

PLUTO: Pushing the Limit of Imitation Learning-based Planning for Autonomous Driving

Jie Cheng, Yingbing Chen, and Qifeng Chen

Abstract—We present PLUTO, a powerful framework that Pushes the Limit of imitation learning-based planning for aUTOonomous driving. Our improvements stem from three pivotal aspects: a longitudinal-lateral aware model architecture that enables flexible and diverse driving behaviors; An innovative auxiliary loss computation method that is broadly applicable and efficient for batch-wise calculation; A novel training framework that leverages contrastive learning, augmented by a suite of new data augmentations to regulate driving behaviors and facilitate the understanding of underlying interactions. We assessed our framework using the large-scale real-world nuPlan dataset and its associated standardized planning benchmark. Impressively, PLUTO achieves state-of-the-art closed-loop performance, beating other competing learning-based methods and surpassing the current top-performed rule-based planner for the first time. Results and code are available at <https://jchengai.github.io/pluto>.

Index Terms—Autonomous driving, imitation learning, learning-based planning.

I. INTRODUCTION

LEARNING-based planning has emerged as a potentially scalable approach for autonomous driving, attracting significant research interest [1]. Imitation-based planning, in particular, has demonstrated noteworthy success in simulations and real-world applications. Yet, the efficacy of learning-based planning remains unsatisfactory. As indicated in [2], conventional rule-based planning outperforms all learning-based alternatives, winning the 2023 nuPlan planning challenge. This paper delineates the principal challenges inherent in learning-based planning and presents our novel solutions, aimed at pushing the boundaries of what is achievable with learning-based planning.

The first challenge lies in acquiring multi-modal driving behaviors. It is observed that while learning-based planners are good at learning longitudinal tasks such as lane following, they struggle with lateral tasks [3], for instance, executing lane changes or navigating around obstacles, even when space permits. We attribute this deficiency to the absence of explicit lateral behavior modeling within the architectural design of the model. Our previous work [4] attempted to address this issue by generating plans that are explicitly conditioned on

nearby reference lines, though it was restricted to producing a maximum of three proposals and did not effectively integrate lateral and longitudinal behavior modeling. In the present study, we enhance this approach through the adoption of a query-based architecture capable of generating an extensive array of proposals by fusing longitudinal and lateral queries. The advanced model design enables our planner to demonstrate diverse and flexible driving behaviors, which we believe is a crucial step toward practical learning-based planning.

Beyond the model architecture, it is recognized that pure imitation learning encompasses inherent limitations, including the propensity for learning shortcuts [5]–[8], distribution shift [3], [5], [9], and causal confusion [10], [11] issues. This paper addresses these pervasive challenges across three dimensions:

(1) *Learning beyond pure imitation loss*. We concur with previous studies [5], [9], [12] that solely relying on imitation loss is insufficient for learning desired driving behaviors. It is imperative to impose explicit constraints during the training phase, particularly within the safety-critical realm of autonomous driving. A prevalent approach is to add auxiliary losses to penalize adverse behaviors, such as collisions and off-road driving, as previously demonstrated in [5], [9]. However, their methods are either designed for heatmap-based output [5] or need a differentiable rasterizer [9] that renders each trajectory point into an image. As a result, the output resolution is restricted to reduce the computation burden. It remains unclear how to realize these losses efficiently for the more modern vector-based models. To bridge this gap, we introduce a novel auxiliary loss calculation methodology predicated on differentiable interpolation. This method not only spans a broad spectrum of auxiliary tasks but also facilitates batch-wise computation within modern deep-learning frameworks, thereby enhancing its applicability and efficiency.

(2) *New data augmentations*. Issues arise when the model undergoes open-loop training and closed-loop testing. For instance, the accumulation of errors over time may lead to input data deviating from the training distribution; the model may rely on unintended shortcuts rather than acquiring knowledge. Data augmentations have been extensively employed to alleviate these problems and have demonstrated effectiveness, e.g., perturbation-based augmentations [3], [5], [9] teach the model to learn to recover from small deviations and dropout-based augmentations prevent learning shortcuts [3]. In addition to these two augmentations, we introduce further augmentation techniques aimed at regulating driving behavior and enhancing interaction learning.

(3) *Learning by contrast*. Imitation learning-based models

Jie Cheng is with the Department of Electronic and Computer Engineering, the Hong Kong University of Science and Technology, Hong Kong SAR (E-mail: jchengai@connect.ust.hk)

Yingbing Chen is with Division of Emerging Interdisciplinary Areas, the Hong Kong University of Science and Technology, Hong Kong SAR (E-mail: ychengz@connect.ust.hk).

Qifeng Chen is with the Department of Computer Science and Engineering, the Hong Kong University of Science and Technology, Hong Kong SAR (E-mail: cqf@ust.hk).

often struggle to recognize underlying causal relationships due to the absence of interactive feedback with the environment [10]. This issue can significantly hinder performance; for instance, a planner may decelerate by mimicking the behavior of nearby agents rather than responding to a red light. Our goal is to address this issue without substantially complicating the training process, as would be the case with reinforcement learning or employing a data-driven simulator. Drawing inspiration from the effectiveness of contrastive learning [13], which enhances representation by differentiating between similar and dissimilar examples, we recognize an opportunity to infuse the model with causal understanding. This is achieved by enabling the model to differentiate between original and modified input data—for example, by excluding the leading vehicles from the autonomous vehicle’s (AV’s) perspective. Building on this approach and the two previously mentioned strategies, we introduce a novel unified framework termed Contrastive Imitation Learning (CIL).

In summary, this study introduces a comprehensive, data-driven planning framework named PLUTO, designed to Push the Limit of imitation learning-based planning for a UTonomous driving. PLUTO incorporates innovative solutions in model architecture, data augmentation, and the learning framework. It has been evaluated using the large-scale real-world nuPlan [14] dataset, where it has demonstrated superior closed-loop performance. Notably, PLUTO surpasses the existing state-of-the-art rule-based planner PDM [2] for the first time, marking a significant milestone in the field. Our main contributions are:

- We introduce a query-based model architecture that simultaneously addresses lateral and longitudinal planning maneuvers, enabling flexible and diverse driving behaviors.
- We propose a novel method for calculating auxiliary loss based on differential interpolation. This method is applicable to a broad spectrum of auxiliary tasks and allows for efficient batch-wise computation in vector-based models.
- We present the Contrastive Imitation Learning (CIL) framework, accompanied by a new set of data augmentations. The CIL framework is aimed at regulating driving behaviors and enhancing interaction learning, without significantly increasing the complexity of training.
- Our evaluation on the large-scale nuPlan dataset demonstrates that PLUTO achieves state-of-the-art performance in closed-loop planning. Our model and benchmark are publicly available.

II. RELATED WORK

A. Imitation-based Planning

Learning to drive by cloning the policies of experienced drivers is likely the most direct and scalable solution to autonomous driving, considering the abundance and affordability of data today. One of the popular methods is end-to-end (E2E) driving [15]. This approach directly learns driving policies from raw sensor data and has made significant strides in a relatively short period. Initially, the focus was on convolutional neural network (CNN)-based models [16], [17] that mapped camera inputs to control policies. This

evolved to incorporate more sophisticated methods [18]–[21] that utilized multi-sensor fusion. More recently, developments spearheaded by entities such as LAV [22] and UniAD [23] have shifted towards a module-based E2E architecture. This approach integrates the processes of perception, prediction, and planning within a unified model [24], [25]. Despite their potential, most E2E strategies extensively depend on high-fidelity simulation environments like CARLA [26] for both training and evaluation. Consequently, these methodologies are plagued by several issues, including a lack of realism and diversity in simulated agents, reliance on imperfect rule-based experts, and the imperative need to bridge the simulation-to-reality gap for applicability in real-world scenarios.

This paper focuses on another research direction, commonly referred to as the mid-to-mid approach, which employs post-perception results as input features. The primary advantage of this method is that the model can concentrate on learning to plan and be trained with real-world data, eliminating sim-to-real transfer concerns. Pioneering approaches such as ChauffeurNet [5], SafetyNet [27], and UrbanDriver [28] have demonstrated the ability to operate autonomous vehicles in real-world environments, with subsequent works building upon these foundations [3], [4], [29]–[31]. These approaches benefit significantly from advancements in the motion forecasting community, including the adoption of vector-based models [4], [28] that excel in prediction tasks, replacing early planning models based on rasterized bird’s-eye-view images [5], [9]. However, many of these models overlook the inherent characteristics of planning tasks, such as the need for closed-loop testing and active decision-making capabilities. In contrast, our proposed framework is specifically designed for planning from the outset. Our network jointly models longitudinal and lateral driving behaviors through a query-based architecture, enabling a flexible and diverse driving style.

Prior studies [5], [9]–[11] have demonstrated the limitations of basic imitation learning and suggested methods for enhancement. In order to address compounding errors, an early solution can be traced back to DAGger [32], which interactively refines the trained model by incorporating additional expert demonstrations. Subsequently, ChauffeurNet [5] introduced perturbation-based augmentation, enabling the model to recover from minor deviations and establishing a standard practice for later research. Adding auxiliary losses such as collision loss and off-road loss [5], [9], [28] is another important aspect to improve the overall performance. However, their method is either designed for heatmap output [5] or requires a differentiable rasterizer [9] that converts the trajectory into a sequence of images with kernel functions. These approaches are not efficiently applicable to vector-based methods. Our research contributes to this field by introducing a novel technique that employs differentiable interpolation to bridge this gap.

B. Contrastive Learning

Contrastive learning [33] is a framework that learns representation by comparing similar and dissimilar pairs, achieving significant success in computer vision [13], [34] and natural language processing [35]. Within the context of autonomous

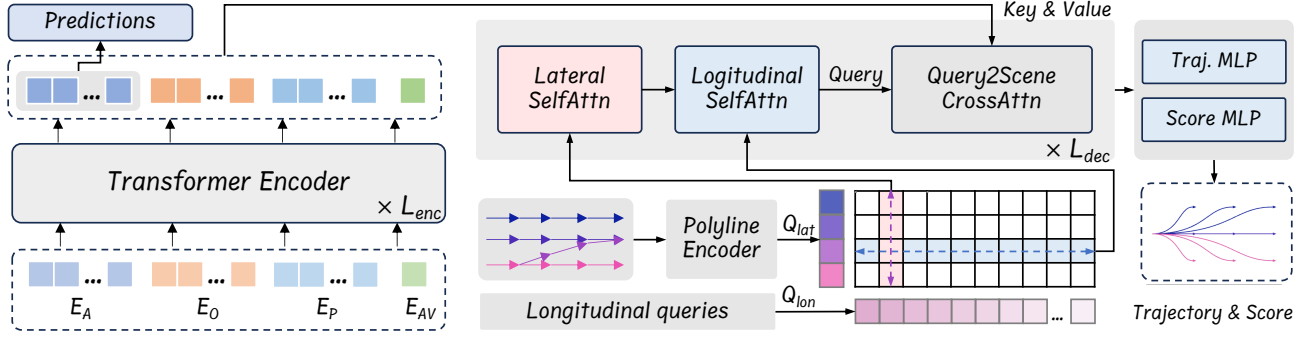


Fig. 1. The architectural overview of the PLUTO model is presented in this section. The model initiates lateral queries Q_{lat} using a polyline encoder based on adjacent reference lines. Simultaneously, longitudinal queries Q_{lon} are established as learnable embeddings. These queries undergo a fusion process via factorized lateral-longitudinal self-attention layers. This integration serves as a basis for the subsequent decoding of trajectories and their associated scores.

driving, a few attempts have been made for motion prediction. Social NCE [36] introduced a social contrastive loss to guide goal generation in pedestrian motion forecasting. Marah *et al.* [37] utilized action-based contrastive learning loss to refine learned trajectory embeddings. FEND [38] employed this approach to recognize long-tail trajectories. These studies underscore the potency of contrastive learning in incorporating domain-specific knowledge into models through the careful selection of positive and negative examples. In our research, we extend its application to the planning domain, aiming to improve driving behavior predictions and facilitate the understanding of implicit interactions among vehicles. As generating negative samples is crucial to contrastive methods, we also introduce a new set of data augmentation functions that defines the contrastive task.

III. METHODOLOGY

A. Problem Formulation

In this study, we explore the task of autonomous driving within dynamic urban settings, considering the autonomous vehicle (AV), N_A dynamic agents, N_S static obstacles, a high-definition map M , and other traffic-related contexts C such as traffic light status. We define the features of agents as $\mathcal{A} = A_{0:N_A}$, where A_0 represents the AV, and static obstacles are denoted by $\mathcal{O} = O_{1:N_S}$. Additionally, we denote the future state of agent a at time t as $\mathbf{y}_a^t$, with the historical and future horizons represented by T_H and T_F , respectively. Our proposed system, PLUTO, is designed to simultaneously generate N_T multi-modal planning trajectories for the AV and a prediction for each dynamic agent. The selection of the final output trajectory, τ^* , is executed by a scoring module, S , which integrates learning-based outcomes with all scene contexts. PLUTO is formulated as follows:

$$\begin{aligned} (\mathbf{T}_0, \boldsymbol{\pi}_0), \mathbf{P}_{1:N_A} &= f(\mathcal{A}, \mathcal{O}, M, C | \phi), \\ (\tau^*, \pi^*) &= \arg \max_{(\tau, \pi) \in (\mathbf{T}_0, \boldsymbol{\pi}_0)} \mathcal{S}(\tau, \pi, \mathbf{P}_{1:N_A}, \mathcal{O}, M, C), \end{aligned} \quad (1)$$

where f denotes the neural network of PLUTO, ϕ is the model parameters, $(\mathbf{T}_0, \boldsymbol{\pi}_0) = \{(\mathbf{y}_{0,i}^{1:T_F}, \pi_i) | i = 1 \dots N_T\}$ is AV's planning trajectories and corresponding confidence scores, $\mathbf{P}_{1:N_A} = \{\mathbf{y}_a^{1:T_F} | a = 1 \dots N_A\}$ are agents' predictions.

The subsequent sections provide a detailed illustration of each component within the PLUTO framework.

B. Input Representation and Scene Encoding

Agent History Encoding. The observational state of each agent at any given time t is denoted as $\mathbf{s}_i^t = (\mathbf{p}_i^t, \theta_i^t, \mathbf{v}_i^t, \mathbf{b}_i^t, \mathbb{I}_i^t)$, where $\mathbf{p}$ and θ represent the agent's position coordinates and heading angle, respectively; $\mathbf{v}$ refers to the velocity vector, $\mathbf{b}$ defines the dimensions (length and width) of the perception bounding box; and $\mathbb{I}$ is a binary indicator signifies the observation status of this frame. We convert the history sequence into vector form by calculating the difference between consecutive time steps: $\hat{\mathbf{s}}_i^t = (\mathbf{p}_i^t - \mathbf{p}_i^{t-1}, \theta_i^t - \theta_i^{t-1}, \mathbf{v}_i^t - \mathbf{v}_i^{t-1}, \mathbf{b}_i^t, \mathbb{I}_i^t)$, resulting in agent's feature vector $F_A \in \mathbb{R}^{N_A \times (T_H - 1) \times 8}$. To extract and condense these historical features, we employ a neighbor attention-based Feature Pyramid Network (FPN) [39], which produces an agent embedding $E_A \in \mathbb{R}^{N_A \times D}$, with D representing the dimensionality of the hidden layers used consistently throughout this paper.

Static Obstacles Encoding. In contrast to motion forecasting tasks where static obstacles are often overlooked, the presence of static obstacles is crucial for ensuring safe navigation. Static obstacles encompass any entities that an AV must not traverse, such as traffic cones or barriers. Each static obstacle within the drivable area is represented by $\mathbf{o}_i = (\mathbf{p}_i, \theta_i, \mathbf{b}_i)$. We use a two-layer multi-layer-perceptron (MLP) to encode static objects features $F_O \in \mathbb{R}^{N_S \times 5}$, resulting in embedding $E_O \in \mathbb{R}^{N_S \times D}$.

AV's State Encoding. Drawing on insights from previous studies [3], [8] that imitation learning tends to adopt shortcuts from historical states, thereby detrimentally affecting performance, our approach only utilize the current state of AV as the input feature. This current state encompasses the AV's position, heading angle, velocity, acceleration, and steering angle. To encode the state feature while avoiding the generation of trajectories based on extrapolated kinematic states, we employ an attention-based state dropout encoder (SDE), as suggested in [3]. The encoded AV's embedding is $E_{AV} \in \mathbb{R}^{1 \times D}$.

Vectorized Map Encoding. The map consists of N_P polylines. These polylines undergo an initial subsampling process to standardize the quantity of points, followed by the computation of a feature vector for each point. Specifically, for

each polyline, the feature of the i -th point encompasses eight channels: $(\mathbf{p}_i - \mathbf{p}_0, \mathbf{p}_i - \mathbf{p}_{i-1}, \mathbf{p}_i - \mathbf{p}_i^{\text{left}}, \mathbf{p}_i - \mathbf{p}_i^{\text{right}})$. Here, $\mathbf{p}_0$ denotes the initial point of the polyline, while $\mathbf{p}_i^{\text{left}}$ and $\mathbf{p}_i^{\text{right}}$ represent the left and right boundary points of the lane, respectively. Incorporating the boundary feature is crucial as it conveys information about the drivable area, essential for planning tasks. The features of the polylines are represented as $F_P \in \mathbb{R}^{N_P \times n_p \times 8}$, where n_p denotes the number of points per polyline. To encode the map features, a PointNet-like [40] polyline encoder is employed, resulting in an encoded feature space $E_P \in \mathbb{R}^{N_P \times D}$.

Scene Encoding. To effectively capture the intricate interactions among various modal inputs, we concatenate different embeddings into a single tensor $E_0 \in \mathbb{R}^{(N_A + N_S + N_P + 1) \times D}$. This tensor is subsequently integrated using a series of L_{enc} Transformer encoders. Due to the vectorization process, the input features are stripped of their global positional information. To counteract this loss, a global positional embedding, denoted as PE , is introduced to each embedding. Following [41], PE represents the Fourier embedding of the global position $(\mathbf{p}, \theta)$, utilizing the most recent positions of agents and static obstacles as well as the initial point of polylines. Additionally, to encapsulate inherent semantic attributes such as agent types, lane speed limits, and traffic light statuses, learnable embeddings E_{attr} are incorporated alongside the input embeddings. E_0 is initialized as

$$E_0 = \text{concat}(E_{AV}, E_A, E_O, E_P) + PE + E_{attr}. \quad (2)$$

The i -th layer of the Transformer encoder is formulated as

$$\begin{aligned} \hat{E}_{i-1} &= \text{LayerNorm}(E_{i-1}), \\ E_i &= E_{i-1} + \text{MHA}(\hat{E}_{i-1}, \hat{E}_{i-1}, \hat{E}_{i-1}), \\ E_i &= E_i + \text{FFN}(\text{LayerNorm}(E_i)), \end{aligned} \quad (3)$$

where $\text{MHA}(q, k, v)$ is the standard multi-head attention [42] function, FFN is the feedforward network layer. We denote E_{enc} as the output of the final layer of the encoder.

C. Multi-modal Planning Trajectory Decoding

The task of planning in autonomous driving is inherently multimodal, as there are often multiple valid behaviors that could be adopted in response to a given driving scenario. For instance, a vehicle might either continue to follow a slower vehicle ahead or opt to change lanes and overtake it. To address this complex issue, we utilize a query-based, DETR-like [43] trajectory decoder. However, directly implementing the learned anchor-free queries has been found to result in mode collapse and training instability, as evidenced in [44]. Drawing inspiration from the observation that driving behaviors can be decomposed into combinations of lateral (e.g., lane changing) and longitudinal (e.g., braking and accelerating) actions, we introduce a semi-anchor-based decoding structure. An illustrative overview of our decoding pipeline is presented in Fig. 1, with subsequent paragraphs detailing the individual components.

Reference Lines as Lateral Queries. Following the methodology outlined in [4], this study employs reference lines as

a high-level abstraction for lateral queries. Reference lines, typically derived from the autonomous vehicle's surrounding lanes on its route, serve as a critical component in conventional vehicle motion planning, guiding lateral driving behaviors. Initially, we identify lane segments within a radius of R_{ref} from the AV's current position. Starting from each identified lane segment, a depth-first search is conducted to explore all potential topological connections, linking their respective lane centerlines. Subsequently, these connected centerlines are truncated to a uniform length and are resampled to maintain a consistent number of points. The approach for representing and encoding the features of reference lines mirrors that of vectorized map encoding, as outlined in Section III-B. Ultimately, the embedded reference lines are utilized as the lateral query $Q_{lat} \in \mathbb{R}^{N_R \times D}$, where N_R represents the number of reference lines.

Factorized Lateral-longitudinal Self-Attention. In addition to Q_{lat} , we employ N_L anchor-free, learnable queries $Q_{lon} \in \mathbb{R}^{N_L \times D}$ to encapsulate the multi-modal nature of longitudinal behaviors. Following this, Q_{lat} and Q_{lon} are combined to create the initial set of lateral-longitudinal queries, denoted as $Q_0 \in \mathbb{R}^{N_R \times N_L \times D}$:

$$Q_0 = \text{Projection}(\text{concat}(Q_{lat}, Q_{lon})), \quad (4)$$

where Projection refers to either a simple linear layer or a multilayer perceptron. Since each query within Q_0 captures only the local region information pertaining to an individual reference line, we utilize self-attention mechanisms on Q_0 to integrate global lateral-longitudinal information across various reference lines. Nonetheless, applying self-attention directly to Q_0 results in computational complexity of $\mathcal{O}(N_R^2 N_L^2)$, which becomes prohibitively high as N_R and N_L increase. Drawing inspiration from similar approaches in the literature [45], we adopt a factorized attention strategy across each axis of Q , effectively reducing the computational complexity to $\mathcal{O}(N_R^2 N_L + N_R N_L^2)$.

Trajectory Decoding. The trajectory decoder consists of a sequence of L_{dec} decoding layers, each comprising three types of attention mechanisms: lateral self-attention, longitudinal self-attention, and query-to-scene cross-attention. These processes are mathematically represented as follows:

$$\begin{aligned} Q'_{i-1} &= \text{SelfAttn}(Q_{i-1}, \text{dim} = 0), \\ \hat{Q}_{i-1} &= \text{SelfAttn}(Q'_{i-1}, \text{dim} = 1), \\ Q_i &= \text{CrossAttn}(\hat{Q}_{i-1}, E_{enc}, E_{enc}). \end{aligned} \quad (5)$$

Here, $\text{SelfAttn}(X, \text{dim} = i)$ indicates the application of self-attention across the i -th dimension of X , and $\text{CrossAttn}(Q, K, V)$ incorporates layer normalization, multi-head attention, and a feed-forward network, analogous to the structure defined in Eq. 3. The decoder's final output, Q_{dec} , is then employed to determine the AV's future trajectory points and their respective scores using two MLPs:

$$T_0 = \text{MLP}(Q_{dec}), \quad \pi_0 = \text{MLP}(Q_{dec}). \quad (6)$$

Each decoded trajectory point has six channels: $[p_x, p_y, \cos \theta, \sin \theta, v_x, v_y]$. Furthermore, to accommodate

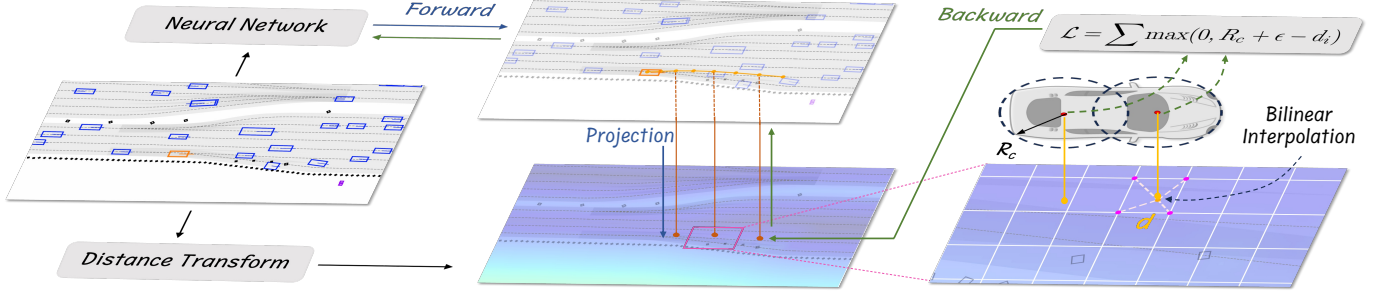


Fig. 2. Illustration of the proposed auxiliary loss computation method. Initially, the trajectory produced by the neural network is mapped onto the image space associated with the cost map. Subsequently, the cost value is obtained via bilinear interpolation and employed in the formation of the loss function. Given the differentiable nature of all processes involved, it is feasible to incorporate auxiliary tasks directly into the framework, allowing for end-to-end training.

scenarios lacking reference lines, an additional MLP head is introduced to directly decode a single trajectory from the encoded features of the AV:

$$\tau^{free} = \text{MLP}(E'_{AV}). \quad (7)$$

Imitation Loss. To avoid mode collapse, we employ the teacher-forcing [46] technique during the training process. Firstly, the endpoint of the ground truth trajectory τ^{gt} is projected relative to reference lines, with the selection of the reference line closest in lateral distance serving as the target reference line. This target reference line is subsequently divided into $N_L - 1$ equal segments by distance. Each segment corresponds to the region managed by each longitudinal query, with the final query accounting for regions extending beyond the target reference line. The query encompassing the projected endpoint is designated as the target query. By integrating the target reference line with the target longitudinal query, we derive the target supervision trajectory, $\hat{\tau}$. For trajectory regression, we employ the smooth L1 loss [47], and for score classification, we utilize the cross-entropy loss, expressed as follows:

$$\begin{aligned} \mathcal{L}_{reg} &= \text{L1}_{smooth}(\hat{\tau}, \tau^{gt}) + \text{L1}_{smooth}(\tau^{free}, \tau^{gt}), \\ \mathcal{L}_{cls} &= \text{CrossEntropy}(\pi_0, \pi_0^*), \end{aligned} \quad (8)$$

where π_0^* signifies the one-hot distribution derived from the index of $\hat{\tau}$. The overall imitation loss is formulated as the sum of these two components, each weighted equally:

$$\mathcal{L}_i = \mathcal{L}_{reg} + \mathcal{L}_{cls}. \quad (9)$$

Prediction Loss. A simple two-layer MLP is used to generate a single modal prediction for each dynamic agent from the encoded agents' embeddings:

$$\mathbf{P}_{1:N_A} = \text{MLP}(E'_A). \quad (10)$$

Firstly, this provides dense supervision which benefits the training [39], [48]. Secondly, the generated predictions play a crucial role in eliminating unsuitable planning proposals during the post-processing stage, as detailed in Section III-F. Denote agent's ground truth trajectory as $\mathbf{P}_{1:N_A}^{gt}$, the prediction loss is

$$\mathcal{L}_p = \text{L1}_{smooth}(\mathbf{P}_{1:N_A}, \mathbf{P}_{1:N_A}^{gt}). \quad (11)$$

D. Efficient Differentiable Auxiliary Loss

As highlighted by earlier studies [5], [12], pure imitation learning does not suffice to preclude undesired outcomes, such as collisions with stationary obstacles or deviations from the drivable path. Therefore, it is essential to incorporate these constraints as auxiliary losses in the model during its training phase. Nevertheless, the integration of these constraints in a manner that is differentiable and enables end-to-end training of the model presents a significant challenge. A frequently adopted method for this purpose is differentiable rasterization. For instance, Zhou *et al.* [9] demonstrates a technique where each trajectory point is converted into rasterized images, using a differentiable kernel function, and subsequently calculates the loss using obstacle masks within the image space. This method, however, is limited by its computational and memory demands, which in turn limits the output resolution (e.g., it permits only large time intervals and short planning horizons). To mitigate these limitations, we propose a novel approach based on differentiable interpolation. This method facilitates the concurrent calculation of auxiliary loss for all trajectory points. We take the drivable area constraint an example to elucidate our proposed method.

Cost Map Construction. The first step of our methodology involves transforming the constraint into a queryable cost-map representation. Specifically, for the drivable area constraint, we employ the widely recognized Euclidean Signed Distance Field (ESDF) for cost representation. This process encompasses mapping the non-drivable areas (e.g., off-road regions) onto an $H \times W$ rasterized binary mask, followed by executing distance transforms on this mask. A distinctive advantage of our approach over existing methods is its elimination of the need to render the trajectory into a series of images, thereby significantly reducing computational demands.

Loss Calculation In accordance with established methodologies in optimization-based vehicle motion planning [49], we model the vehicle's shape using N_c covering circles. The trajectory point determines the centers of these circles, which can be derived in a differentiable manner. As illustrated in Fig. 2, for each covering circle i associated with a trajectory point, we obtain its signed distance value d_i through projection and bilinear interpolation. To ensure adherence to the drivable area constraint, we apply a penalty to the model when d_i falls

Algorithm 1 Drivable Area Loss Pseudo-code

```

# Pytorch style pseudo-code
# traj: Trajectory, [B, T, 4] (x, y, cos, sin)
# offset: constant offset of the centers
# sdf: Signed Distance Feild [B, H, W, 1]
# res: rasterization resolution of the SDF
# Rc: radius of the covering circle
# epsilon: safety threshold

def DrivableAreaLoss(traj, sdf, offset, res, Rc,
  , epsilon):
    H, W = sdf.shape[1:3]
    centers = traj[..., None, :2] + offset *
        traj[..., None, 2:4]

    # projection
    centers_pixel = torch.stack([centers[...,
        0] / resolution, -centers[..., 1] /
        resolution], dim=-1)
    grid = centers_pixel / torch.tensor([W//2,
        H//2])

    # query distance in batch
    distance = F.sample_grid(sdf.unsqueeze(1),
        grid, mode="bilinear").squeeze(1)

    # Hinge loss
    cost = Rc + epsilon - distance
    loss_mask = cost > 0
    cost.masked_fill_(~loss_mask, 0)
    loss = F.l1_loss(cost, torch.zeros_like(
        cost), reduction="none").sum(-1)
    loss = loss.sum() / (loss_mask.sum() + 1e
        -6)

    return loss

```

below the circle's radius R_c . The auxiliary loss is:

$$\mathcal{L}_{aux} = \frac{1}{T_f} \sum_{t=1}^{T_f} \sum_{i=1}^{N_c} \max(0, R_c + \epsilon - d_i^t), \quad (12)$$

where ϵ is a safety threshold. Eq. 12 is also applicable to punish collisions with a slight change to the cost map construction.

In practice, d_i and $\mathcal{L}_{aux}$ can be differentially and efficiently calculated batch-wise with the modern deep learning framework as shown in Algorithm 1. It is important to note that our approach is versatile and not confined to ESDF-based representations alone. Any cost representation that allows for continuous querying, such as potential fields, can be incorporated with an appropriately designed loss function.

E. Contrastive Imitation Learning Framework

We introduce the Contrastive Imitation Learning (CIL) framework, designed to effectively address the challenges of distribution shift and causal confusion within a coherent and straightforward structure. Illustrated in Fig. 3, the CIL framework comprises four essential steps:

- Given a training scenario data sample, denoted as $\mathbf{x}$, we apply both a positive data augmentation module, $\mathcal{T}^+$, and a negative one, $\mathcal{T}^-$, to generate a positive sample $\mathbf{x}^+$

and a negative sample $\mathbf{x}^-$. Positive augmentations are those that preserve the validity of the original ground truth (e.g., see Fig. 4a), while negative augmentations alter the original causal structure, rendering the original ground truth inapplicable.

- The Transformer encoder, as detailed in Sect. III-B, is utilized to derive the latent representations $\mathbf{h}^{(\cdot)}$ of both the original and augmented data samples. Subsequently, these representations are mapped to a new space, represented as $\mathbf{z}, \mathbf{z}^+, \mathbf{z}^-$, by a two-layer MLP projection head.
- A triplet contrastive loss is calculated to enhance the agreement between $\mathbf{z}$ and $\mathbf{z}^+$ while decreasing the similarity between $\mathbf{z}$ and $\mathbf{z}^-$.
- Finally, trajectories for the original and positively augmented data samples are decoded, and both the imitation loss and an auxiliary loss are computed.

In practice, we randomly sample a minibatch of N_{bs} samples. Each sample undergoes positive and negative augmentation, executed by augmentors randomly chosen from the sets $\mathcal{T}^+$ and $\mathcal{T}^-$, respectively. This augmentation triples the total number of samples to $3N_{bs}$. All samples are processed by the same encoder and projection head. Let $\text{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T \mathbf{v} / \|\mathbf{u}\| \|\mathbf{v}\|$ denote the dot product between the l_2 normalized $\mathbf{u}$ and $\mathbf{v}$, the softmax-based triple contrastive loss [50] is defined as:

$$\mathcal{L}_c = -\log \frac{\exp(\text{sim}(\mathbf{z}, \mathbf{z}^+)/\sigma)}{\exp(\text{sim}(\mathbf{z}, \mathbf{z}^+)/\sigma) + \exp(\text{sim}(\mathbf{z}, \mathbf{z}^-)/\sigma)}, \quad (13)$$

where σ denotes the temperature parameter. The contrastive loss is computed across all triplets in the mini-batch. Besides this, we provide supervision to both the original and positively augmented samples using the unmodified ground truth trajectory. Note that negatively augmented samples are only used to calculate the contrastive loss as their origin ground truth may be invalid after augmentation. The overall training loss comprises four components: imitation loss, prediction loss, auxiliary loss, and contrastive loss, represented as:

$$\mathcal{L} = w_1 \mathcal{L}_i + w_2 \mathcal{L}_p + w_3 \mathcal{L}_{aux} + w_4 \mathcal{L}_c. \quad (14)$$

Data augmentations. Data augmentation is the key for contrastive learning to work. While perturbation-based augmentations are prevalent, alternative augmentation strategies remain insufficiently explored. In this context, we present six carefully crafted augmentation functions that defines the contrastive task, with illustrative examples provided in Fig. 4.

- (1) *State Perturbation* $\in \mathcal{T}^+$ (Fig. 4a): introduces minor, randomly generated disturbances to the autonomous vehicle's current position, velocity, acceleration, and steering angle. This augmentation is intended to enable the model to learn recovery strategies for slight deviations from the training distribution. The CIL framework aims to maximize the similarity between the latent representations of original and augmented samples, thereby enhancing the model's resilience to error accumulation.
- (2) *Non-interactive Agents Dropout* $\in \mathcal{T}^+$ (Fig. 4b): omits agents from the input scenario that do not interact with

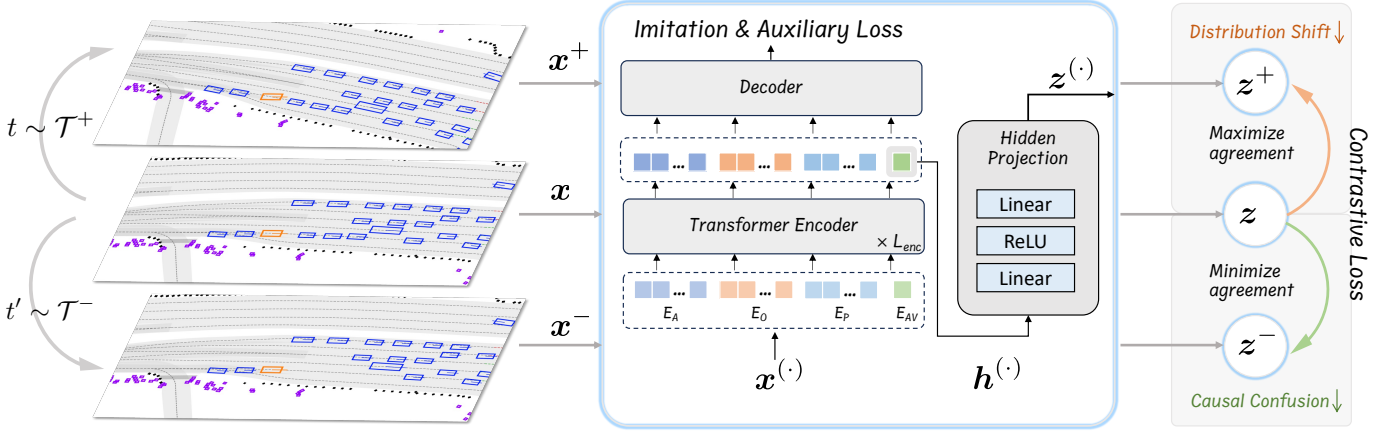


Fig. 3. Illustration of the proposed contrastive imitation learning (CIL) framework. For any input data, we apply two data augmentation functions from different augmentation modules ($t \sim \mathcal{T}^+$ and $t' \sim \mathcal{T}^-$) to obtain a positive sample and a negative sample. The projected latent embeddings $z^{(\cdot)}$ are used to calculate the conservative loss which maximizes the agreement between z^+ and z and minimizes the agreement of z^- and z .

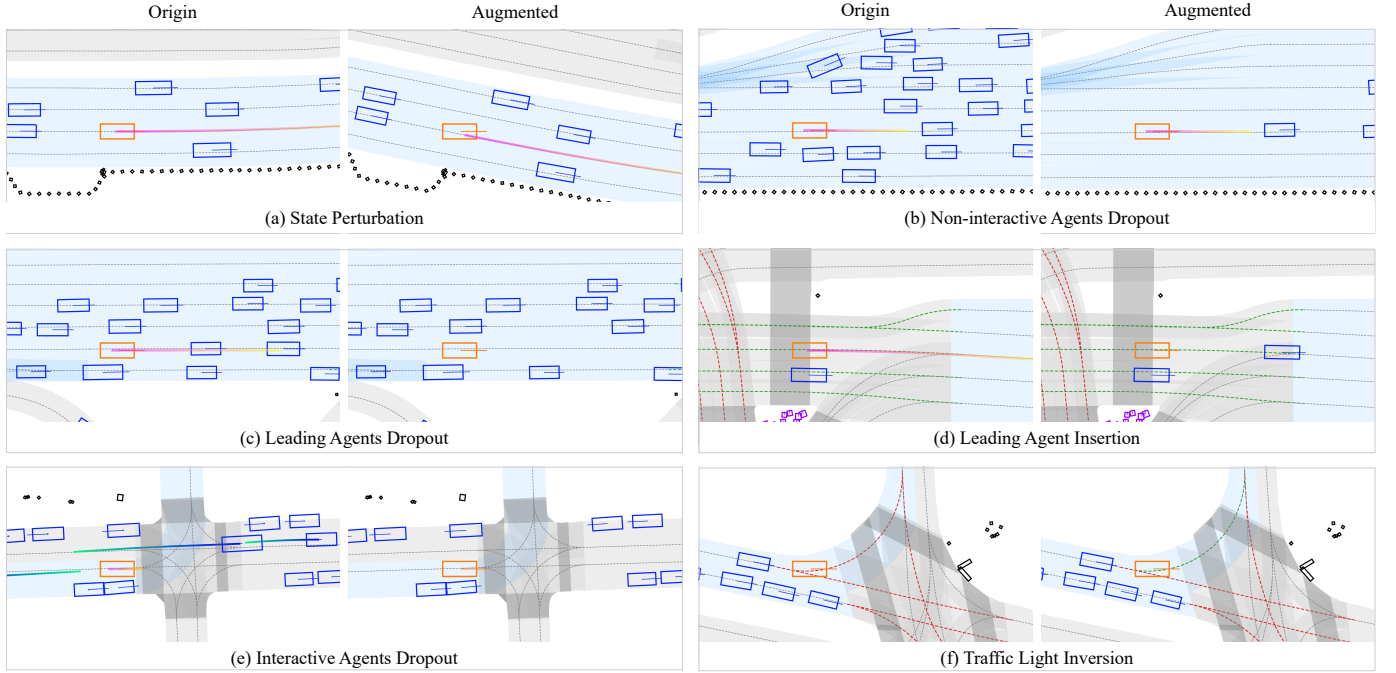


Fig. 4. Exemplary scenarios of the proposed data augmentations. In each group, the figure on the left denotes the origin scenario and the right one shows the augmented scene. AV is marked in the orange, other vehicle agents are in blue, and dash-colored lines on the lanes denote the traffic light status. (a)-(b) belongs to the positive augmentations $\mathcal{T}^+$ and (c)-(f) are negative augmentations $\mathcal{T}^-$.

the AV in the near future. Interactive agents are identified through the intersection of their future bounding boxes with the AV's trajectory. This augmentation prevents the model from learning behaviors by mimicking non-interactive agents, thereby encouraging the model to discern genuine causal relationships with interactive agents.

- (3) *Leading Agents Dropout* $\in \mathcal{T}^-$ (Fig. 4c): removes all agents located ahead of the AV. Special consideration is given to leading-following dynamics, a prevalent situation in real-world driving. This augmentation trains the model on leading-following behaviors to prevent rear-end collisions.

- (4) *Leading Agent Insertions* $\in \mathcal{T}^-$ (Fig. 4d): introduces a leading vehicle into the AV's original path, at a position where the AV's expected trajectory becomes invalid (e.g., would result in a collision). The inserted vehicle's trajectory data is sourced from a randomly selected agent in the current mini-batch to maintain data realism.

- (5) *Interactive Agent Dropout* $\in \mathcal{T}^-$ (Fig. 4e): excludes agents that have direct or indirect interactions with the AV. Identification of interactive agents follows the methodology outlined in *Non-interactive Agents Dropout*. This function aims to train the model on less intuitive interactions within complex scenarios, such as unprotected left

Algorithm 2 Trajectory planning process of PLUTO

Input: Init state $\mathbf{y}_0$, scenario feature $\mathbf{x}$, constant K, α, N_T
procedure TRAJECTORY SELECTION
 $\mathbf{T}_0, \boldsymbol{\pi}_0, \mathbf{P}_{1:N_A} = \text{PLUTO}(\mathbf{x})$ ▷ Run model inference
 $\mathbf{T}_0 = \text{TopK}(\mathbf{T}_0, \boldsymbol{\pi}_0, K)$ ▷ Select Top-K trajectories
 Initialize rollouts $\tilde{\mathbf{T}}_0 = [\mathbf{y}_0]$
 for t in $1 \dots N_T$ **do** ▷ Forward Simulation
 $\mathbf{u}_0 = \text{LQRTracker}(\mathbf{y}_0, \mathbf{T}_0, t)$
 $\mathbf{y}_0 = \text{BicycleModel}(\mathbf{y}_0, \mathbf{u}_0)$
 $\tilde{\mathbf{T}}_0 = [\tilde{\mathbf{T}}_0, \mathbf{y}_0]$
 end for
 $\pi_{rule} = \text{RuleBasedEvaluator}(\tilde{\mathbf{T}}_0, \mathbf{P}_{1:N_A}, \mathbf{x})$
 $\tau^* = \mathbf{T}_0[i], i = \text{argmax}(\pi_{rule} + \alpha\pi_0)$
 return τ^*
end procedure

turns and lane changes.

- (6) *Traffic Light Inversion* $\in \mathcal{T}^-$ (Fig. 4f): in scenarios where the AV approaches an intersection governed by traffic lights without a leading vehicle, the traffic light status is reversed (e.g., from red to green). This function teaches the model to adhere to basic traffic light rules.

These augmentation functions are designed with minimal inductive bias to ensure broad applicability. The contrastive learning task facilitates an implicit feedback mechanism, providing implicit reward signals. These signals reinforce adherence to fundamental driving principles.

F. Planning and Post-processing

In the context of trajectory planning, our objective is to select a deterministic future trajectory from the diverse outcomes provided by the multi-modal outputs, as discussed in Section III-C. Rather than merely selecting the most likely trajectory, we integrate a post-processing module to serve as an additional safety verification mechanism, as illustrated in Algorithm 2.

Upon extracting the scenario’s features, the model is executed to generate multi-modal planning trajectories $\mathbf{T}_0 \in \mathbb{R}^{N_R N_L \times T_F \times 6}$, associated confidence scores $\boldsymbol{\pi}_0 \in \mathbb{R}^{N_R N_L}$, and predictions for the agents’ movements $\mathbf{P}_{1:N_A} \in \mathbb{R}^{N_A \times T_F \times 2}$. Given that the total trajectory count $N_R N_L$ for $\mathbf{T}_0$ can be extensive, an initial filtering step retains only the top K trajectories, ranked by their confidence scores, to streamline subsequent computations.

Following [2], a closed-loop forward simulation is performed on $\mathbf{T}_0$ to obtain simulated rollouts $\tilde{\mathbf{T}}_0$, utilizing a linear quadratic regulator (LQR) for trajectory tracking and a kinematic bicycle model for state updates. It has been noted [3] that trajectory-based imitation learning may not fully account for the dynamics of the underlying system, potentially leading to discrepancies between the model’s planned trajectory and its actual execution. To mitigate this issue, our assessment relies on the simulated rollouts rather than the model’s direct output, thus narrowing the gap.

Subsequently, a rule-based evaluator assigns scores π_{rule} to each simulated rollout based on criteria such as progress,

driving comfort, and adherence to traffic regulations, in alignment with the framework established in [2]. This evaluation also incorporates predictions of agents’ trajectories $\mathbf{P}_{1:N_A}$ to calculate the time-to-collision (TTC) metric, excluding rollouts that result in at-fault collisions. The ultimate score combines the initial learning-based confidence score π_0 with the rule-based score π_{rule} via the equation:

$$\boldsymbol{\pi} = \pi_{rule} + \alpha\pi_0, \quad (15)$$

where α represents a fixed weighting factor. The selection of the final trajectory τ^* is based on maximizing $\boldsymbol{\pi}$. Unlike the post-processing step described in [30], [31], which typically utilizes an optimizer to refine the trajectory, our post-processing module acts solely as a trajectory selector, leaving the original planning trajectory unaltered. We regard the post-processing step as a proxy to inject human preference or control into the black-boxed neural network, acknowledging its current limitations, and providing a lower-bound safety assurance to mitigate the risk of catastrophic accidents.

IV. EXPERIMENTS

A. Experiment Setup

nuPlan. Our model was trained and evaluated using the nuPlan dataset [14]. This dataset comprises 1,300 hours of real-world driving data, encompassing up to 75 labeled scenario types. It introduces the first publicly accessible, large-scale planning benchmark for autonomous driving through its associated closed-loop simulation framework. Each simulation conducts a 15-second rollout at a frequency of 10 Hz, during which the autonomous vehicle is managed by a planner and tracker. Traffic agents within these simulations are controlled in two distinct manners: *non-reactive*, wherein agents’ states are determined based on logged trajectories, and *reactive*, wherein agents are governed by an Intelligent Driver Model [51] planner.

Benchmark and Metrics. For all experiments, we use a standardized training split of 1M frames sampled from all scenario types. For evaluation, we use the **Val14** benchmark [2], which contains up to 100 scenarios from the 14 scenario types specified in the nuPlan planning challenge, resulting in a total number of 1090 scenarios (we filter out a few scenarios that initialized as failed in the reactive simulations).

NuPlan employs three principal evaluation metrics: the open-loop score (OLS), the non-reactive closed-loop score (NR-score), and the reactive closed-loop score (R-score). Given that previous studies have demonstrated a minimal correlation between open-loop prediction performance and closed-loop planning effectiveness, we only focus on closed-loop performance in this paper. The closed-loop score is calculated as a weighted average of several key metrics:

- (1) *No ego at-fault collisions*: A collision is identified when the autonomous vehicle’s (AV) bounding box intersects with that of other agents or static obstacles. However, collisions initiated by other agents, such as rear-end collisions, are disregarded.

- (2) *Time to collision (TTC) within bound*: The TTC is defined as the time it would take for the AV and another entity to collide if they continue on their current trajectories and speeds. This metric mandates that the TTC exceeds a specified threshold.
- (3) *Drivable area compliance*: This criterion requires that the AV remains within the boundaries of the drivable roadway at all times, ensuring adherence to the designated driving area.
- (4) *Comfortableness*: The comfort of the AV is quantified by examining the minimum and maximum values of its longitudinal and lateral accelerations and jerks, as well as its yaw rate and acceleration. These parameters are assessed against established thresholds derived from empirical data.
- (5) *Progress*: Progress is evaluated by comparing the distance covered by the AV along its planned route to that achieved by an expert driver, expressed as a percentage.
- (6) *Speed limit compliance*: This metric checks whether the AV's speed falls within the legal limits prescribed for the roadway it is traversing.
- (7) *Driving direction compliance*: This measure penalizes deviations from the correct driving direction, particularly incidents where the AV is found traveling against the flow of traffic.

A more detailed description and calculation of the metrics can be found at [52]. We use the non-reactive closed-loop score (denoted as **score** if not specified) as our primary overall performance evaluation metric.

Baselines. In this study, we conduct a comparative analysis between PLUTO and both existing and state-of-the-art (SOTA) methodologies utilizing the nuPlan benchmark to demonstrate the efficacy of our proposed method. The baselines for comparison are categorized into three groups: rule-based, pure learning, and hybrid approaches. Rule-based methods rely on manually engineered rules without incorporating learning processes. In contrast, pure learning methods employ neural networks to directly generate the final planned trajectory, omitting any refinement or post-processing stages. Hybrid methods, however, include a post-processing module to refine or adjust outcomes derived from learning-based techniques. The benchmarked methods are outlined as follows:

- (1) **Intelligent Driver Model (IDM)** [51]: This is a classic, time-continuous car-following model extensively utilized in traffic simulations. We employ the official implementation as referenced in the literature [14].
- (2) **PDM-Closed** [2]: Identified as the winning entry in the 2023 nuPlan planning challenge, this method generates a series of proposals by integrating IDM policies with varying hyperparameters, subsequently selecting the optimal one through a rule-based scoring system. Despite its simplicity, it has proven effective in practice and currently holds the SOTA performance. Its open-source implementation is utilized in our study.
- (3) **PDM-Open** [2]: This approach, centered around a predictive model that conditions on the centerline and utilizes MLPs, is implemented through an available open-source version.

- (4) **GC-PGP** [53]: A predictive model that focuses on goal-conditioned lane graph traversals.
- (5) **RasterModel**: A CNN-based model that interprets the input scenario as a multi-channel image, as described in referenced literature [14].
- (6) **UrbanDriver** [28]: A learning-based planner that leverages vectorized inputs through PointNet-based polyline encoders and Transformers. This model is assessed through its open-loop re-implementation, incorporating historical data perturbation during its training phase.
- (7) **PlanTF** [3]: A strong pure imitation learning baseline that leverages a Transformer architecture to explore efficient design in imitation learning. Despite its simplicity, it stands as the current SOTA among pure learning models.
- (8) **GameFormer** [31]: Modeled on DETR-like interactive planning and prediction based on level-k games, the output from this model serves as an initial estimate, which is further refined through a nonlinear optimizer. The official open-source code is used for implementation purposes.
- (9) **PlanTF-H**: This method enhances PlanTF by integrating a post-processing module as described in Sect. III-F, thereby converting it into a hybrid approach.

B. Implementation Details

We present two variations PLUTO[†] and PLUTO, differing only in that PLUTO[†] omits the post-processing step. Feature extraction focuses on map elements and agents within a 120-meter radius of the autonomous vehicle. We adhere to the nuPlan challenge by setting the planning and historical data horizons at 8 seconds and 2 seconds, respectively. The model incorporates auxiliary tasks designed to penalize off-road driving and collisions. Training was conducted using 4 RTX3090 GPUs, with a batch size of 128 over 25 epochs. We utilized the AdamW optimizer, applying a weight decay of $1e^{-4}$. The learning rate is linearly increased to $1e^{-3}$ over the first three epochs and then follows a cosine decay pattern throughout the remaining epochs. The loss weights w_{1-4} are uniformly assigned a value of 1.0. The training finishes in 45 hours with CIL and 22 hours without it. Details on further parameter settings can be found in Table I.

TABLE I
PARAMETERS USED IN PLUTO[†] AND PLUTO

Notation	Parameters	Values
T_H	Historical timesteps	20
T_F	Future timesteps	80
D	Hidden dimension	128
L_{enc}	Num. encoder layers	4
L_{dec}	Num. decoder layers	4
N_L	Num. lon. queries	12
N_c	Num. covering circles	3
$[H, W]$	Cost map size	500×500
-	Cost map resolution	0.2m
α	Score weight	0.3
σ	Temperature parameter	0.1

TABLE II
CLOSED-LOOP PLANNING RESULTS ON THE VAL14 BENCHMARK. ALL METRICS ARE HIGHER THE BETTER.

Type	Planner	Score	Collisions	TTC	Drivable	Comfort	Progress	Speed	R-score
Expert	Log-Replay	93.68	98.76	94.40	98.07	99.27	98.99	96.47	81.24
Rule-based	IDM [51]	79.31	90.92	83.49	94.04	94.40	86.16	97.33	79.31
	PDM-Closed [2]	93.08	98.07	93.30	99.82	95.52	92.13	99.83	93.20
Pure Learning	PDM-Open [2]	50.24	74.54	69.08	87.89	99.54	69.86	97.72	54.86
	GC-PGP [53]	61.09	85.87	80.18	89.72	90.00	60.32	99.34	54.91
	RasterModel [14]	66.92	86.97	81.46	85.04	81.46	80.60	98.03	64.66
	UrbanDriver [28]	67.72	85.60	80.28	90.83	100.0	80.83	91.58	64.87
	PlanTF [3]	85.30	94.13	90.73	96.79	93.67	89.83	97.78	77.07
	PLUTO [†] (w/o post.)	89.04	96.18	93.28	98.53	96.41	89.56	98.13	80.01
Hybrid	GameFormer [31]	82.95	94.32	86.77	94.87	93.39	89.04	98.67	83.88
	PlanTF-H [3]	89.96	97.06	93.38	97.79	91.08	92.90	98.01	88.08
	PLUTO (Ours)	93.21	98.30	94.04	99.72	91.93	93.65	98.20	92.06

V. RESULTS AND DISSCUSION

A. Comparison with State of the Art

The comparative analysis with other methods on the Val14 benchmark is detailed in Table II. Initially, our purely learning-oriented variant, PLUTO[†], surpasses all prior baselines dedicated to pure learning. Significantly, when compared to the leading model, PlanTF, PLUTO[†] demonstrates marked improvements across nearly all evaluated metrics, with particular enhancements observed in metrics pertinent to safety (*e.g.*, *Collisions* improved from 94.13 to 96.18, *TTC* from 90.73 to 93.28, and *Drivable* from 96.79 to 98.53). These results highlight the constraints of models based solely on imitation and underscore the effectiveness of incorporating auxiliary loss and the design of the CIL framework.

Furthermore, our hybrid model, PLUTO, attains the highest scores across all baselines, surpassing the current state-of-the-art rule-based model, PDM-Closed, for the first time. This achievement emphasizes the promise of learning-based approaches in planning. Remarkably, the performance of our methods closely aligns with that of the log-reply expert (scoring 93.68 vs. 93.21), indicating a significant stride towards expert-level planning.

In addition to quantitative outcomes, our method also presents an advantage in driving behavior over PDM-Closed. Given that PDM-Closed primarily focuses on speed planning, its capability in lateral maneuvers is somewhat restricted, limiting its ability to execute lane-change actions. In contrast, PLUTO is designed to consider both longitudinal and lateral movements, thanks to the query-based architecture of our model. This capability will be further illustrated through case studies in the section on qualitative results.

B. Qualitative Results

Fig. 5 presents selected scenarios from the nuPlan test set, showcasing the robust performance of our framework in complex, interactive urban driving situations through the exhibition of diverse, human-like behaviors. The scenarios are detailed as follows:

- The autonomous vehicle navigates to an adjacent empty lane to enhance efficiency and subsequently halts at a red light at the intersection. Observations from sequences a-2 and a-3 reveal that PLUTO concurrently evaluates multiple potential plans for different behaviors (illustrated by gray candidate trajectories), enhancing the planning process's flexibility and resemblance to human driving.
- In a scenario requiring the AV to maneuver around a roundabout, its path is narrowed by a parked vehicle. Our planning system adeptly navigates around this obstacle while yielding to an oncoming vehicle in a constrained space, exemplifying our method's competence in managing static obstacles and interacting with other vehicles.
- Encountering a stationary vehicle within its lane, the AV executes a left lane change to bypass the obstacle, subsequently returning to its original lane to adhere to the intended route. This scenario underscores the planner's dynamic decision-making capabilities and its adeptness in route adherence and road topology comprehension.
- During a left-turn maneuver in heavy traffic, the AV patiently waits for an opportune moment to execute the turn, illustrating our method's effectiveness in navigating intersections in high-density traffic conditions.
- Upon following a slower vehicle, the AV opts to accelerate and overtake, showcasing a behavior that aligns closely with natural human driving and highlighting our proposed method's adaptability.

In summary, PLUTO exhibits advanced and varied driving behaviors unattainable through simplistic speed-planning methods (*e.g.*, PDM-Closed). Its ability to execute natural lane changes, navigate around obstacles, and dynamically modify decisions in interactive scenarios marks a significant advancement towards the realization of practical learning-based planning. For further insights, including videos, we direct interested readers to our project website.

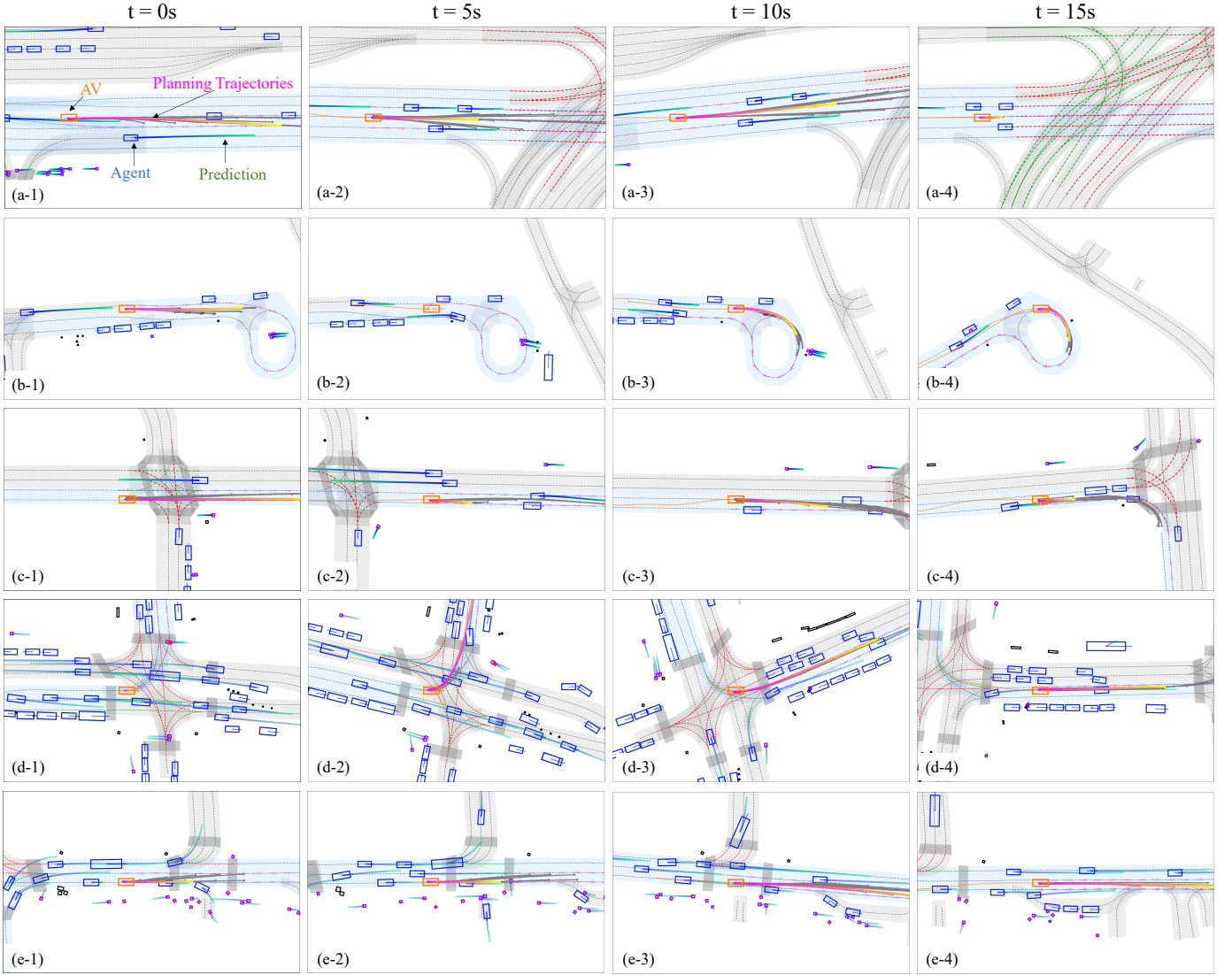


Fig. 5. Qualitative results of closed-loop planning for five representative scenarios from the test set. Each scenario (every row) lasts 15 seconds and we take 4 snapshots with a 5-second interval. Purple dash arrows denote the reference lines, the light blue lanes denote the global routing plan and other important legends are marked in Fig. a-1. It is recommended to refer to our [project website](#) for more vivid videos.

C. Ablation Studies

For all ablation studies, we evaluate on a subset of nuPlan (non-overlapping with the **Val14** benchmark), which contains 20 scenarios for each of the 14 scenarios types.

Influence of Each Component. Table II shows the trajectory from a base model to the top-performing learning-based planner. The initial model, denoted as $\mathcal{M}_0$, is constructed on the architecture depicted in Sect. III-B and III-C, employing solely imitation loss for training. $\mathcal{M}_0$ demonstrates performance on par with the previous SOTA pure learning-based method, PlanTF, an achievement we ascribe to the enhanced query-based architecture.

The introduction of the state dropout encoder (SDE) in $\mathcal{M}_1$, which randomly masks the autonomous vehicle’s kinematic states during training to avert the generation of shortcut trajectories by state extrapolation, results in marked improvements over $\mathcal{M}_0$ across almost all metrics.

$\mathcal{M}_2$ incorporates an auxiliary loss designed to penalize deviations from the drivable area and collisions. This modification leads to enhancements in both the *Collision* metric (from 97.37 to 97.98) and the *Drivable* metric (from 95.92 to 98.38). We would like to highlight that despite the seemingly minor difference in total scores between $\mathcal{M}_2$ and $\mathcal{M}_1$, the disparity in their actual performance is substantial. We direct interested readers to the project website for the model ablation results. The $\mathcal{M}_3$ model underscores the importance of integrating a reference line-free decoding head to effectively handle scenarios where reference lines are absent, such as in parking lots.

Further, $\mathcal{M}_4$ is trained using the proposed contrastive imitation learning framework, achieving a significant uplift in performance from 90.69 to 91.66. This improvement is noteworthy, particularly as it is already approaching the expert’s performance.

Ultimately, $\mathcal{M}_5$ attains the best overall performance, though

TABLE III
ABLATION STUDY OF THE INFLUENCE OF EACH COMPONENT

Model	Description	Score	Collisions	TTC	Drivable	Comfort	Progress	Speed	Direction
–	PlanTF [3]	87.55	96.39	90.76	96.79	93.57	90.50	96.75	99.40
$\mathcal{M}_0$	Base	87.04	95.92	91.43	95.92	95.92	90.90	91.01	100.0
$\mathcal{M}_1$	$\mathcal{M}_0$ + SDE	89.64	97.37	95.14	97.57	96.36	90.16	96.91	100.0
$\mathcal{M}_2$	$\mathcal{M}_1$ + Auxiliary loss	90.03	97.98	93.52	98.38	96.76	90.04	97.21	99.39
$\mathcal{M}_3$	$\mathcal{M}_2$ + Ref. free head	90.69	97.57	94.74	99.19	94.33	89.86	97.13	99.60
$\mathcal{M}_4$	$\mathcal{M}_3$ + CIL	91.66	97.59	94.38	99.60	96.39	91.30	96.58	99.60
$\mathcal{M}_5$	$\mathcal{M}_4$ + Post-processing	93.57	98.39	95.58	99.60	93.17	93.32	97.08	99.80
–	Expert (Log-Replay)	94.24	98.59	95.58	98.39	99.60	99.06	94.53	99.80

TABLE IV
IMPACT OF DIFFERENT LONGITUDINAL QUERIES N_L (BASED ON $\mathcal{M}_4$)

N_L	Score	Collisions	TTC	Comfort	Progress
6	88.89	96.56	93.12	96.36	89.47
12	91.66	97.59	94.38	96.39	91.30
18	90.18	97.17	95.14	95.95	90.88
24	87.90	95.58	93.57	95.58	89.14

TABLE V
IMPACT OF DIFFERENT K IN POST-PROCESSING (BASED ON $\mathcal{M}_5$)

K	Score	Collisions	TTC	Comfort	Progress
10	93.18	98.19	95.18	93.57	92.73
20	93.57	98.39	95.58	93.17	93.32
30	93.58	98.39	95.98	91.57	93.66
40	93.02	98.39	94.38	90.76	93.79

at a minor trade-off in the *Comfort* metric. This compromise stems from the increased incidence of emergency stops triggered by the safety checker, thereby enhancing safety-related metrics.

Number of Longitudinal Queries. Table IV presents the outcomes associated with various quantities of longitudinal queries N_L (utilizing model $\mathcal{M}_4$). The results indicate that a setting of $N_L = 12$ yields the most favorable performance among four tested variants. This suggests an appropriate number of queries is necessary to cover all the longitudinal behaviors for a planning horizon of 8s. An increase in N_L beyond this point detracts from performance. This decline can likely be attributed to the sufficiency of $N_L = 12$ in capturing a diverse array of behaviors; additional queries become redundant, potentially increasing the training difficulty.

Top-K in Coarse Selection. In the planning cycle, PLUTO generates a total of $N_R \times N_L$ trajectories. Empirical evidence suggests that trajectories associated with low confidence scores often exhibit inferior quality. Consequently, employing a preliminary selection process based on confidence scores proves advantageous in eliminating such trajectories, thereby expediting subsequent post-processing. As illustrated in Table V, setting $K = 20$ turns out to be appropriate.

TABLE VI
IMPACT OF α IN TRAJECTORY SELECTION (BASED ON $\mathcal{M}_5$)

α	Score	Collisions	TTC	Comfort	Progress
Rule	90.64	96.79	91.77	80.32	98.43
0.1	92.27	97.99	92.71	85.14	96.48
0.3	93.57	98.39	95.58	93.17	93.32
0.5	93.50	98.79	95.98	97.19	90.73
0.7	93.12	98.39	95.98	96.38	90.60
0.9	93.26	98.39	96.39	98.38	90.74
$\mathcal{M}_4$	91.66	97.59	94.38	96.39	91.30

TABLE VII
CONSTANT VELOCITY VS. LEARNED PREDICTION (BASED ON $\mathcal{M}_5$)

Prediction	Score	Collisions	TTC	Comfort	Progress
Const. vel.	92.82	97.79	95.19	90.76	93.18
Learned	93.57	98.39	95.58	93.17	93.32

Weight of the Learning-based Score. As demonstrated in Equation 15, the final score is derived from the combined weighted contributions of the rule-based and the learning-based scores. This study examines the impact of the weight parameter α , with the findings detailed in Table VI. Firstly, it is observed that incorporating the learning-based score significantly enhances performance compared to relying solely on the rule-based score (*i.e.*, $\alpha = 0$). The limitation of the rule-based score lies in its hand-crafted nature, which may not accurately represent all possible scenarios, whereas the learning-based score offers greater generalizability by dynamically adapting to the input features. Furthermore, a combined approach proves superior to using a purely learning-based score ($\mathcal{M}_4$), indicating that the current model, while advanced, still benefits from the inclusion of a rule-based component as a form of safety assurance. Based on optimal performance, $\alpha = 0.3$ has been selected as the default setting.

Prediction method. In this study, we contrast the performance of our learned prediction model against the constant velocity prediction employed in PDM-Closed and presented in Table

VII. It is evident that employing learned predictions for planning yields superior results compared to the simplistic constant velocity prediction, as it can more accurately discern the behaviors of the agents. Despite our prediction model producing only a singular modal trajectory for each agent, it works well in practice.

VI. CONCLUSION

In this study, we introduce PLUTO, a pioneering data-driven planning framework that extends the capabilities of imitation learning within the autonomous driving domain. We propose innovative solutions concerning model architecture, data augmentation, and the learning framework, effectively addressing enduring challenges in imitation learning. The query-based model architecture furnishes the planner with the capacity for adaptable driving behaviors across both longitudinal and lateral dimensions. Our novel method for computing auxiliary loss, based on differentiable interpolation, offers a new approach for integrating constraints into the model. Additionally, the employment of a contrastive imitation learning framework, coupled with an advanced set of data augmentation techniques, enhances the acquisition of desired behaviors and comprehension of intrinsic interactions. Experimental evaluations utilizing real-world driving datasets demonstrate that our approach sets a new benchmark for closed-loop performance in the field. Notably, PLUTO surpasses the previously best-performing rule-based planner, establishing a significant breakthrough in autonomous driving research.

Limitations and Future Work. In our approach, we predict a single trajectory for each dynamic agent. This methodology yields satisfactory outcomes in practical applications; nevertheless, the generation of meaningful joint multimodal predictions and their efficient incorporation into planning strategies represent significant areas for future research. The addition of a post-processing module has been demonstrated to improve overall performance effectively. However, it cannot handle scenarios where all generated trajectories are unusable. Transitioning the post-processing function to an intermediary role that directly influences trajectory generation could present a more advantageous strategy.

REFERENCES

- [1] S. Teng, X. Hu, P. Deng, B. Li, Y. Li, Y. Ai, D. Yang, L. Li, Z. Xuanyuan, F. Zhu *et al.*, “Motion planning for autonomous driving: The state of the art and future perspectives,” *IEEE Transactions on Intelligent Vehicles*, 2023.
- [2] D. Dauner, M. Hallgarten, A. Geiger, and K. Chitta, “Parting with misconceptions about learning-based vehicle motion planning,” in *Conference on Robot Learning (CoRL)*, 2023.
- [3] J. Cheng, Y. Chen, X. Mei, B. Yang, B. Li, and M. Liu, “Rethinking imitation-based planner for autonomous driving,” in *International Conference on Robotics and Automation (ICRA)*, 2024.
- [4] J. Cheng, R. Xin, S. Wang, and M. Liu, “Mnpn: Multi-policy neural planner for urban driving,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 549–10 554.
- [5] M. Bansal, A. Krizhevsky, and A. Ogale, “Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst,” *arXiv preprint arXiv:1812.03079*, 2018.
- [6] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. Cun, “Off-road obstacle avoidance through end-to-end learning,” *Advances in neural information processing systems*, vol. 18, 2005.
- [7] D. Wang, C. Devin, Q.-Z. Cai, P. Krähenbühl, and T. Darrell, “Monocular plan view networks for autonomous driving,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2876–2883.
- [8] C. Wen, J. Lin, T. Darrell, D. Jayaraman, and Y. Gao, “Fighting copycat agents in behavioral cloning from observation histories,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 2564–2575, 2020.
- [9] J. Zhou, R. Wang, X. Liu, Y. Jiang, S. Jiang, J. Tao, J. Miao, and S. Song, “Exploring imitation learning for autonomous driving with feedback synthesizer and differentiable rasterization,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1450–1457.
- [10] P. De Haan, D. Jayaraman, and S. Levine, “Causal confusion in imitation learning,” *Advances in neural information processing systems*, vol. 32, 2019.
- [11] L. Cultrera, F. Becattini, L. Seidenari, P. Pala, and A. Del Bimbo, “Addressing limitations of state-aware imitation learning for autonomous driving,” *IEEE Transactions on Intelligent Vehicles*, 2023.
- [12] Y. Lu, J. Fu, G. Tucker, X. Pan, E. Bronstein, B. Roelofs, B. Sapp, B. White, A. Faust, S. Whiteson *et al.*, “Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios,” *NeurIPS 2022 Workshop on Machine Learning for Autonomous Driving*, 2022.
- [13] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.
- [14] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari, “nuPlan: A closed-loop ml-based planning benchmark for autonomous vehicles,” *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021.
- [15] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, “End-to-end autonomous driving: Challenges and frontiers,” *arXiv preprint arXiv:2306.16927*, 2023.
- [16] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, “Learning by cheating,” in *Conference on Robot Learning*. PMLR, 2020, pp. 66–75.
- [17] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, “Exploring the limitations of behavior cloning for autonomous driving,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9329–9338.
- [18] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, “Transfuser: Imitation with transformer-based sensor fusion for autonomous driving,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.
- [19] K. Chitta, A. Prakash, and A. Geiger, “Neat: Neural attention fields for end-to-end autonomous driving,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 793–15 803.
- [20] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu, “Safety-enhanced autonomous driving using interpretable sensor fusion transformer,” in *Conference on Robot Learning*. PMLR, 2023, pp. 726–737.
- [21] X. Jia, P. Wu, L. Chen, J. Xie, C. He, J. Yan, and H. Li, “Think twice before driving: Towards scalable decoders for end-to-end autonomous driving,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 983–21 994.
- [22] D. Chen and P. Krähenbühl, “Learning from all vehicles,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 222–17 231.
- [23] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang *et al.*, “Planning-oriented autonomous driving,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 853–17 862.
- [24] B. Jiang, S. Chen, Q. Xu, B. Liao, J. Chen, H. Zhou, Q. Zhang, W. Liu, C. Huang, and X. Wang, “Vad: Vectorized scene representation for efficient autonomous driving,” *arXiv preprint arXiv:2303.12077*, 2023.
- [25] S. Hu, L. Chen, P. Wu, H. Li, J. Yan, and D. Tao, “St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning,” in *European Conference on Computer Vision*. Springer, 2022, pp. 533–549.
- [26] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “Carla: An open urban driving simulator,” in *Conference on robot learning*. PMLR, 2017, pp. 1–16.
- [27] M. Vitelli, Y. Chang, Y. Ye, A. Ferreira, M. Wolczyk, B. Osinski, M. Niendorf, H. Grimmer, Q. Huang, A. Jain *et al.*, “Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 897–904.
- [28] O. Scheel, L. Bergamini, M. Wolczyk, B. Osinski, and P. Ondruska, “Urban driver: Learning to drive from real-world demonstrations using

- policy gradients,” in *Conference on Robot Learning*. PMLR, 2022, pp. 718–728.
- [29] S. Pini, C. S. Perone, A. Ahuja, A. S. R. Ferreira, M. Niendorf, and S. Zagoruyko, “Safe real-world autonomous driving by learning to predict and plan with a mixture of experts,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 069–10 075.
- [30] Z. Huang, H. Liu, J. Wu, and C. Lv, “Differentiable integrated motion prediction and planning with learnable cost function for autonomous driving,” *IEEE transactions on neural networks and learning systems*, 2023.
- [31] Z. Huang, H. Liu, and C. Lv, “Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3903–3913.
- [32] S. Ross, G. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.
- [33] R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping,” in *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR’06)*, vol. 2. IEEE, 2006, pp. 1735–1742.
- [34] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.
- [35] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, “Learning transferable visual models from natural language supervision,” in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.
- [36] Y. Liu, Q. Yan, and A. Alahi, “Social nce: Contrastive learning of socially-aware motion representations,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 118–15 129.
- [37] M. Halawa, O. Hellwich, and P. Bideau, “Action-based contrastive learning for trajectory prediction,” in *European Conference on Computer Vision*. Springer, 2022, pp. 143–159.
- [38] Y. Wang, P. Zhang, L. Bai, and J. Xue, “Fend: A future enhanced distribution-aware contrastive learning framework for long-tail trajectory prediction,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 1400–1409.
- [39] J. Cheng, X. Mei, and M. Liu, “Forecast-MAE: Self-supervised pre-training for motion forecasting with masked autoencoders,” *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.
- [40] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.
- [41] Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang, “Query-centric trajectory prediction,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 863–17 873.
- [42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [43] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *European conference on computer vision*. Springer, 2020, pp. 213–229.
- [44] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, “Multimodal motion prediction with stacked transformers,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7577–7586.
- [45] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal *et al.*, “Scene transformer: A unified architecture for predicting multiple agent trajectories,” *arXiv preprint arXiv:2106.08417*, 2021.
- [46] R. J. Williams and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks,” *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.
- [47] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
- [48] S. Shi, L. Jiang, D. Dai, and B. Schiele, “Motion transformer with global intention localization and local movement refinement,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 6531–6543, 2022.
- [49] J. Cheng, Y. Chen, Q. Zhang, L. Gan, C. Liu, and M. Liu, “Real-time trajectory planning for autonomous driving with gaussian process and incremental refinement,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8999–9005.
- [50] J. Goldberger, G. E. Hinton, S. Roweis, and R. R. Salakhutdinov, “Neighbourhood components analysis,” *Advances in neural information processing systems*, vol. 17, 2004.
- [51] M. Treiber, A. Hennecke, and D. Helbing, “Congested traffic states in empirical observations and microscopic simulations,” *Physical review E*, vol. 62, no. 2, p. 1805, 2000.
- [52] Motional. nuplan metrics. [Online]. Available: https://nuplan-devkit.readthedocs.io/en/latest/metrics_description.html
- [53] M. Hallgarten, M. Stoll, and A. Zell, “From prediction to planning with goal conditioned lane graph traversals,” *arXiv preprint arXiv:2302.07753*, 2023.



Jie Cheng received the B.S. degree from Huazhong University of Science and Technology, Wuhan, China, in 2019. He is currently pursuing the Ph.D. degree in the Department of Electronic and Computer Engineering, the Hong Kong University of Science and Technology, HKSAR, China, supervised by Prof. Qifeng Chen. His research mainly focuses on motion planning and motion forecasting for autonomous driving.



Yingbing Chen (Student Member, IEEE) received the B.Sc. degree from Northwestern Polytechnical University, Xian, China, in 2015, and the M.S. degree from Xiamen University, Xiamen, China, in 2018. He is currently working toward the Ph.D. degree with the Robotics and MultiPerception Laboratory, Hong Kong University of Science and Technology, Hong Kong. His current research interests include machine learning, motion and behavioral planning for autonomous driving and robotics.



Qifeng Chen is an assistant professor of the Department of Computer Science and Engineering and the Department of Electronic and Computer Engineering at The Hong Kong University of Science and Technology. He received his Ph.D. in computer science from Stanford University in 2017. His research interests include image processing and synthesis and 3D vision. He won the MIT Tech Review's 35 Innovators under 35 in China and the Google Faculty Research Award in 2018.