to different scenarios
Uni-modal	Probability-based	Latent variables	[37, 39, 40, 69, 93]	<ul style="list-style-type: none"> These latent variables can be obtained from a pre-defined distribution or a learned distribution by NNs. A latent variable will be added to the fused traffic scene encoding to form a unique final encoding for each modality. Pros: easy to implement with GFs Cons: lack of interpretability and modal collapse
	Uni-header	-	[29, 30, 49, 51, 54, 68, 77]	<ul style="list-style-type: none"> Predict a single trajectory based on the traffic scene encoding and one prediction header. Not able to capture the multi-modal characteristics.

TABLE 6 Comparison of state-of-the-art model performance based on NGSIM.

Literature	Backbone	Interaction mining	Scene mining	Fusion	RMSE				
					1 s	2 s	3 s	4 s	5 s
IH-Net [68]	LSTM, AM	CNN	–	–	0.45	0.81	1.11	1.50	1.86
HLTP [94]	LSTM	AM, CNN, TF	–	–	0.48	0.99	1.54	2.22	3.09
GRIP [54]	LSTM	CNN, GNN	–	–	0.64	1.13	1.80	2.62	3.60
BAT [47]	LSTM	AM	–	–	0.23	0.81	1.54	2.52	3.62
Polar-V [46]	LSTM	Pooling	–	–	0.25	0.84	1.58	2.53	3.75
ST-LSTM [49]	LSTM	LSTM	–	–	0.56	1.19	1.93	2.78	3.76
CS-LSTM [45]	LSTM	CNN	–	–	0.61	1.27	2.09	3.10	4.37
M-LSTM [33]	LSTM	Implicit	–	–	0.58	1.26	2.12	3.24	4.66
MATF [69]	LSTM	CNN	CNN	Concat.	0.67	1.51	2.51	3.71	5.12
D-LSTM [30]	LSTM	Implicit	Implicit	–	0.49	1.41	2.60	4.06	5.79
S-LSTM [29]	LSTM, MLP	Implicit	Implicit	–	0.72	2.00	3.76	5.97	9.01

TABLE 7 Comparison of DL-based model performance on argoverse.

Literature	Backbone	Interaction mining	Scene mining	Fusion	K = 1			K = 6			Params (K)	GFLOPs
					min ADE	min FDE	MR	min ADE	min FDE	MR		
SEPT [97]	TF, MLP	AM	AM	AM	1.44	3.18	0.52	0.73	1.06	0.10	–	–
QCNet [100]	AM	AM	AM	AM	–	–	–	0.73	1.07	0.11	–	–
Wayformer [98]	TF	AM	AM	AM	–	–	–	0.77	1.16	0.12	–	2.13 [99]
LAformer [93]	GRU	AM	GRU	AM	–	–	–	0.77	1.16	–	2654 [93]	–
ProphNet [99]	TF	MLP	MLP	AM	1.52	3.33	–	0.77	1.14	0.11	–	0.40 [99]
HiVT [102]	MLP	AM	MLP	AM, MLP	–	–	–	0.77	1.17	0.13	2529 [102]	–
Multipath++ [101]	LSTM, MLP	LSTM, AM	MLP	AM	–	–	–	0.79	1.21	0.13	–	2.19 [99]
mfTransformer [104]	TF	TF	TF	TF	–	–	–	0.82	1.28	0.14	2469 [104]	0.41 [104]
HVTD [59]	AM	GCN	GCN	AM	1.59	3.62	0.58	0.82	1.27	0.15	892 [59]	–
mmTransformer [42]	TF	TF	GNN	TF	1.77	3.78	0.59	0.84	1.34	0.15	2607 [104]	0.18 [104]
LaneGCN [76]	MLP	AM	GCN	FC, GCN	1.71	3.78	0.59	0.87	1.36	0.16	3701 [104]	1.07 [104]
LaneRCNN [81]	MLP	GCN	GCN	GCN	1.69	3.69	0.57	0.90	1.45	0.12	–	–
HOME [61]	GRU, MLP	AM	CNN	–	1.69	3.65	0.57	0.94	1.45	0.10	5100 [104]	4.80 [104]
LaPred [78]	MLP, LSTM	FC	LSTM	FC	1.48	3.29	–	–	–	–	1830 [105]	3.61 [105]
HeteroGCN [96]	MLP, GRU	GCN	GCN	GCN	1.57	3.41	0.54	–	–	–	–	–
SSL-Lanes [82]	MLP	AM	GCN	AM	1.63	3.56	0.57	–	–	–	–	–
DSP [80]	MLP, CNN	GAT	GCN	GAT	1.69	3.71	0.58	–	–	–	–	–
VectorNet [77]	MLP	GCN	MLP	GCN	1.81	4.01	–	–	–	–	72K [77]	0.04 [77]

the feature matrix by cross attention. This feature encoding is then concatenated with the decoder LSTM hidden state to predict the future position. This combination of LSTM and AM improves the ability to extract historical features and makes IH-Net rank first in the NGSIM dataset considering its RMSE of 1.86 m at the final prediction time stamp.

In NGSIM, although road scene information like the corresponding driving lane of vehicles and deviations to the lane

markings are provided, road layout information is seldom used for a lack of a fine-grained road layout map.

Table 7 compares some state-of-the-art methods based on the Argoverse 1 Motion Forecast Dataset [92]. The Argoverse dataset provides varying traffic scenes in urban scenarios. An accurate trajectory prediction relies on a comprehensive understanding of the traffic scene and a good handling of manoeuvre uncertainties. The former challenge calls for

powerful methodologies on agent interaction and road scene mining while the latter is usually resolved by a multi-modal prediction.

In Argoverse, the observation horizon and prediction horizon are 2 s and 3 s, respectively. The selected metrics are minADE, minFDE, and MR (the distance threshold of MR is 2 m). Different methods are sorted in ascending order by minADE ($K = 6$), where a smaller value indicates a better result in the multi-modal prediction. If the minADE ($K = 6$) does not exist, the models are sorted by minADE ($K = 1$) instead. In addition, the numbers of trainable parameters and giga floating-point operations (GFLOPs) of some prediction models are listed, the latter of which reflects the overall computation required to complete the prediction. These values are collected from [77, 93, 99, 102, 104, 105] and are highly related to model sizes and their computation complexity.

Argoverse samples anchor points across the whole drivable area (DA) in a dense and even-distributed way. Benefiting from these prior points, it is convenient to relate the dynamic and static traffic scene elements to these points and create a uniform discrete representation in rasterised RGB images or multi-channel grid maps. CNNs are usually used as backbones to predict the multi-modal predictions in these cases.

Despite the efficiency of CNNs, the rasterisation of traffic scenes can be time-consuming, potentially hindering real-time applications. To address this issue, VectorNet [77] introduces a novel approach by representing agent trajectories and lane boundaries as sparse polyline elements. It leverages GNNs to extract high-dimensional features within each polyline, while AM is applied to capture interactions among all agent and lane boundary polylines. VectorNet achieves a minADE of 1.810 m and a minFDE of 4.010 m. Although its performance does not surpass CNN-based models like HOME, which has a minADE of 1.690 m, a minFDE of 3.650 m, and a parameter count of 5100K, VectorNet stands out with its lightweight model of only 72K parameters and its capability to explore both local and global road scenes. Consequently, it is still widely used in other methods as a preprocessing step for road scene information [42, 97].

LaneGCN [76] builds on VectorNet's concept by representing lane segments using their centrelines and constructing lane graphs based on the connection types of these centrelines. Unlike VectorNet, which fuses agent interactions and road scene information in a single step by creating a heterogeneous graph containing both lane and agent nodes, LaneGCN introduces a map-aware agent encoding process that is performed step by step. First, agents' encodings are aggregated into their surrounding lane nodes using AM. Next, GCN propagates information among the connected lane nodes. Finally, the agent encodings are updated by aggregating the lane node features back to the agents through AM. This fusion process enables agents to better perceive the traffic and geometric characteristics of lanes, leading to a more accurate understanding of traffic scenarios, as evidenced by a minADE of 1.710 m and a minFDE of 3.780 m. This hierarchical fusion approach is also employed by models like LaneRCNN [81], DSP [80], and HeteroGCN [96]. In these methods, for enhanced local exploration, lane nodes

surrounding the agent are retrieved to form a local graph and GNNs are used to propagate information among both agent and lane nodes. Subsequently, a global graph is constructed to facilitate interaction mining on a broader scale.

From Table 7, it can be seen that top-performing models surpass the others by incorporating AM and TF technologies, particularly excelling in multi-modal evaluations. Generally, these models achieve reliable multi-modal predictions by equipping TF decoders with learnable queries to generate predictions, drawing inspiration from the TF-based object detection method DETR [106]. These learnable queries are updated through the process of interaction mining and road scene mining, thus capturing the traffic scene context.

mmTransformer [42] is among the first models to apply TFs. It utilises three stacked TFs to extract historical motion features of agents, aggregate global static map information and mine agent-agent interactions. Within each TF, the learnable decoder queries are updated with the output of the TF encoder, gradually incorporating traffic scene features. This strategy turns out to be efficient and effective, outperforming GNN-based and CNN-based solutions with a minADE ($K = 6$) of 0.840 m and a minADE ($K = 1$) of 1.360 m. The performance of scene understanding is further enhanced by focusing on the neighbourhood of the TV and then integrating this local information with a global operator, as demonstrated in HiVT [102]. In this approach, a local region is identified around each agent, where agent-agent and agent-lane interactions are explored using AM at each time step. These local features are then integrated and processed through self-attention to learn spatial interactions among agents.

Advancing this concept, a more comprehensive approach involves combining all map encodings and agent encodings and applying self-attention to simultaneously capture interactions and road scene context. This method has proven more effective in traffic scene analysis, as seen in models like ProphetNet [99], Wayformer [98] and SEPT [97]. SEPT [97], which ranks first in Table 7, further enhances its ability to capture agent motion and road scene features through self-supervised learning. Specifically, its agent motion and road scene encoders are pre-trained before the trajectory prediction training stage to reconstruct partially masked trajectory and road borderline samples by self-supervised learning, in which smaller deviations from the reconstructed sample to the ground truth indicate a deeper understanding of the traffic scenes. Then, these pre-trained encoders are trained together with the decoder in the motion prediction training stage.

Considering the comprehensive reasoning abilities of AM and TF, there appears to be a trend to replace the GNN-based message propagation mechanism with the attention mechanism, though vehicles and road layout elements are still represented in a graph.

From Table 7, a notable feature associated with the application of AMs (including TFs) is the significant increase in the model size due to the additional parameters required in attention computation. Models employing LSTM or MLP backbones typically have fewer learnable parameters, often less than 2000K, as shown by LaPred [78] and VectorNet [77].

In contrast, TF-based methods like mmTransformer [42] and mfTransformer [104] generally have parameter sizes around 2500K.

CNN-based methods usually have moderate parameter sizes in a single CNN layer. However, thorough scene exploration often requires stacked CNN layers with progressively larger channel sizes, leading to models with considerable learnable parameters, as exemplified by HOME [61]. LaneGCN [76], despite using an MLP-based backbone and GCN for scene mining, exhibits a much larger model size of up to 3701K. This is primarily due to the multiple applications of AMs for capturing agent-to-lane, lane-to-agent, and agent-to-agent interactions.

While model size partially reflects computational complexity, the relationship between learnable parameters and computational demands (measured by GFLOP) is not strictly linear. For example, LaPred [78] has fewer parameters (1830K) but a higher GFLOP (3.619) compared to LaneGCN [76]. Similarly, mfTransformer [104] achieves a low GFLOP (0.408) despite its larger parameter size. This misalignment is because computational efficiency depends not only on model size but also on model inputs. The computational complexity of GNN-based models like LaneGCN [76] and VectorNet [77] depends on the number of graph nodes in the scene. And this vectorised map input is more efficient than the rasterised image input used in models like HOME [61].

Though AMs introduce more parameters and therefore increase computation complexity, their efficiency could be enhanced by uniform and compact structures [99]. For example, removing computationally expensive operations like multi-axis attention and autoregressive sequential prediction can reduce the inference costs, with a comparison of Wayformer [98] and ProphNet [99]. Operations like parallel interaction mining in mfTransformer [104] further improve the efficiency. Moreover, methods incorporating AMs can achieve low GFLOPs by rearranging target traffic scene elements and focusing on effective feature exploration during agent interaction mining and scene mining. For instance, HiVT [102] reduces the attention scope to a moderate range, which turns out to speed up inference.

5.4 | Discussions of future research directions

The following are some of the outstanding challenges and research gaps in vehicle trajectory prediction and could be considered as opportunities for future works:

1. **Time efficiency:** Most of the existing works in the literature mainly focus on the improvement of prediction accuracy. The complexity surges because a single NN is stacked multiple times for dedicated feature mining and different types of NNs are integrated for heterogeneous data processing, for example, image data and numerical data. Those can potentially propose challenges in real-time applications. Thus, in model design, the runtime performance characteristics of NNs should be taken into consideration, for example, if parallel computation is supported. Some metrics of computational burden should also be included during the model evaluation to assess the applicability of trajectory prediction.
2. **Storage requirement:** The implementation of the trajectory prediction models relies on both dynamic and static memories. The intermediate outputs between two sub-modules and dynamic sequential data, like historical trajectory data, consume dynamic memories. At the same time, road scene data, like road layout data, may occupy static memories. Thus, it is worth studying prediction models with light and integrated structures to enable fewer intermediate outputs. The application of sparsely represented road layout data should be explored because, despite some limitations in accurate scene representation, it enables efficient vehicle-map matching and minimal storage requirements.
3. **Model generalisation:** Currently, trajectory prediction models are trained in a small variety of scenarios, like several roundabouts, cross intersections and highways, limited by the availability of the datasets. These datasets contain distinct manoeuvres, for example, datasets collected in urban might pay more attention to turning instead of lane changing on the highways and even lane-changing manoeuvres are different among different scenarios. Also, some data imbalance exists in these datasets. For example, going-through trajectories might dominate the manoeuvres while few turning trajectories exist. These two issues result in poor performances when the models are exposed to scenarios with different geometries, topologies and traffic characteristics. So, the applicability of the proposed models to different scenarios should be carefully evaluated. Besides, weather conditions are hardly considered. They are also crucial elements for trajectory prediction accuracy. A model trained on sunny days might fail on rainy days because agents' behaviour will be different.
4. **Accurate prediction of heterogeneous traffic agents:** Most datasets only provide monotonous scenarios, with either vehicles or vulnerable road users dominating the traffic scenarios. Based on these datasets, a trained model might not perform well while predicting trajectories of heterogeneous traffic agents, since they might have significantly different motion patterns. A prediction model that enables an accurate prediction of all types of road actors will improve the prediction and the safety of AVs, since heterogeneous traffic is very common, especially in urban areas.
5. **Robustness and turbulence tolerance:** Complete observability of all traffic agents and smooth tracking are assumed in most of the available datasets. This is not the case in many real-world applications, because measurement noises and occlusions are important factors to consider from the perspective of onboard sensors. The performance of prediction models should therefore be assessed while considering these factors.
6. **Insufficient scenario information:** Assumptions are made that the EV has full observation of the TV with a historical horizon. However, this is not true, especially the newly perceived TV with a short observation horizon. These TVs

might pose a great threat to the EV and it is hard for current trajectory prediction models to estimate their motions due to a lack of enough historical information. Further, some prediction models deeply use complete road layout information. If the lanes are occupied temporally, for example, due to road obstructions such as accidents or construction, the trained prediction model might struggle to anticipate their motion accurately, as lane-changing manoeuvres in these situations can differ significantly from those in normal driving conditions.

7. **Model ensemble:** A prediction model can hardly be generalised among various scenarios. With pre-trained models, solutions can be explored to identify the characteristics of the traffic scenario and adaptively select the most suitable model to infer the predictions, not limited to meta-learning.
8. **Prediction result leverage:** The purpose of trajectory prediction is to help the EV better understand the dynamic traffic scenarios during risk assessment and trajectory planning tasks. It is still an open question to make use of the prediction results, especially when the prediction results are represented with Gaussian uncertainties or multi-modal predictions are provided. A probabilistic scenario representation with a long horizon, initialised from the prediction results, introduces massive input information and thus poses challenges to trajectory planning algorithms.

6 | CONCLUSION

The DL-based methods used in vehicle trajectory prediction were reviewed by organising them into three categories: trajectory prediction backbone, interaction mining, and road scene mining. In the prediction backbone, vehicle trajectory prediction highly depends on vehicle historical states, which are usually obtained by stacking historical data across the temporal dimension. For the representation of vehicle interactions and road scenes, relevant data are arranged in specific structures, like graphs and grids. These data are sometimes further converted into formats that are suited for the operations of NNs, typically to encode temporal or spatial features. For example, vector maps are often converted to rasterised images for the convenience of CNN processing. This analysis revealed why a specific NN architecture is chosen according to the component scope and the input data type and how similar NN architectures could be used in very different components of the prediction model, with different scopes and input structures. Furthermore, the performance of several prediction models was compared according to some commonly used evaluation metrics. It was found that appropriate encodings of vehicle interaction and road scenes improve the prediction accuracy because they provide a better understanding of the driving scenario. In addition, the results showed that AMs have the ability to mine both vehicle interaction and road scenes, achieving state-of-the-art prediction performance compared to other NNs. At last, some potential research directions were outlined in this article, mainly focusing on the practical application of vehicle trajectory prediction models.

AUTHOR CONTRIBUTIONS

Chenhui Yin: Conceptualization; data curation; formal analysis; investigation; methodology; resources; visualization; writing—original draft; writing—review & editing. **Marco Cecotti:** Conceptualization; formal analysis; investigation; methodology; project administration; supervision; writing—review & editing. **Daniel Auger:** Supervision; writing—review & editing. **Abbas Fotouhi:** Supervision; writing—review & editing. **Haobin Jiang:** Funding acquisition; supervision; writing—review & editing.

ACKNOWLEDGEMENTS

This work was supported in part by the China Scholarship Council under Grant No. 202108690001 and the Graduate Research and Innovation Projects of Jiangsu Province under Grant KYCX21_3334.

CONFLICT OF INTEREST STATEMENT

No potential conflicts of interest have been identified.

DATA AVAILABILITY STATEMENT

No new data were created or analysed during this study. Data sharing is not applicable to this article.

OPEN ACCESS POLICY

For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) license to any Accepted Manuscript version arising.

ORCID

Marco Cecotti  <https://orcid.org/0000-0002-4457-3631>

REFERENCES

1. Barth, A., Franke, U.: Where will the oncoming vehicle be the next second? In: IEEE Intelligent Vehicles Symposium (IV), pp. 1068–1073. IEEE, Piscataway (2008)
2. Ammoun, S., Nashashibi, F.: Real time trajectory prediction for collision risk estimation between vehicles. In: International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 417–422. IEEE, Piscataway (2009)
3. Huang, Y., Du, J., Yang, Z., Zhou, Z., Zhang, L., Chen, H.: A survey on trajectory-prediction methods for autonomous driving. IEEE Trans. Intell. Veh. 7(3), 652–674 (2022)
4. Zhang, Z., Wang, C., Zhao, W., Cao, M., Liu, J.: Ego vehicle trajectory prediction based on time-feature encoding and physics-intention decoding. IEEE Trans. Intell. Transp. Syst. 25(7), 6527–6542 (2024)
5. Vasquez, D., Fraichard, T.: Motion prediction for moving objects: A statistical approach. In: IEEE International Conference on Robotics and Automation, pp. 3931–3936. IEEE, Piscataway (2004)
6. Lefèvre, S., Vasquez, D., Laugier, C.: A survey on motion prediction and risk assessment for intelligent vehicles. Robomech J. 1(1), 1 (2014)
7. Hu, W., Xiao, X., Fu, Z., Xie, D., Tan, T., Maybank, S.: A system for learning statistical motion patterns. IEEE Trans. Pattern Anal. Mach. Intell. 28(9), 1450–1464 (2006)
8. Chen, G., Gao, Z., Hua, M., Shuai, B., Gao, Z.: Lane change trajectory prediction considering driving style uncertainty for autonomous vehicles. Mech. Syst. Signal Process. 206, 110854 (2024)
9. Mozaffari, S., Al-Jarrah, O.Y., Dianati, M., Jennings, P., Mouzakitis, A.: Deep learning-based vehicle behavior prediction for autonomous driving applications: A review. IEEE Trans. Intell. Transp. Syst. 23(1), 33–47 (2022)

10. Gomes, I., Wolf, D.: A review on intention-aware and interaction-aware trajectory prediction for autonomous vehicles (2022)
11. Huang, R., Zhuo, G., Xiong, L., Lu, S., Tian, W.: A review of deep learning-based vehicle motion prediction for autonomous driving. *Sustainability* 15(20), 14716 (2023)
12. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* 65(6), 386–408 (1958)
13. Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: LSTM: A search space odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* 28(10), 2222–2232 (2017)
14. Cho, K., van Merriënboer, B., Gulcehre, C., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* (2014)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Red Hook, NY (2012)
16. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. *arXiv* (2014)
17. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. *arXiv* (2016)
18. Vaswani, A., Shazeer, N., Parmar, N., et al.: Attention is all you need. In: *Advances in Neural Information Processing Systems*. Curran Associates, Red Hook, NY (2017)
19. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., et al.: Generative adversarial networks. *arXiv* (2014)
20. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. In: *Advances in Neural Information Processing Systems*. Curran Associates, Red Hook, NY (2015)
21. Hegde, C., Dash, S., Agarwal, P.: Vehicle trajectory prediction using GAN. In: *International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp. 502–507. IEEE, Piscataway (2020)
22. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. *arXiv* (2017)
23. Liu, Q., Lathrop, B., Butakov, V.: Vehicle lateral position prediction: A small step towards a comprehensive risk assessment system. In: *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 667–672. IEEE, Piscataway (2014)
24. Tomar, R., Verma, S.: Safety of lane change maneuver through a priori prediction of trajectory using neural networks. *Netw. Protoc. Algorithms* 4(1), 4–21 (2012)
25. Luo, W., Yang, B., Urtasun, R.: Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 3569–3577. IEEE, Piscataway (2018)
26. Casas, S., Luo, W., Urtasun, R.: IntentNet: Learning to predict intention from raw sensor data. In: *Conference on Robot Learning*, pp. 947–956 (2018)
27. Casas, S., Gulino, C., Liao, R., Urtasun, R.: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. *arXiv* (2019)
28. Yang, B., Luo, W., Urtasun, R.: PIXOR: Real-time 3D object detection from point clouds. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7652–7660. IEEE, Piscataway (2018)
29. Althché, F., de La Fortelle, A.: An LSTM network for highway trajectory prediction. In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 353–359. IEEE, Piscataway (2017)
30. Xin, L., Wang, P., Chan, C.-Y., Chen, J., Li, S.E., Cheng, B.: Intention-aware long horizon trajectory prediction of surrounding vehicles using dual LSTM networks. *arXiv* (2019)
31. Zyner, A., Worrall, S., Nebot, E.: Naturalistic driver intention and path prediction using recurrent neural networks. *IEEE Trans. Intell. Transp. Syst.* 21(4), 1584–1594 (2020)
32. Park, S.H., Kim, B., Kang, C.M., Chung, C.C., Choi, J.W.: Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 1672–1678. IEEE, Piscataway (2018)
33. Deo, N., Trivedi, M.M.: Multi-modal trajectory prediction of surrounding vehicles with maneuver based LSTMs. In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 1179–1184. IEEE, Piscataway (2018)
34. Giuliari, F., Hasan, I., Cristani, M., Galasso, F.: Transformer networks for trajectory forecasting. In: *International Conference on Pattern Recognition (ICPR)*, pp. 10335–10342. IEEE, Piscataway (2021)
35. Quintanar, A., Fernández-Llorca, D., Parra, I., Izquierdo, R., Sotelo, M.A.: Predicting vehicles trajectories in urban scenarios with transformer networks and augmented information. In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 1051–1056. IEEE, Piscataway (2021)
36. Gao, K., Li, X., Chen, B., et al.: Dual transformer based prediction for lane change intentions and trajectories in mixed traffic environment. *IEEE Trans. Intell. Transp. Syst.* 24(6), 6203–6216 (2023)
37. Lee, N., Choi, W., Vernaza, P., Choy, C.B., Torr, P.H.S., Chandraker, M.: DESIRE: Distant future prediction in dynamic scenes with interacting agents. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2165–2174. IEEE, Piscataway (2017)
38. Cho, K., Ha, T., Lee, G., Oh, S.: Deep predictive autonomous driving using multi-agent joint trajectory prediction and traffic rules. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2076–2081. IEEE, Piscataway (2019)
39. Dulian, A., Murray, J.C.: Multi-modal anticipation of stochastic trajectories in a dynamic environment with conditional variational autoencoders. *arXiv* (2021)
40. NN, S., Liu, B., Pittaluga, F., Chandraker, M.: SMART: Simultaneous multi-agent recurrent trajectory prediction. *arXiv* (2020)
41. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI*, pp. 234–241. Springer, Cham (2015)
42. Liu, Y., Zhang, J., Fang, L., Jiang, Q., Zhou, B.: Multimodal motion prediction with stacked transformers. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7573–7582. IEEE, Piscataway (2021)
43. Wang, Y., Zhao, S., Zhang, R., Cheng, X., Yang, L.: Multi-vehicle collaborative learning for trajectory prediction with spatio-temporal tensor fusion. *IEEE Trans. Intell. Transp. Syst.* 23(1), 236–248 (2022)
44. Yu, J., Zhou, M., Wang, X., Pu, G., Cheng, C., Chen, B.: A dynamic and static context-aware attention network for trajectory prediction. *ISPRS Int. J. Geo-Inf.* 10(5), 336 (2021)
45. Deo, N., Trivedi, M.M.: Convolutional social pooling for vehicle trajectory prediction. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1549–15498. IEEE, Piscataway (2018)
46. Peng, Y., Zhou, Z., Li, L.: Trajectories prediction of vehicles at the intersection based on LSTM neural network. In: *COTA International Conference of Transportation Professionals*, pp. 2386–2397. IEEE, Piscataway (2019)
47. Liao, H., Li, Z., Shen, H., et al.: BAT: Behavior-aware human-like trajectory prediction for autonomous driving. *Proc. AAAI Conf. Artif. Intell.* 38(9), 10332–10340 (2024)
48. Fei, C., He, X., Ji, X.: Multi-modal vehicle trajectory prediction based on mutual information. *IET Intell. Transp. Syst.* 14(3), 148–153 (2020)
49. Dai, S., Li, L., Li, Z.: Modeling vehicle interactions via modified LSTM models for trajectory prediction. *IEEE Access* 7, 38287–38296 (2019)
50. Ding, W., Chen, J., Shen, S.: Predicting vehicle behaviors over an extended horizon using behavior interaction network. In: *International Conference on Robotics and Automation (ICRA)*, pp. 8634–8640. IEEE, Piscataway (2019)
51. Zhang, K., Zhao, L., Dong, C., Wu, L., Zheng, L.: AI-TP: Attention-based interaction-aware trajectory prediction for autonomous driving. *IEEE Trans. Intell. Veh.* 8(1), 73–83 (2023)
52. Gao, Y., Fu, J., Feng, W., Xu, T., Yang, K.: Surrounding vehicle trajectory prediction under mixed traffic flow based on graph attention network. *Phys. Stat. Mech. Its Appl.* 639, 129643 (2024)
53. Sheng, Z., Huang, Z., Chen, S.: Kinematics-aware multigraph attention network with residual learning for heterogeneous trajectory prediction. *J. Intell. Connect. Veh.* 7(2), 138–150 (2024)

54. Li, X., Ying, X., Chuah, M.C.: GRIP: Graph-based interaction-aware trajectory prediction. In: IEEE Intelligent Transportation Systems Conference (ITSC), pp. 3960–3966. IEEE, Piscataway (2019)
55. Li, J., Shi, H., Guo, Y., Han, G., Yu, R., Wang, X.: TraGCAN: Trajectory prediction of heterogeneous traffic agents in IoV systems. *IEEE Internet Things J.* 10(8), 7100–7113 (2023)
56. Ren, Y.: EMSIN: Enhanced Multi-Stream Interaction Network for Vehicle Trajectory Prediction. *IEEE Trans. Fuzzy Syst.* 33(1), 54–68 (2025)
57. Ma, Y., Zhu, X., Zhang, S., Yang, R., Wang, W., Manocha, D.: TrafficPredict: Trajectory prediction for heterogeneous traffic-agents. *arXiv* (2019)
58. Li, Z., Lu, C., Yi, Y., Gong, J.: A hierarchical framework for interactive behaviour prediction of heterogeneous traffic participants based on graph neural network. *IEEE Trans. Intell. Transp. Syst.* 23(7), 9102–9114 (2022)
59. Tang, Y.: Hierarchical vector transformer vehicle trajectories prediction with diffusion convolutional neural networks. *Neurocomputing* 580, 127526 (2024)
60. Sadid, H., Antoniou, C.: Dynamic spatio-temporal graph neural network for surrounding-aware trajectory prediction of autonomous vehicles. *IEEE Trans. Intell. Veh.* 1–14 (2024)
61. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: HOME: Heatmap output for future motion estimation. *arXiv* (2021)
62. Yang, Z., Wan, Y., Du, L., Zhang, W., Yang, X., Han, Y.: Vehicle trajectory prediction based on attention optimized with real-scene sampling. *Syst. Sci. Control Eng.* 12(1), 2347889 (2024)
63. Song, H., Luan, D., Ding, W., Wang, M.Y., Chen, Q.: Learning to predict vehicle trajectories with model-based planning. *arXiv* (2021)
64. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: GOHOME: Graph-oriented heatmap output for future motion estimation. *arXiv* (2021)
65. Messaoud, K., Yahiaoui, I., Verroust-Blondet, A., Nashashibi, F.: Attention based vehicle trajectory prediction. *IEEE Trans. Intell. Veh.* 6(1), 175–185 (2021)
66. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social LSTM: Human trajectory prediction in crowded spaces. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 961–971. IEEE, Piscataway (2016)
67. Cai, J., Jiang, H., Wang, J., Li, A.: Multi-head attention-based intelligent vehicle lane change decision and trajectory prediction model in highways. *J. Intell. Transp. Syst.* 1–18 (2024)
68. Li, C., Liu, Z., Lin, S., Wang, Y., Zhao, X.: Intention-convolution and hybrid-attention network for vehicle trajectory prediction. *Expert Syst. Appl.* 236, 121412 (2024)
69. Zhao, T., Xu, Y., Monfort, M., et al.: Multi-agent tensor fusion for contextual trajectory prediction. *arXiv* (2019)
70. Min, H., Xiong, X., Wang, P., Zhang, Z.: A hierarchical LSTM-based vehicle trajectory prediction method considering interaction information. *Automot. Innov.* 7(1), 71–81 (2024)
71. Messaoud, K., Deo, N., Trivedi, M.M., Nashashibi, F.: Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation. In: IEEE Intelligent Vehicles Symposium (IV), pp. 165–170. IEEE, Piscataway (2021)
72. Djuric, N., Radosavljevic, V., Cui, H., et al.: Short-term motion prediction of traffic actors for autonomous driving using deep convolutional networks. *arXiv* (2018)
73. Deo, N., Trivedi, M.M.: Trajectory forecasts in unknown environments conditioned on grid-based plans. *arXiv* (2021)
74. Cui, H., Radosavljevic, V., Chou, F.-C., et al.: Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In: International Conference on Robotics and Automation (ICRA), pp. 2090–2096. IEEE, Piscataway (2019)
75. Hou, L., Li, S.E., Yang, B., Wang, Z., Nakano, K.: Integrated graphical representation of highway scenarios to improve trajectory prediction of surrounding vehicles. *IEEE Trans. Intell. Veh.* 8(2), 1638–1651 (2023)
76. Liang, M., Yang, B., Hu, R., et al.: Learning lane graph representations for motion forecasting. In: Computer Vision – ECCV, pp. 541–556. Springer, Cham (2020)
77. Gao, J., Sun, C., Zhao, H., et al.: VectorNet: Encoding HD maps and agent dynamics from vectorized representation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11522–11530. IEEE, Piscataway (2020)
78. Kim, B., Park, S.H., Lee, S., et al.: LaPred: Lane-aware prediction of multi-modal future trajectories of dynamic agents. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 14631–14640. IEEE, Piscataway (2021)
79. Deo, N., Wolff, E., Beijbom, O.: Multimodal trajectory prediction conditioned on lane-graph traversals. In: Conference on Robot Learning, pp. 203–212. PMLR, New York (2022)
80. Zhang, L., Li, P., Chen, J., Shen, S.: Trajectory Prediction with Graph-based Dual-scale Context Fusion. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 11374–11381. IEEE, Piscataway (2022)
81. Zeng, W., Liang, M., Liao, R., Urtasun, R.: LaneRCNN: Distributed representations for graph-centric motion forecasting. *arXiv* (2021)
82. Bhattacharyya, P., Huang, C., Czarnecki, K.: SSL-Lanes: Self-supervised learning for motion forecasting in autonomous driving. *arXiv* (2022)
83. Lan, Z., Liu, L., Fan, B., Lv, Y., Ren, Y., Cui, Z.: Traj-LLM: A new exploration for empowering trajectory prediction with pre-trained large language models. *IEEE Trans. Intell. Veh.* 1–14 (2024)
84. Liu, J., Lin, H., Wang, X., Wu, L., Garg, S., Hassan, M.M.: Reliable trajectory prediction in scene fusion based on spatio-temporal Structure Causal Model. *Inf. Fus.* 107, 102309 (2024)
85. Phan-Minh, T., Grigore, E.C., Boulton, F.A., Beijbom, O., Wolff, E.M.: CoverNet: Multimodal behavior prediction using trajectory sets. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 14062–14071. IEEE, Piscataway (2020)
86. Ngiam, J., Caine, B., Vasudevan, V., et al.: Scene transformer: A unified architecture for predicting multiple agent trajectories. *arXiv* (2022)
87. Guo, H., Meng, Q., Zhao, X., Liu, J., Cao, D., Chen, H.: Map-enhanced generative adversarial trajectory prediction method for automated vehicles. *Inf. Sci.* 622, 1033–1049 (2023)
88. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: THOMAS: Trajectory heatmap output with learned multi-agent sampling. *arXiv* (2022)
89. Luo, C., Sun, L., Dabiri, D., Yuille, A.: Probabilistic multi-modal trajectory prediction with lane attention for autonomous vehicles. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2370–2376. IEEE, Piscataway (2020)
90. Mo, X., Xing, Y., Liu, H., Lv, C.: Map-adaptive multimodal trajectory prediction using hierarchical graph neural networks. *IEEE Robot. Autom. Lett.* 8(6), 3685–3692 (2023)
91. Yang, M., Zhang, B., Wang, T., et al.: Vehicle interactive dynamic graph neural network based trajectory prediction for internet of vehicles. *IEEE Internet of Things J.* 11(22), 35777–35790 (2024)
92. Chang, M.-F., Lambert, J., Sangkloy, P., et al.: Argoverse: 3D tracking and forecasting with rich maps. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8740–8749. IEEE, Piscataway (2019)
93. Liu, M., Cheng, H., Chen, L., et al.: LAformer: Trajectory prediction for autonomous driving with lane-aware scene constraints. *arXiv* (2023)
94. Liao, H., Li, Y., Li, Z., et al.: A cognitive-based trajectory prediction approach for autonomous driving. *IEEE Trans. Intell. Veh.* 9(4), 4632–4643 (2024)
95. Chai, Y., Sapp, B., Bansal, M., Anguelov, D.: MultiPath: multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv* (2019)
96. Gao, X., Jia, X., Li, Y., Xiong, H.: Dynamic scenario representation learning for motion forecasting with heterogeneous graph convolutional recurrent networks. *IEEE Robot. Autom. Lett.* 8(5), 2946–2953 (2023)
97. Lan, Z., Jiang, Y., Mu, Y., Chen, C., Li, S.E.: SEPT: Towards efficient scene representation learning for motion prediction. *arXiv* (2023)

98. Nayakanti, N., Al-Rfou, R., Zhou, A., Goel, K., Refaat, K.S., Sapp, B.: Wayformer: Motion forecasting via simple & efficient attention networks. arXiv (2022)
99. Wang, X., Su, T., Da, F., Yang, X.: ProphNet: Efficient agent-centric motion forecasting with anchor-informed proposals. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 21995–22003. IEEE, Piscataway (2023)
100. Zhou, Z., Wang, J., Li, Y., Huang, Y.: Query-centric trajectory prediction. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 17863–17873. IEEE, Piscataway (2023)
101. Varadarajan, B., Hefny, A., Srivastava, A., et al.: MultiPath++: Efficient information fusion and trajectory aggregation for behavior prediction. In: International Conference on Robotics and Automation (ICRA), pp. 7814–7821. IEEE, Piscataway (2022)
102. Zhou, Z., Ye, L., Wang, J., Wu, K., Lu, K.: HiVT: Hierarchical vector transformer for multi-agent motion prediction. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8813–8823. IEEE, Piscataway (2022)
103. Traffic Analysis Tools: Next generation simulation—FHWA operations. <https://ops.fhwa.dot.gov/trafficanalysisistools/ngsim.htm>. Accessed August 2024
104. He, B., Li, Y.: Multi-future Transformer: Learning diverse interaction modes for behaviour prediction in autonomous driving. IET Intell. Transp. Syst. 16(9), 1249–1267 (2022)
105. L. Li, X. Wang, D. Yang, Y. Ju, et al.: Real-time heterogeneous road-agents trajectory prediction using hierarchical convolutional networks and multi-task learning. IEEE Trans. Intell. Veh. 9(2), 4055–4069 (2024)
106. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Computer Vision – ECCV, pp. 213–229. Springer, Cham (2020)

How to cite this article: Yin, C., Cecotti, M., Auger, D.J., Fotouhi, A., Jiang, H.: Deep-learning-based vehicle trajectory prediction: A review. IET Intell. Transp. Syst. 19, e70001 (2025). <https://doi.org/10.1049/itr2.70001>

APPENDIX

To give a summary of all approaches listed in Section 5.3, their pros and cons are listed in the following tables (Tables A1 and A2).

TABLE A1 Summary of characteristics of methods in Table 6.

Works	Pros	Cons
CS-LSTM [45]	<ul style="list-style-type: none"> Retain the spatial relationship among SVs while extracting interaction by using CNN 	<ul style="list-style-type: none"> The topology of SVs does not cater to curved roads
MATF [69]	<ul style="list-style-type: none"> Provide a comprehensive traffic scene encoding by concatenating agent encodings and road scene encodings in a multi-channel grid map Improve the prediction accuracy by GAN 	<ul style="list-style-type: none"> Redundant interaction mining exists because all vehicles on the scene are identified as SVs
BAT [47]	<ul style="list-style-type: none"> Improve the generalisation of the model by identifying SVs in polar coordinates Use a more informative representation of agents' behaviour in the form of a high-dimension vector, instead of discrete behaviour categories 	<ul style="list-style-type: none"> Lose some interaction features because of applying max pooling to aggregate the encodings of SVs
Polar-V [46]	<ul style="list-style-type: none"> Improve the generalisation of the model by identifying SVs in polar coordinates 	<ul style="list-style-type: none"> Lose some interaction features because of applying max pooling to aggregate the encodings of SVs
M-LSTM [33]	<ul style="list-style-type: none"> Facilitate a manoeuvre-aware prediction by using a separate LSTM module to identify the driving manoeuvres 	<ul style="list-style-type: none"> Limit the performance of NNs in manoeuvre identification because the number of driving manoeuvre patterns might be beyond predefined discrete types
HLTP [94]	<ul style="list-style-type: none"> Mimic the human perceptual patterns in identifying SVs and mining their interactions with the TV Mock the selective and dynamic focusing of human vision in interaction mining by a shift-window attention 	<ul style="list-style-type: none"> Have not taken into consideration the road scene information and adjusted identification strategy of SVs according to the road geometries
GRIP [54]	<ul style="list-style-type: none"> Create a spatiotemporal graph representation by connecting the agents to their historical states and their spatially adjacent SVs Avoid the high computational burden and storage burden by dividing temporal and spatial connections into a separate adjacency matrix 	<ul style="list-style-type: none"> Over-smoothing might happen in multi-step GCN operation while mining agent interaction CNN might be insufficient to extract temporal features, like motion patterns, of agents
S-LSTM [29]	<ul style="list-style-type: none"> Provide a simple structure in the application of LSTM-based solutions in trajectory prediction 	<ul style="list-style-type: none"> Have insufficient interaction mining by encoding both interaction and road scenes implicitly Have a too-long observation stage, like 10 s
ST-LSTM [49]	<ul style="list-style-type: none"> Fasten the learning process by calculating the importance coefficient of SVs according to prior knowledge Improve the prediction accuracy by a two-stage process: generating preliminary trajectory predictions and then correcting them by adding additional spatial interaction information 	<ul style="list-style-type: none"> Lower the efficiency by arranging SVs pair-wisely with the TV and calculating pair-wise interaction Do not consider the interaction among SVs
D-LSTM [30]	<ul style="list-style-type: none"> Create a unified framework of trajectory prediction for lane keeping and lane change manoeuvres Reduce the effect of data imbalance by introducing recognition of target lanes in the prediction model 	<ul style="list-style-type: none"> Need post-process of the predictions since the longitudinal and lateral states of agents are predicted separately Rely on accurate target lane prediction
IH-Net [68]	<ul style="list-style-type: none"> Avoid temporal feature loss in the decoder by attending to all of LSTM's historical hidden states with AM Improve the prediction accuracy of lane-changing behaviours by applying temporal attention and spatial attention to all of LSTM's historical hidden states 	<ul style="list-style-type: none"> Explore the interaction between agents only based on their states at individual time steps, without historical information

TABLE A2 Summary of characteristics of methods in Table 7.

Works	Pros	Cons
SEPT [97]	<ul style="list-style-type: none"> • Provide simple and powerful traffic scene mining by extracting spatiotemporal relationships of agent-agent and agent-map in a parallel way • Improve the spatiotemporal feature extraction ability of traffic scene encoders by leveraging self-supervised learning 	<ul style="list-style-type: none"> • Lead to feature loss in the temporal dimension because of collapsing the temporal dimension by max pooling
mmTransformer [42]	<ul style="list-style-type: none"> • Refine the initial trajectory proposals hierarchically by fusing agent-to-agent and agent-to-map mining results step-by-step • Provide a well-structured traffic scene fusion by exploring agent-to-agent per time step, temporal dependencies per agent and agent-to-lane interactions hierarchically 	<ul style="list-style-type: none"> • Fuse agent-to-agent interactions into the proposals twice which might cause redundancy and increase computation complexity
Wayformer [98]	<ul style="list-style-type: none"> • Enable adaptive positional embeddings in AM-based traffic scene encoders to improve scene-mining abilities • Provide simple and powerful traffic scene mining by creating a heterogeneous mix of static road scene and dynamic agent inputs and extracting spatiotemporal relationships of agent-agent and agent-map in a parallel way 	<ul style="list-style-type: none"> • Fail to perform sufficient exploration along the lane but the neighbour lane pieces
LAformer [93]	<ul style="list-style-type: none"> • Narrow down the scope of traffic scene mining by dynamic selection of the most potential lane segments • Provide a well-structured traffic scene fusion by exploring lane-to-agent, agent-to-lane and agent-to-agent interactions hierarchically • Improve the prediction accuracy by a two-stage process: generating prior predictions based on lane-aware proposals and then refining trajectories 	<ul style="list-style-type: none"> • Rely heavily on the right identification of currently occupied lanes • Generate unexpected proposal patterns because of introducing a random latent vector
mfTransformer [104]	<ul style="list-style-type: none"> • Provide simple and powerful traffic scene mining by extracting spatiotemporal relationships of agent-agent and agent-map in a parallel way • Provide efficient road scene mining by restricting the attention between points within a lane 	<ul style="list-style-type: none"> • Extract unrepresentative lane features by self-attention of all points inside a lane • Extract insufficient trajectory features in the temporal dimension because of applying an average pooling across the temporal dimension
HiVT [102]	<ul style="list-style-type: none"> • Provide a fine-grained and efficient exploration of local traffic scenes by operating agent-centric context extraction and compensate for the restricted local receptive fields by applying another global interaction among local regions • Perform efficient traffic scene mining by narrowing down the attention space at local regions of each time step and using factorized attention across spatial and temporal dimensions separately • Provide a well-structured traffic scene fusion by exploring agent-to-agent per time step, temporal dependencies per agent and agent-to-lane interactions hierarchically 	<ul style="list-style-type: none"> • Produce insufficient agent-to-agent and agent-to-lane interaction mining because of focusing on relative states of agents just at one individual time step instead of the complete historical horizon
LaPred [78]	<ul style="list-style-type: none"> • Generate a lane-level traffic scene understanding by relating agents' features to those of instance-level lane entities • Produce a comprehensive understanding between agents and their surrounding lanes by aggregating all relevant lane information into the target agent 	<ul style="list-style-type: none"> • Encode the neighbour information implicitly, in which way the interaction between neighbours might not be sufficiently extracted • Lack of flexibility in lane representation since the candidate lane should have the same length • Only identify the agents on the candidate lane as the neighbours which is not adequate in agent interaction mining
LaneGCN [76]	<ul style="list-style-type: none"> • Enable customized exploration depths by setting different aggregation steps along different lane directions • Enable efficient long-range exploration by dilated connection between graph nodes • Provide a well-structured traffic scene fusion by exploring agent-to-lane, lane-to-lane, lane-to-agent and agent-to-agent interactions hierarchically 	<ul style="list-style-type: none"> • Represent the location of the lane piece by its centroid which might be insufficient, especially in the case of a long lane • Challenge the storage if the map is fine-grained with a high resolution
HVTD [59]	<ul style="list-style-type: none"> • Generate a comprehensive encoding of local regions by heterogeneous attention among both agent and lane encodings • Adaptively learn the importance of different neighbours by arranging them in a graph creating a learnable adjacency matrix • Enable the acquisition of both local and global information by decomposing the traffic scene mining into local patches and global interactions 	<ul style="list-style-type: none"> • Fail to perform sufficient exploration along the lane but the neighbour lane pieces • Challenge on-chip storage because of maintaining a dynamic graph across the temporal dimension

(Continues)

TABLE A2 (Continued)

Works	Pros	Cons
ProphNet [99]	<ul style="list-style-type: none"> • Provide simple and powerful traffic scene mining by extracting spatiotemporal relationships of agent-agent and agent-map in a parallel way • Generate adaptive trajectory anchors from dynamic traffic scenes and avoid the multimodality collapse by adding additional goal-based embeddings generated from historical encodings of the target agent and initial learnable proposals 	<ul style="list-style-type: none"> • Fail to enhance the agent-to-lane interactions because of using only the agent encoding at the current step instead of its complete historical encoding
QCNet [100]	<ul style="list-style-type: none"> • Realise efficient and fine extraction of agents' temporal features by conducting factorized attention at each past time step • Generate adaptive trajectory anchors from dynamic traffic scenes and initial proposals to avoid modal collapse, and improve their diversity by self-attention • Improve the prediction accuracy by a two-stage process: generating trajectory anchors and then leveraging these anchors to refine the predictions 	<ul style="list-style-type: none"> • Fail to better aggregate lane information by identifying relevant lane elements by a range search instead of a topology-based search along the current lane element
Multipath++ [101]	<ul style="list-style-type: none"> • Generate informative anchors in the form of high-dimension embeddings and update them with fused traffic scene encodings • Create a novel and efficient gated fusion module to replace the computationally intensive cross-attention 	<ul style="list-style-type: none"> • Use max pooling as one of the key technologies in the proposed gated fusion module which inevitably causes information loss
HOME [61]	<ul style="list-style-type: none"> • Generate uniform and informative traffic scene representation by rasterised images • Realise efficient traffic scene feature extraction by the combination of rasterised images and CNNs 	<ul style="list-style-type: none"> • Cause information redundancy while matching the size of the agent's encoding and that of the road scene encoding • Raise computation burden in the process of traffic scene rasterization • Cause information loss of the road scene because of overlapping of road scene elements in the rasterised image
LaneRCNN [81]	<ul style="list-style-type: none"> • Better explore the local regions by extracting surrounding lane segments which an agent might go to and creating agent-centric subgraphs • Enable efficient message passing globally by projecting lane segments of local graphs into a global lane graph 	<ul style="list-style-type: none"> • Represent the location of the lane piece by its centroid which might be insufficient, especially in the case of a long lane • Challenge on-chip storage by maintaining a dynamic graph for each agent • Cause computational burden while calculating relative relations between a lane segment and its surroundings at each time step
HeteroGCN [96]	<ul style="list-style-type: none"> • Provide a heterogeneous representation of the traffic scenarios by grouping both agents and lane elements in a dynamic graph • Narrow down the interaction mining scope by searching relevant lane nodes according to map topologies 	<ul style="list-style-type: none"> • Have redundant information in the identified lane segment since they are selected by depth-first-search and some of them are not reachable in the prediction horizon
SSL-Lanes [82]	<ul style="list-style-type: none"> • Self-learning is used to train the traffic scene encoder with a better understanding of both local and global map structure 	<ul style="list-style-type: none"> • Enhance different traffic scene encoders with self-learning once at a time, which did not provide a comprehensive enhancement of the overall prediction performance
DSP [80]	<ul style="list-style-type: none"> • Create a hierarchical graph representation that emphasizes both local traversability and global topologies, enabling the agent to explore space of adjacent lanes that is geometrically close but topologically distant • Bridge nodes between the local graph and the global graph according to nodes' geometrical adjacency to enable message passing 	<ul style="list-style-type: none"> • Challenge the on-chip storage because both local and global graphs are constructed with a high resolution
VectorNet [77]	<ul style="list-style-type: none"> • Provide a tight representation of the traffic scene by treating the agent trajectories and map elements as sequences of vectors • Enable traffic scene mining both locally and globally by constructing two hierarchical graphs • Enhance the global interaction mining performance by self-learning 	<ul style="list-style-type: none"> • Partly lose spatial features of vector sequences because of max pooling operations on polyline-level elements • Provide only an unimodal solution



DSPACE

<https://dspace.org/>

Deeplearningbased vehicle trajectory prediction: a review

Yin, Chenhui; Cecotti, Marco; Auger, Daniel J.; Fotouhi, Abbas; Jiang, Haobin

2025-01

Institution of Engineering and Technology (IET)

<https://doi.org/10.1049/itr2.70001>

Yin C, Cecotti M, Auger DJ, et al., (2025) Deeplearningbased vehicle trajectory prediction: a review. IET Intelligent Transport Systems, Volume 19, January/December 2025, Article number e70001