Risk-Aware Self-Consistent Imitation Learning for Trajectory Planning in Autonomous Driving

Yixuan Fan, Yali Li, and Shengjin Wang*

[1] Department of Electronic Engineering, Tsinghua University
[2] Beijing National Research Center for Information Science and Technology
fan-yx21@mails.tsinghua.edu.cn, {liyali13, wgsgj}@tsinghua.edu.cn

Abstract. Planning for the ego vehicle is the ultimate goal of autonomous driving. Although deep learning-based methods have been widely applied to predict future trajectories of other agents in traffic scenes, directly using them to plan for the ego vehicle is often unsatisfactory. This is due to misaligned objectives during training and deployment: a planner that only aims to imitate human driver trajectories is insufficient to accomplish driving tasks well. We argue that existing training processes may not endow models with an understanding of how the physical world evolves. To address this gap, we propose **RaSc**, which stands for **R**isk-**a**ware **S**elf-**c**onsistent imitation learning. RaSc not only imitates driving trajectories, but also learns the motivations behind human driver behaviors (to be risk-aware) and the consequences of its own actions (by being self-consistent). These two properties stem from our novel prediction branch and training objectives regarding Time-To-Collision (TTC). Moreover, we enable the model to better mine hard samples during training by checking its self-consistency. Our experiments on the large-scale real-world nuPlan dataset demonstrate that RaSc outperforms previous state-of-the-art learning-based methods, in both open-loop and, more importantly, closed-loop settings.

1 Introduction

Using data-driven methods to achieve driving planning is a rapidly developing field in autonomous driving. Unlike trajectory prediction, planning pursues generating feasible, safe, efficient, and comfortable motion for the ego vehicle, not just output trajectories that are close to human driver records in terms of distance-based metrics. In the context of planning, closed-loop testing serves to evaluate whether the former goal is achieved, whereas open-loop testing serves to evaluate the latter. Notably and maybe counterintuitively, closed-loop scores and open-loop scores are often weakly correlated or even negatively correlated [11, 14, 59, 66]. One reason that leads to this phenomenon is that open-loop testing metrics fail to reflect real driving proficiency: when model outputs deviate only slightly from ground truth, collisions may actually occur; while when deviations are large, the model may have provided another viable path.

* Corresponding Author.

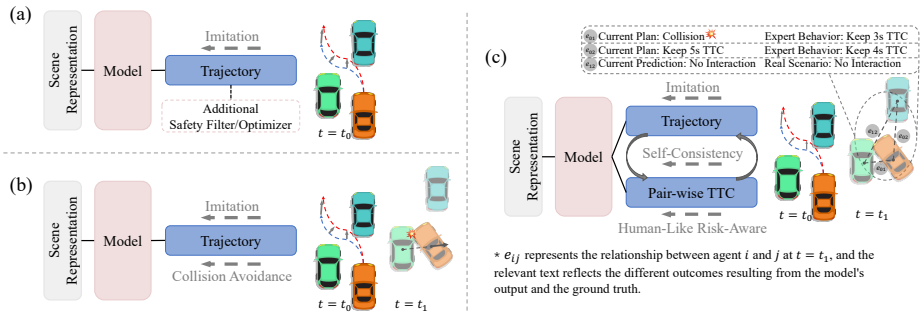

Fig. 1: Different imitation learning paradigms for safe planning. **(a)** [8, 27, 29, 51, 61] The trainable model is only supervised by trajectory imitation, while safety is ensured by an additional non-parameterized safety filter/optimizer. **(b)** [2, 32, 65, 67] Handcrafted collision avoidance constraints are added as training objectives. **(c)** Our method learns human awareness of risks from data and enables the model to understand the consequences of its own actions through self-consistency constraints. This is achieved through training objectives related to pair-wise Time-To-Collision.

Analogous to the little correlation between open-loop and closed-loop performance in evaluation, existing training processes of imitation learning models suffer from misaligned training objectives and actual deployment needs since supervision signals almost solely come from human driver trajectories, as shown in Fig. 1 (a-b). To improve closed-loop performance, current methods adopted remedial approaches that explicitly avoid collisions, by incorporating a non-parameterized safety filter/optimizer or penalizing unsafe behaviors in the network output. However, we argue that these approaches are insufficient for guiding the network to learn the motivations behind human driver behaviors.

In this paper, we adopt Time-To-Collision (TTC) [23] as the medium to capture human comprehension of dynamic information in traffic scenes and as a cue for planning. TTC is defined as the time for two objects to collide if they continue at their present speed and heading. Compared to distance, TTC incorporates both position and velocity information, making it more flexible, as safe distances obviously differ at different speeds. A larger TTC is not always better; while an autonomous driving system maintaining a sufficiently large TTC ensures safety, it can be overly cautious and affect efficiency. Reviewing the literature on transportation, TTC has long been a vital consideration in evaluating the safety of traffic environments [56], and the use of TTC as a cue for decision-making in traffic scenarios is well-established [26]. For human drivers, information about TTC is one of the most critical factors affecting their visual control of braking [35]. To the best of our knowledge, We are the first to integrate the concept of TTC into pure learning-based trajectory planning, and to demonstrate its effectiveness in aiding neural networks to comprehend decision-making mechanisms.

Instead of manually designing safety constraints for different scenarios, we adopt pair-wise future TTC prediction to learn human drivers' risk awareness

and management skills from data. Such data includes not only the ego vehicle's experiences, but all its historical observations, as similar intelligence manifests in all traffic participants. Furthermore, the TTCs computed from the model's output trajectories should be consistent with the TTCs predicted by the model, establishing self-consistency. This enables the model to understand the consequences of its own behaviors. The vision achieved by these two designs is illustrated in Fig. 1 (c), where the model understands the interactive relationships between agents and better comprehends the ego vehicle's capability of seeking overtaking opportunities in the depicted road scene.

It is noteworthy that supervision from imitating human TTCs and the self-consistent constraint can produce divergent gradients. To address this, we designed the weighting for the self-consistency constraint at different training stages. Moreover, by examining the network's self-consistency, we obtain superior online hard example mining (OHEM). Relevant designs enable the model to find the right learning direction in most cases.

In summary, with the proposed RaSc model, our main contribution is threefold: (1) We introduce risk-aware imitation, which leverages TTC data from human driving records to guide our model in learning human drivers' motivations and risk awareness skills; (2) We propose self-consistent planning, by constraint our model to perform consistent trajectory planning and TTC estimation, thereby teaching it to interpret its own actions and comprehend the consequences of its behavior; (3) We present a superior OHEM strategy, which can find more accurate hard samples by employing a self-consistency check.

We conduct experiments on the nuPlan dataset, and attain state-of-the-art performance. We believe our method serves as a pioneering work in addressing the fundamental mismatch between open-loop training and closed-loop deployment of imitation learning-based planning models.

2 Related Work

Ego Forecasting. Deep learning methods have shown adeptness at understanding complex driving scenarios and mapping them to potential trajectories of human drivers, where they significantly outperform non-parametric methods represented by the Intelligent Driver Model (IDM) [53]. Ego forecasting models can process raw sensor signals from cameras, lidars, etc. [5,8,9,43,47,60,62], or utilize perception results as inputs, projecting scenes in a BEV space [2,6,13,25,34,36] or analyzing vectorized representations [18,33,37,40,49,54]. Driven by datasets like nuScenes [3], Argoverse [7,58], Waymo Open Motion Dataset [16], etc., and open-loop metrics, related methods have made remarkable progress in aspects including scene representation [15,19,20,31,63,64], model architecture with attention mechanisms [39,42,55,68], among others. Moreover, the future trajectories of different agents influence each other. Therefore, joint prediction across multiple targets has been shown beneficial for both performance and speed [17,20,30,41]. These studies form the foundation for the method designed in this paper.

From Ego Forecasting to Safe Planning. If our prediction of trajectories were perfect, we would only need to control the vehicle to move along the predicted trajectory to realize driving like human drivers. However, this ideal situation has not materialized with the advancement of prediction algorithms. This stems from the inherent limitations of behavior cloning [11, 12, 14, 48, 57, 59, 66], in that they lack lucid explanations of the decision-making process.

To achieve better closed-loop performance, previous studies incorporated manually designed prior knowledge through additional modules. The designs of these modules differ across works, but all make important contributions to the results of corresponding studies. In works adopting the paradigm in Fig. 1 (a), [8] designed a two-layer safety mechanism, first checking for collisions based on trajectory prediction results, applying a hard brake if a collision is detected; then using another neural network specialized for predicting brake values as a safety redundancy. [61] performs safety checks on planned trajectories in order of confidence. [27, 29] preset cost functions related to safety, comfort, etc., and optimize trajectories through the Gauss-Newton method and CasADi [1] ipopt solver respectively. [51] solves its proposed optimization problem using linear programming. In works adopting the paradigm in Fig. 1 (b), [2,67] generate loss when collisions or driving off-road occur under the rasterized representation, and the loss is proportional to the overlapping area with corresponding objects. [32] applies three manually designed safety constraints under the vectorized representation. [28] utilizes both paradigms (a) and (b). In addition, sampling-based planning methods such as [5,44,65] also incorporate the summarized safety mechanisms. Compared to these previous studies, our method aims to systematically enhance the neural network's intrinsic understanding of driving motivations.

3 Methodology

3.1 Problem Formulation and Approach Overview

We denote the ego vehicle as A_0 and the context agents as $A_{1:N}$. Each agent including the ego vehicle has a semantic class (i.e., vehicle, bicycle, or pedestrian), and its state at time t is denoted as s_i^t, where i is the agent index. We also introduce a vectorized high-definition map (including map elements denoted as $M_{1:M}$) and traffic light signals. For the ego vehicle, we have route roadblocks identifying its target road. Assuming the current time point is $t = 0$, given the states of all agents in the previous H time steps and the current time step $s_{0:N}^{-H:0}$, the model needs to make motion decision for the ego vehicle in the F future time steps $s_0^{1:F}$. In practice, we provide L possible plans in parallel along with corresponding confidence scores.

We also predict K possible future trajectories for all context agents across the F future time steps $s_{1:N}^{k,1:F}$, $k = 1, \cdots, K$. This is because predicting the future trajectories of other agents in the scene holds great importance for the ego vehicle's planning and is also an important basis for the methodology devised in this paper. Sec. 3.2 describes how we represent the scene and achieve joint

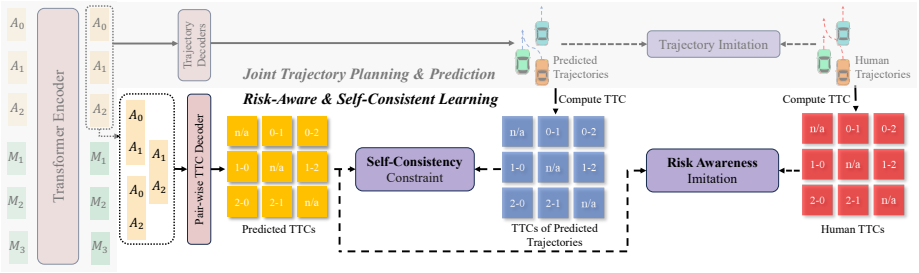

Fig. 2: Overview of our proposed RaSc framework. It expands our joint trajectory planning and prediction baseline (the gray part) by adding a pair-wise future TTC decoder (depicted in Fig. 3). The risk awareness imitation loss is computed by comparing the TTCs predicted by the model with the TTCs derived from human drivers' trajectories, and the self-consistency constraint is computed by comparing the predicted TTCs with the TTCs derived from the model's planned and predicted trajectories.

trajectory planning and prediction. Notably, the method proposed in this paper is generic, and this part may be flexibly adapted.

The module we propose in this paper extends the conventional joint planning and prediction pipeline. In Sec. 3.3, we introduce our pair-wise TTC decoder, whose function is to predict the TTCs between any two considered agents at P future time points. Building on this, Sec. 3.4 presents how we enable the model to learn human drivers' risk control abilities from data, while achieving self-consistent outputs to explain its own behaviors. We refer the reader to Fig. 2 for a visualized understanding of the model framework. To improve training efficiency and performance, Sec. 3.5 describes how we realize precise online hard example mining by leveraging the introduced concept of self-consistency.

3.2 Joint Trajectory Planning and Prediction

We employ a vectorized representation to encode information from traffic participants' historical states and the high-definition map. Each scene element is encoded in its local coordinate system. For agents, the coordinates' origin and main axis orientation are their position and heading at the last observable timestamp. For each lane center line, the coordinate origin is the average coordinate of all points on the segment, and the main axis orientation aligns with the direction from the first point to the last along the driving direction. For each crosswalk or stop line, the origin is the average coordinate of vertices of each element's polygon, and the main axis orientation is randomly selected.

For the ego vehicle, we encode its state at the last observable timestamp using an MLP, following [10, 21, 27, 61]. For each of the other agents, we use an LSTM [24] to encode the historical states. The state of any agent at each timestamp comprises position, velocity, acceleration, and length-width dimensions. As for map elements, we employ PointNets [45] for encoding.

To embed global position information, the ego vehicle's pose at the last observable timestamp is referenced for the origin and orientation of the global coordinate system. Then the transformation between each element's local coordinate and the global coordinate is utilized to compute a position embedding for each scene element:

$$p_i = \text{MLP}([\Delta x_i, \Delta y_i, \sin \Delta\theta_i, \cos \Delta\theta_i]) \tag{1}$$

Where Δx_i, Δy_i, and $\Delta\theta_i$ represent the displacement in x, y coordinates, and orientation between the local coordinate system of scene element i and the global coordinate system, respectively.

Attributes including scene element types, traffic light states for each lane, and binary variables indicating whether each lane belongs to the route for the ego vehicle, are represented by learnable embeddings, respectively. The agent and map embeddings are summed with their corresponding attribute embeddings and position embeddings, and then concatenated into a combined sequence. We employ Transformer [55] Encoder layers to achieve information passing. Finally, MLPs are applied to tokens of the ego vehicle and other agents to obtain the planned and predicted trajectories. The output size of the planning head and prediction head are $L \times F \times 3$ and $K \times F \times 3$, respectively. 3 for the lateral position, longitudinal position, and orientation. There are also output heads for estimating confidence scores. Predicting multiple possible future trajectories for context agents accounts for the multi-modality of intelligent agents' behaviors. Predicting multiple ego vehicle trajectories retains backup plans during deployment. By default, we use the trajectory with the highest confidence.

3.3 TTC Prediction

Estimating future TTCs between traffic participants with neural networks is one of the central pieces of our method. Future TTCs refer to TTCs at a future time point, it is calculated by agents' position, velocity, and heading at that time point. The hyperparameter introduced here is how far into the future we want to predict TTCs. We can also estimate TTCs at multiple future time points simultaneously. The contextualized tokens output by the Transformer Encoder are utilized by the trajectory decoder to plan or predict trajectories in the local coordinate systems of agents. However, computing TTCs requires the relative positional relationship between two agents. Therefore, we re-add the global position embeddings to these output tokens to form the input for the pair-wise TTC decoder.

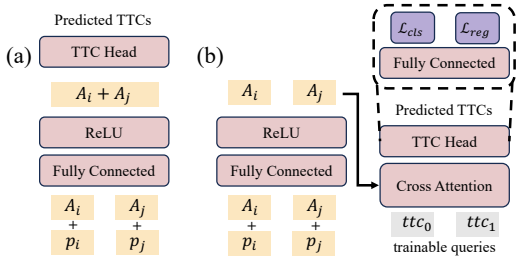

Fig. 3: Diagram of the two pair-wise TTC decoder designs. $\mathcal{L}_{cls}$ and $\mathcal{L}_{reg}$ are introduced in Sec. 3.4.

For the input tokens constructed for the TTC decoder, we first apply a transformation to the feature space using a fully connected linear layer, followed by a ReLU activation. The basic requirement the TTC decoder needs to satisfy is permutation invariance. Accordingly, we design two optional structures, as shown in Fig. 3. In structure (a), we simply add up the input tokens representing the two agents; in structure (b), we use a learnable embedding as the query for each future time point we estimate TTCs, and inquire about the two agents through a multi-head cross attention layer. Finally, a fully connected layer is utilized as the head to output prediction results. Regarding structure (a), the output head dimension is $P * 2$, where P is the number of future time points we estimate TTCs; for structure (b), the output head dimension is 2. The specifics of what the outputs stand for will be described in Sec. 3.4. We use structure (b) by default because it has better performance.

3.4 Training Objective

Trajectory Imitation. For trajectory planning and prediction, we adopt the Winner-Takes-All principle for loss computing, which applies loss to the trajectory deemed best for each agent. Here, best refers to the trajectories with the minimum final displacement error compared to the trajectories recorded in the dataset. We use the Smooth L1 loss for trajectory regression and the Cross Entropy loss for confidence estimation, the trajectory imitation loss for each agent is a weighted sum of them. Recognizing the importance of trajectories at earlier timestamps, we apply an exponentially decaying weight to the loss at each timestamp. Considering that context agents closer to the ego vehicle typically have a greater impact on its planning and also more accurate position and size annotations (since they are closer to the ego vehicle's sensing system), we weight the loss for each context agent based on its distance from the ego vehicle, using an exponentially decaying law. We denote the trajectory imitation loss for the ego vehicle and the weighted sum of the losses for all context agents are $\mathcal{L}_{plan}$ and $\mathcal{L}_{pred}$, respectively, the total trajectory imitation loss for each scenario will be $\mathcal{L}_{traj} = \mathcal{L}_{plan} + \mathcal{L}_{pred}$.

Risk Awareness Imitation. Our model learns human risk control abilities in traffic scenarios by encouraging the future TTCs predicted by our model to match the corresponding TTCs of human traffic participants. For the $N + 1$ agents in a scenario, their pairwise combinations at each future time point where TTC needs to be estimated result in $N(N+1)/2$, while the majority of them have infinite TTCs (means collision will never happen at their status). To enhance efficiency, we pre-label the training set by identifying all agent pairs with a TTC of less than 30 seconds at all considered future time points. During training, we focus on these pairs and a randomly selected subset of negative samples, thereby constructing the set of agent pairs to be considered, termed $\mathbb{Q}$, for each scenario, keeping the number of agent pairs to be computed at an $\mathcal{O}(N)$ level.

Our pair-wise TTC decoder outputs two values for each pair of agents at each future time point where TTC is estimated, representing whether the TTC to be predicted is below a predefined threshold and the numerical value of the

predicted TTC, respectively. The training objective for risk awareness imitation for each training scenario is defined as:

$$\mathcal{L}_{ra} = \mathcal{L}_{ttc}\left(\{p_{t,i,j}\}, \{ttc_{t,i,j}\}; \left\{ttc^*_{t,i,j}\right\}\right) = \frac{1}{|\mathbb{Q}|}\sum_{(t,i,j)\in\mathbb{Q}} (w_i + w_j)\cdot \\ \left(\lambda\mathcal{L}_{cls}\left(p_{t,i,j}, \sigma\left(ttc_{th} - ttc^*_{t,i,j}\right)\right) + p^*_{t,i,j}\mathcal{L}_{reg}\left(ttc_{t,i,j}, ttc^*_{t,i,j}\right)\right) \tag{2}$$

Here, t, i, j represent the indices for a specific future time point and an agent pair within the set $\mathbb{Q}$, which denotes an element referring to agents i and j at future time t. The term $p_{t,i,j}$ is the predicted probability that this agent pair will have a TTC less than a threshold value ttc_{th} at the future time point t, ttc_{th} is set to 10 seconds. $ttc_{t,i,j}$ is the TTC predicted by the decoder. $ttc^*_{t,i,j}$ is the TTC we calculate from trajectories in the dataset. The ground truth label $p^*_{t,i,j}$ is set to 1 if $ttc^*_{t,i,j} < ttc_{th}$, and to 0 otherwise. $\mathcal{L}_{cls}$ and $\mathcal{L}_{reg}$ are the focal loss [38] and the Smooth L1 loss, used to compute classification and regression loss respectively. λ is the weight of the classification loss. σ is the Sigmoid function, we use soft labels to enable the model to leverage more information beyond binary classifications. The term $p^*_{t,i,j}\mathcal{L}_{reg}$ means the regression loss is activated only for interacting agent pairs ($p^*_{t,i,j} = 1$) and is disabled otherwise ($p^*_{t,i,j} = 0$). w_i, w_j are weights corresponding to the two agents. As aforementioned, we assign higher weights to agents closer to the ego vehicle.

Self-Consistency Constraint. To enable the model to account for its own behaviors, we apply a self-consistency constraint by encouraging the two outputs of the model to agree with each other. Continuing with the previously mentioned set of agent pairs $\mathbb{Q}$ for each scenario, we now replace the supervisory signal for TTC estimation from the TTCs calculated from trajectories in the dataset for risk awareness imitation ($\left\{ttc^*_{t,i,j}\right\}$), with $\left\{ttc^{traj}_{t,i,j}\right\}$, the TTCs calculated from trajectories planned and predicted by the model. For the multiple output trajectories, we use the one with the highest confidence. This allows us to compute the self-consistency constraint following Eq. (2):

$$\mathcal{L}_{sc} = \mathcal{L}_{ttc}\left(\{p_{t,i,j}\}, \{ttc_{t,i,j}\}; \left\{ttc^{traj}_{t,i,j}\right\}\right) \tag{3}$$

Overall Training Objective. The total loss for each scenario during training is a weighted sum of the three aforementioned training objectives:

$$\mathcal{L} = \lambda_{traj}\mathcal{L}_{traj} + \lambda_{ra}\mathcal{L}_{ra} + \lambda_{sc}\mathcal{L}_{sc} \tag{4}$$

Here, $\lambda_{traj}, \lambda_{ra}, \lambda_{sc}$ represent the weights assigned to the trajectory imitation loss, risk awareness imitation loss, and self-consistency constraint, respectively. Notably, Since the model has poor performance in predicting trajectories during the early stages of training, λ_{sc} is designed to gradually increase according to a sine law as training progresses: $\lambda_{sc} = \lambda^0_{sc}\sin(\frac{step}{total\_steps}\cdot\frac{\pi}{2})$, where λ^0_{sc} is the maximum value of λ_{sc}, $step$ is the current training step and $total\_steps$ is the total training steps. In the later stages of training, λ_{sc} significantly surpasses λ_{ra} to enable the OHEM described below to work and obtain more closed-loop performance gains.

3.5 OHEM Induced by Self-Consistency

Traditional Online Hard Example Mining (OHEM) [52] accelerates convergence by sorting sample losses to identify hard samples. For current decision models based on imitation learning, the loss value indicates the discrepancy between the model's output trajectories and those in the dataset. Given the multi-modal nature of human traffic participants' behaviors, a model with certain performance might generate high-confidence trajectories that significantly differ from the ground truth but are still plausible for some samples. Training on these samples could negatively impact the model, and directly applying OHEM might exacerbate this issue. On the other hand, a model may exhibit poor closed-loop performance on some samples with low loss values, indicating that despite the output trajectories being spatially close to the ground truth, minor discrepancies could lead to severe outcomes such as collisions. Current methods might mistakenly classify these samples as easy due to their low loss values. The above phenomenon is illustrated in Fig. 4.

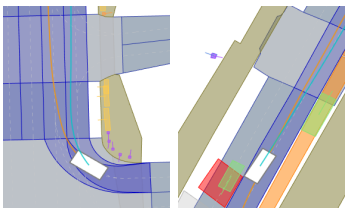

Fig. 4: Two common situations for a learning-based planner. In the left image, the highest confidence trajectory planned for the ego vehicle (the blue line) is perfectly reasonable but distant from the human record (the orange line). In the right image, the planned trajectory is close to the human record, but a small gap causes the ego vehicle to sideswipe a parked car.

We address this challenge by identifying hard samples through an examination of the model's self-consistency. For each training sample, our first step is to check the prediction loss of the context agents who have future TTCs with the ego vehicle within 10 seconds in the human records. If the prediction loss of any of these agents is higher than the average of the training batch, this sample is categorized as a hard sample and is kept for training. Otherwise, we proceed to the next step. In the second step, we categorize the samples into the following four types, with corresponding treatments:

- Both $\mathcal{L}_{plan}$ and $\mathcal{L}_{sc}$ are low (lower than the average of the training batch). These samples are considered easy for the model. To prevent the model from forgetting these samples and to maintain a large enough batch size, we include the loss from these samples in the backward propagation with a certain probability.
- High $\mathcal{L}_{plan}$ but low $\mathcal{L}_{sc}$: These samples may be the aforementioned wrong hard samples. To prevent model collapse, we include the loss from these samples in the backward propagation with a certain probability.
- Low $\mathcal{L}_{plan}$ but high $\mathcal{L}_{sc}$: Some of these samples may be challenging for the TTC decoder: despite the model producing trajectories similar to those of human drivers, it fails to adequately explain its own actions. Some other of these may be the aforementioned wrong easy samples: despite the planned

trajectories close to those of human drivers, the results are quite different. We retain these samples for training.
– Both $\mathcal{L}_{plan}$ and $\mathcal{L}_{sc}$ are high: These samples are hard and are kept.

The treatments outlined above, especially the treatment of the second type of samples, are only meaningful when the model has achieved sufficient performance levels. Therefore, we only apply OHEM in the later stages of training. Moreover, before deciding to discard the loss from any sample, we perform an additional check on the TTC calculated from the model's highest confidence planned and predicted trajectories, between the ego vehicle and other agents. If the ego vehicle has any TTC less than one second, we do not discard the loss from this sample.

4 Experiments

Benchmarks. We use the nuPlan [4] dataset to evaluate our method. It provides 1,200 hours of diverse real-world driving data and a closed-loop simulator. Evaluations are conducted in three different settings: (1) In open-loop evaluation, the model is scored by comparing its predicted trajectory against human driver records, using open-loop metrics including average/final displacement error, average/final heading error, and miss rate. (2) In closed-loop evaluation with non-reactive agents, the planned trajectory controls the ego vehicle in the simulator using an LQR controller, with replayed non-reactive agents, and performance is measured by closed-loop metrics including those related to safety, efficiency, comfort, etc. (3) In closed-loop evaluation with reactive agents, other vehicles react to the ego vehicle using an IDM [53] policy and closed-loop metrics are used to measure performance. More details about experiment settings are illustrated in [4]. We benchmarked our model on three validation sets, namely Val14 from [14], Test14-random and Test14-hard from [10]. Test14-hard consists of specially selected hard scenarios.

4.1 Comparison with SOTA

Tab. 1 compares the performance of RaSc with others. Our pure learning-based model demonstrates strengths especially in open-loop and non-reactive closed-loop evaluations when compared to other learning-based approaches. In reactive closed-loop evaluations, our method shows a slight underperformance, which can be attributed to its reliance on predicting the behavior of context agents. The IDM-driven agents exhibit significantly different behavior patterns from those observed in the dataset.

To further enhance the model's reliability, we can also incorporate a post-processing module into our method, combining paradigms (a) and (c) in Fig. 1. This module conducts two checks on the planned trajectories: (1) the TTCs between the ego vehicle and other agents, calculated from the model's planned trajectories and the highest confidence predicted trajectories. Planned trajectories resulting in any TTC less than one second are flagged as dangerous; (2) the

Table 1: Comparison with state-of-the-arts. RaSc* refers to our model with post-processing, as described in Sec. 4.1. OL means open-loop evaluation, NR-CL means nonreactive closed-loop evaluation and R-CL means reactive closed-loop evaluation. All metrics range from 0 to 100, with higher scores indicating better performance. The runtime includes feature extraction and model inference.

Planners		Val14			Test14-random			Test14-hard			Time (ms)
Type	Method	OL	NR-CL	R-CL	OL	NR-CL	R-CL	OL	NR-CL	R-CL	
Human	Log-Replay	100	94	80	100	94	76	100	86	69	-
Rule-based	IDM [53]	38	76	77	34	70	72	20	56	62	30
Learning-based	PlanCNN [46]	64	73	72	63	70	68	52	49	52	71
	UrbanDriver [50]	82	53	50	82	63	61	77	52	49	106
	GC-PGP [22]	83	59	55	77	56	51	74	43	40	113
	PDM-Open [14]	86	50	54	84	53	57	79	34	36	**28**
	PlanTF [10]	89	85	**77**	87	**86**	**81**	83	73	62	140
	RaSc (Ours)	**90**	**86**	75	**91**	**86**	76	**86**	**75**	**63**	75
w/ post-processing	GameFormer [29]	-	-	-	79	81	79	75	67	69	395
	PDM-Closed [14]	42	**93**	**92**	46	90	**92**	26	65	**75**	127
	PDM-Hybrid [14]	84	**93**	**92**	82	90	**92**	74	65	**75**	139
	RaSc* (Ours)	**87**	91	86	**88**	**91**	82	**82**	**77**	65	**92**

presence of off-road situations. Should these occur, the trajectories are flagged as dangerous. In instances where high-confidence trajectories are deemed dangerous, we consider other candidate trajectories. If all candidate trajectories are evaluated as dangerous, we resort to initiating a hard brake. Comparison of our model with post-processing between others is in the latter part of Tab. 1.

To improve the speed of training and inference, we implemented a two-dimensional TTC calculation method capable of efficient batch processing on GPUs. We also use a lightweight structural design. These ensure that our approach has a significant efficiency advantage over other methods with comparable performance.

4.2 Ablation Study

Effect of Each Design. We evaluate the contribution of each design in our model. As our approach is specifically aimed at enhancing the model's closed-loop performance, we report our results in the non-reactive closed-loop evaluation. Additionally, we report on two critical safety-related metrics: no ego at fault collisions, indicating whether the ego vehicle has not been at fault in any possible collisions, and time to collision within bound (0.95s), reflecting whether the ego vehicle maintains a sufficient TTC from other agents. The results, as illustrated in Tab. 2, demonstrate that all our design components contribute to performance gains in our model.

Comparison of TTC Decoder Designs. In Sec. 3.3 we introduce two structures for predicting future TTCs that meet our design criteria. Their perfor-

Table 2: Ablation for designs of our method. RA means applying the risk awareness imitation, SC means applying the self-consistency constraint, OHEM means applying the OHEM we propose. Coll means the metric "no ego at fault collisions", TTC means the metric "time to collision within bound" (not same as in the rest of the paper).

RA	SC	OHEM	Val14			Test14-random			Test14-hard		
			NR-CL	Coll	TTC	NR-CL	Coll	TTC	NR-CL	Coll	TTC
			79	87	82	78	88	81	66	82	77
✓			83	93	88	82	94	89	72	85	80
✓	✓		85	95	89	85	**95**	91	73	88	81
✓	✓	✓	**86**	**97**	**92**	**86**	**95**	**92**	**75**	**90**	**85**

Table 3: Comparison between different pair-wise TTC decoder designs on the Val14 set. P means precision, R means recall, RMSE means root mean square error.

TTC decoder	Driving Score			Risk Awareness			Self-Consistency		
	OL	NR-CL	R-CL	P↑	R↑	RMSE↓	P↑	R↑	RMSE↓
(a) Add up	89	82	72	0.61	0.55	0.79	0.69	0.75	0.53
(b) Cross attention	**90**	**86**	**75**	**0.65**	**0.71**	**0.52**	**0.72**	**0.89**	**0.14**

Table 4: Comparison between different time point choices for TTC estimation on the Val14 set. P means precision, R means recall, RMSE means root mean square error.

Time Point	Driving Score			Risk Awareness			Self-Consistency		
	OL	NR-CL	R-CL	P↑	R↑	RMSE↓	P↑	R↑	RMSE↓
0.5s	90	84	75	0.70	0.86	0.40	0.75	0.93	0.10
1.0s	90	86	75	0.65	0.71	0.52	0.72	0.89	0.14
1.5s	90	85	73	0.62	0.68	0.61	0.68	0.85	0.29
2.0s	89	82	69	0.59	0.64	0.65	0.66	0.79	0.42
0.5s & 1.0s	90	85	74	0.66	0.78	0.44	0.73	0.92	0.12
1.0s & 1.5s	90	86	73	0.63	0.70	0.55	0.70	0.86	0.24

mance is compared in Tab. 3, focusing on their influence on the model's final driving capabilities and their abilities in predicting pair-wise TTCs. The latter is evaluated through the precision and recall for the classification task and the root mean square error for the regression task, for predicting the ego vehicle's future TTCs with all context agents. The cross-attention-based model, with its higher capacity, can capture the interrelations between two agents more flexibly, thereby significantly outperforming the alternative.

Table 5: Comparison between different OHEM strategies. Coll means the metric "no ego at fault collisions", TTC means the metric "time to collision within bound" (not same as in the rest of the paper).

OHEM strategy	Val14			Test14-random			Test14-hard		
	NR-CL	Coll	TTC	NR-CL	Coll	TTC	NR-CL	Coll	TTC
w/o OHEM	85	95	89	85	**95**	91	73	88	81
Trajectory loss based	83	91	87	84	92	89	70	85	80
Self-consistency based	**86**	**97**	**92**	**86**	**95**	**92**	**75**	**90**	**85**

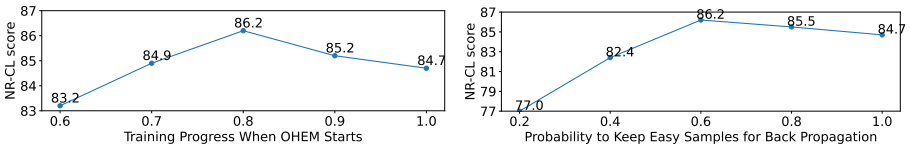

Fig. 5: Comparison between different OHEM configurations on the Val14 set. Starting OHEM at 1.0 training progress or keeping easy samples at 1.0 probability means not applying OHEM.

Choice of the Time Point for TTC Estimation. The choice of how far into the future to predict TTCs is an important hyperparameter for our method. In Tab. 4, we present the impact of this choice on both the model's final driving capability and its performance in predicting pair-wise TTCs for the ego vehicle. Predicting TTCs at more distant future time points is inherently more challenging, yet an appropriately challenging TTC estimation can better facilitate improvements in driving capability, particularly in closed-loop evaluations. We find that predicting TTCs at 1.0 seconds into the future offers the most significant benefit to driving capability.

Comparison between OHEM strategies. In Tab. 5, we compare our proposed self-consistency-induced OHEM against the traditional OHEM based on trajectory imitation loss, with a particular focus on their impact on closed-loop performance. The experimental results indicate that selecting hard samples based on trajectory imitation loss leads to performance degradation, and our method demonstrates a clear advantage.

Tuning OHEM Configuration. As discussed in Sec. 3.5, the effectiveness of our proposed OHEM strategy is contingent on the model already having attained a certain level of performance; thus, it is only feasible to apply our OHEM in the later stages of training. It is also necessary to retain the two types of easy samples with a certain probability. Fig. 5 illustrates the impact of these two hyperparameters on the model's performance in the non-reactive closed-loop evaluation. For optimal performance, we begin OHEM after 80% of the training process and retain easy samples with a 0.6 probability.

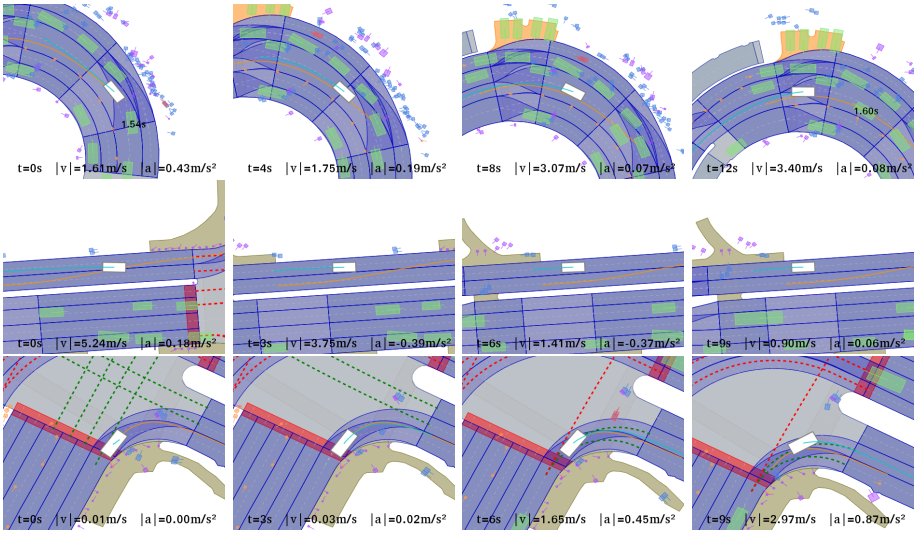

Fig. 6: Visualized results. Each row presents four sequential snapshots of one scenario, progressing from left to right. The current time step, ego vehicle's speed, and acceleration are documented at the bottom of each image. The bounding box in white indicates the ego vehicle, the orange line depicts a 15-second human driving trajectory from the dataset, and the blue line represents the trajectory planned by our model for the next 8 seconds. In the first scenario, the first and fourth images each highlight a context agent with the model's prediction of its TTC with the ego vehicle in one second. From top to bottom, the scenarios illustrate: (1) the ego vehicle accelerates and safely navigates through a complex pickup/dropoff area; (2) the ego vehicle notices pedestrians crossing the road and brakes to wait; (3) the ego vehicle waits for pedestrians to cross and then accelerates to initiate a right turn.

4.3 Visualized Results

In Fig. 6, we showcase some visualized experimental results of our method in the non-reactive closed-loop simulation.

5 Conclusion

We present RaSc, an efficient model for trajectory planning. By incorporating Time-To-Collision as a basis for driving decisions into training, we develop an approach that enables an imitation learning-based model to (1) learn human drivers' risk awareness abilities and the motivations behind their actions; (2) comprehend the consequences of its own actions; (3) identify hard samples during training. RaSc achieves significant closed-loop performance gains and surpasses previous state-of-the-arts. By RaSc, we demonstrate a way of combining traditional safety metrics with scalable learning algorithms, to bridge the prominent mismatch between open-loop training and closed-loop deployment of planning models, thus creating more reliable and aware autonomous driving systems.

Acknowledgements

This study is supported by Tsinghua University-Toyota Joint Research Center for AI Technology of Automated Vehicle (TTAD 2023-07).

References

1. Andersson, J.A., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M.: Casadi: a software framework for nonlinear optimization and optimal control. Mathematical Programming Computation **11**, 1–36 (2019)
2. Bansal, M., Krizhevsky, A., Ogale, A.: Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. arXiv preprint arXiv:1812.03079 (2018)
3. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: CVPR. pp. 11621–11631 (2020)
4. Caesar, H., Kabzan, J., Tan, K.S., Fong, W.K., Wolff, E., Lang, A., Fletcher, L., Beijbom, O., Omari, S.: nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. arXiv preprint arXiv:2106.11810 (2021)
5. Casas, S., Sadat, A., Urtasun, R.: Mp3: A unified model to map, perceive, predict and plan. In: CVPR. pp. 14403–14412 (2021)
6. Chai, Y., Sapp, B., Bansal, M., Anguelov, D.: Multipath: Multiple probabilistifc anchor trajectory hypotheses for behavior prediction. arXiv preprint arXiv:1910.05449 (2019)
7. Chang, M.F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al.: Argoverse: 3d tracking and forecasting with rich maps. In: CVPR. pp. 8748–8757 (2019)
8. Chen, D., Krähenbühl, P.: Learning from all vehicles. In: CVPR. pp. 17222–17231 (2022)
9. Chen, D., Zhou, B., Koltun, V., Krähenbühl, P.: Learning by cheating. In: CoRL. pp. 66–75. PMLR (2020)
10. Cheng, J., Chen, Y., Mei, X., Yang, B., Li, B., Liu, M.: Rethinking imitation-based planner for autonomous driving. arXiv preprint arXiv:2309.10443 (2023)
11. Codevilla, F., Lopez, A.M., Koltun, V., Dosovitskiy, A.: On offline evaluation of vision-based driving models. In: ECCV. pp. 236–251 (2018)
12. Codevilla, F., Santana, E., López, A.M., Gaidon, A.: Exploring the limitations of behavior cloning for autonomous driving. In: ICCV. pp. 9329–9338 (2019)
13. Cui, H., Radosavljevic, V., Chou, F.C., Lin, T.H., Nguyen, T., Huang, T.K., Schneider, J., Djuric, N.: Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In: ICRA. pp. 2090–2096. IEEE (2019)
14. Dauner, D., Hallgarten, M., Geiger, A., Chitta, K.: Parting with misconceptions about learning-based vehicle motion planning. arXiv preprint arXiv:2306.07962 (2023)
15. Deo, N., Wolff, E., Beijbom, O.: Multimodal trajectory prediction conditioned on lane-graph traversals. In: CoRL. pp. 203–212. PMLR (2022)
16. Ettinger, S., Cheng, S., Caine, B., Liu, C., Zhao, H., Pradhan, S., Chai, Y., Sapp, B., Qi, C.R., Zhou, Y., et al.: Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In: ICCV. pp. 9710–9719 (2021)

17. Fan, Y., Liu, X., Li, Y., Wang, S.: Look before you drive: Boosting trajectory forecasting via imagining future. In: IROS. pp. 5551–5558. IEEE (2023)
18. Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., Schmid, C.: Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In: CVPR. pp. 11525–11533 (2020)
19. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: Gohome: Graph-oriented heatmap output for future motion estimation. In: ICRA. pp. 9107–9114. IEEE (2022)
20. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: Thomas: Trajectory heatmap output with learned multi-agent sampling. In: ICLR (2022)
21. Guo, K., Jing, W., Chen, J., Pan, J.: Ccil: Context-conditioned imitation learning for urban driving. arXiv preprint arXiv:2305.02649 (2023)
22. Hallgarten, M., Stoll, M., Zell, A.: From prediction to planning with goal conditioned lane graph traversals. arXiv preprint arXiv:2302.07753 (2023)
23. Hayward, J.C.: Near miss determination through use of a scale of danger (1972)
24. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
25. Hong, J., Sapp, B., Philbin, J.: Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In: CVPR. pp. 8454–8462 (2019)
26. Van der Horst, A., Hogema, J.: Time-to-collision and collision avoidance systems. Verkeersgedrag in Onderzoek (1994)
27. Hu, Y., Li, K., Liang, P., Qian, J., Yang, Z., Zhang, H., Shao, W., Ding, Z., Xu, W., Liu, Q.: Imitation with spatial-temporal heatmap: 2nd place solution for nuplan challenge. In: CVPRW (2023)
28. Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., et al.: Planning-oriented autonomous driving. In: CVPR. pp. 17853–17862 (2023)
29. Huang, Z., Liu, H., Mo, X., Lyu, C.: Gameformer planner: A learning-enabled interactive prediction and planning framework for autonomous vehicles. In: CVPRW (2023)
30. Jia, X., Sun, L., Zhao, H., Tomizuka, M., Zhan, W.: Multi-agent trajectory prediction by combining egocentric and allocentric views. In: CoRL. pp. 1434–1443. PMLR (2022)
31. Jia, X., Wu, P., Chen, L., Liu, Y., Li, H., Yan, J.: Hdgt: Heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding. IEEE TPAMI (2023)
32. Jiang, B., Chen, S., Xu, Q., Liao, B., Chen, J., Zhou, H., Zhang, Q., Liu, W., Huang, C., Wang, X.: Vad: Vectorized scene representation for efficient autonomous driving. In: ICCV (2023)
33. Khandelwal, S., Qi, W., Singh, J., Hartnett, A., Ramanan, D.: What-if motion prediction for autonomous driving. arXiv preprint arXiv:2008.10587 (2020)
34. Konev, S., Brodt, K., Sanakoyeu, A.: Motioncnn: A strong baseline for motion prediction in autonomous driving. arXiv preprint arXiv:2206.02163 (2022)
35. Lee, D.N.: A theory of visual control of braking based on information about time-to-collision. Perception **5**(4), 437–459 (1976)
36. Lee, N., Choi, W., Vernaza, P., Choy, C.B., Torr, P.H., Chandraker, M.: Desire: Distant future prediction in dynamic scenes with interacting agents. In: CVPR. pp. 336–345 (2017)
37. Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., Urtasun, R.: Learning lane graph representations for motion forecasting. In: ECCV. pp. 541–556. Springer (2020)

38. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV. pp. 2980–2988 (2017)
39. Liu, Y., Zhang, J., Fang, L., Jiang, Q., Zhou, B.: Multimodal motion prediction with stacked transformers. In: CVPR. pp. 7577–7586 (2021)
40. Mercat, J., Gilles, T., El Zoghby, N., Sandou, G., Beauvois, D., Gil, G.P.: Multi-head attention for multi-modal joint vehicle motion forecasting. In: ICRA. pp. 9638–9644. IEEE (2020)
41. Mo, X., Huang, Z., Xing, Y., Lv, C.: Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network. IEEE TITS **23**(7), 9554–9567 (2022)
42. Ngiam, J., Vasudevan, V., Caine, B., Zhang, Z., Chiang, H.T.L., Ling, J., Roelofs, R., Bewley, A., Liu, C., Venugopal, A., et al.: Scene transformer: A unified architecture for predicting future trajectories of multiple agents. In: ICLR (2022)
43. Ohn-Bar, E., Prakash, A., Behl, A., Chitta, K., Geiger, A.: Learning situational driving. In: CVPR. pp. 11296–11305 (2020)
44. Phan-Minh, T., Howington, F., Chu, T.S., Lee, S.U., Tomov, M.S., Li, N., Dicle, C., Findler, S., Suarez-Ruiz, F., Beaudoin, R., et al.: Driving in real life with inverse reinforcement learning. arXiv preprint arXiv:2206.03004 (2022)
45. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR. pp. 652–660 (2017)
46. Renz, K., Chitta, K., Mercea, O.B., Koepke, A., Akata, Z., Geiger, A.: Plant: Explainable planning transformers via object-level representations. arXiv preprint arXiv:2210.14222 (2022)
47. Rhinehart, N., McAllister, R., Levine, S.: Deep imitative models for flexible inference, planning, and control. arXiv preprint arXiv:1810.06544 (2018)
48. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: AISTATS. pp. 627–635. JMLR Workshop and Conference Proceedings (2011)
49. Salzmann, T., Ivanovic, B., Chakravarty, P., Pavone, M.: Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In: ECCV. pp. 683–700. Springer (2020)
50. Scheel, O., Bergamini, L., Wolczyk, M., Osiński, B., Ondruska, P.: Urban driver: Learning to drive from real-world demonstrations using policy gradients. In: CoRL. pp. 718–728. PMLR (2022)
51. Shao, H., Wang, L., Chen, R., Li, H., Liu, Y.: Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In: CoRL. pp. 726–737. PMLR (2023)
52. Shrivastava, A., Gupta, A., Girshick, R.: Training region-based object detectors with online hard example mining. In: CVPR. pp. 761–769 (2016)
53. Treiber, M., Hennecke, A., Helbing, D.: Congested traffic states in empirical observations and microscopic simulations. Physical review E **62**(2), 1805 (2000)
54. Varadarajan, B., Hefny, A., Srivastava, A., Refaat, K.S., Nayakanti, N., Cornman, A., Chen, K., Douillard, B., Lam, C.P., Anguelov, D., et al.: Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In: ICRA. pp. 7814–7821. IEEE (2022)
55. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NeurIPS **30** (2017)
56. Vogel, K.: A comparison of headway and time to collision as safety indicators. Accident analysis & prevention **35**(3), 427–433 (2003)
57. Wen, C., Lin, J., Darrell, T., Jayaraman, D., Gao, Y.: Fighting copycat agents in behavioral cloning from observation histories. NeurIPS **33**, 2564–2575 (2020)

58. Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J.K., et al.: Argoverse 2: Next generation datasets for self-driving perception and forecasting. arXiv preprint arXiv:2301.00493 (2023)
59. Wu, H., Phong, T., Yu, C., Cai, P., Zheng, S., Hsu, D.: What truly matters in trajectory prediction for autonomous driving? arXiv preprint arXiv:2306.15136 (2023)
60. Wu, P., Jia, X., Chen, L., Yan, J., Li, H., Qiao, Y.: Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. NeurIPS **35**, 6119–6132 (2022)
61. Xi, W., Shi, L., Cao, G.: An imitation learning method with data augmentation and post processing for planning in autonomous driving. In: CVPRW (2023)
62. Xu, H., Gao, Y., Yu, F., Darrell, T.: End-to-end learning of driving models from large-scale video datasets. In: CVPR. pp. 2174–2182 (2017)
63. Ye, M., Cao, T., Chen, Q.: Tpcn: Temporal point cloud networks for motion forecasting. In: CVPR. pp. 11318–11327 (2021)
64. Zeng, W., Liang, M., Liao, R., Urtasun, R.: Lanercnn: Distributed representations for graph-centric motion forecasting. In: IROS. pp. 532–539. IEEE (2021)
65. Zeng, W., Wang, S., Liao, R., Chen, Y., Yang, B., Urtasun, R.: Dsdnet: Deep structured self-driving network. In: ECCV. pp. 156–172. Springer (2020)
66. Zhai, J.T., Feng, Z., Du, J., Mao, Y., Liu, J.J., Tan, Z., Zhang, Y., Ye, X., Wang, J.: Rethinking the open-loop evaluation of end-to-end autonomous driving in nuscenes. arXiv preprint arXiv:2305.10430 (2023)
67. Zhou, J., Wang, R., Liu, X., Jiang, Y., Jiang, S., Tao, J., Miao, J., Song, S.: Exploring imitation learning for autonomous driving with feedback synthesizer and differentiable rasterization. In: IROS. pp. 1450–1457. IEEE (2021)
68. Zhou, Z., Ye, L., Wang, J., Wu, K., Lu, K.: Hivt: Hierarchical vector transformer for multi-agent motion prediction. In: CVPR. pp. 8823–8833 (2022)