Differentiable Integrated Motion Prediction and Planning With Learnable Cost Function for Autonomous Driving

Zhiyu Huang, *Graduate Student Member, IEEE*, Haochen Liu, *Graduate Student Member, IEEE*, Jingda Wu, *Graduate Student Member, IEEE*, and Chen Lv, *Senior Member, IEEE*

Abstract— Predicting the future states of surrounding traffic participants and planning a safe, smooth, and socially compliant trajectory accordingly are crucial for autonomous vehicles (AVs). There are two major issues with the current autonomous driving system: the prediction module is often separated from the planning module, and the cost function for planning is hard to specify and tune. To tackle these issues, we propose a differentiable integrated prediction and planning (DIPP) framework that can also learn the cost function from data. Specifically, our framework uses a differentiable nonlinear optimizer as the motion planner, which takes as input the predicted trajectories of surrounding agents given by the neural network and optimizes the trajectory for the AV, enabling all operations to be differentiable, including the cost function weights. The proposed framework is trained on a large-scale real-world driving dataset to imitate human driving trajectories in the entire driving scene and validated in both open-loop and closed-loop manners. The open-loop testing results reveal that the proposed method outperforms the baseline methods across a variety of metrics and delivers planning-centric prediction results, allowing the planning module to output trajectories close to those of human drivers. In closed-loop testing, the proposed method outperforms various baseline methods, showing the ability to handle complex urban driving scenarios and robustness against the distributional shift. Importantly, we find that joint training of planning and prediction modules achieves better performance than planning with a separate trained prediction module in both open-loop and closed-loop tests. Moreover, the ablation study indicates that the learnable components in the framework are essential to ensure planning stability and performance. Code and Supplementary Videos are available at https://mczhi.github.io/DIPP/.

Index Terms— Autonomous driving, differentiable motion planning, multiagent interactive prediction.
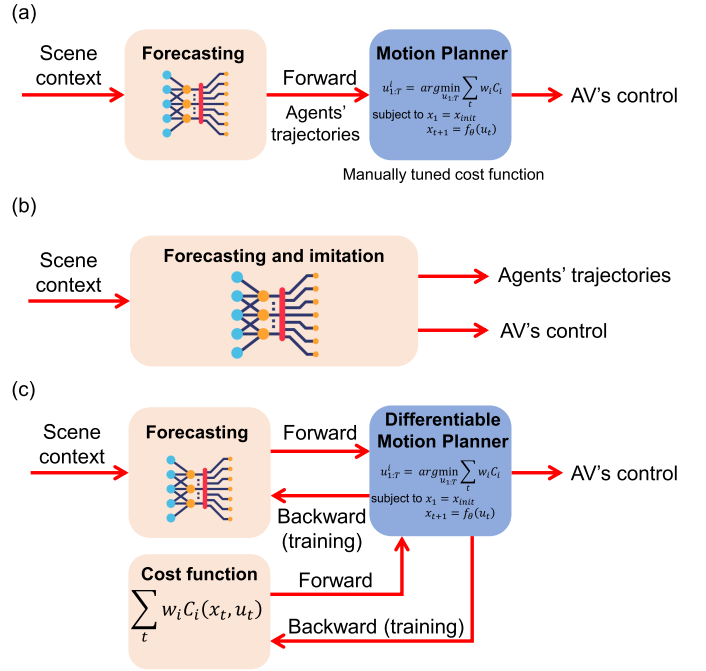

Fig. 1. Three different motion planning paradigms. (a) Traditional sequential prediction and planning. (b) End-to-end method. (c) Our proposed method.

I. INTRODUCTION

MAKING safe, socially compatible, and human-like decisions is a fundamental capability of autonomous vehi-

cles (AVs). While learning-based end-to-end decision-making methods, such as imitation learning (IL) and reinforcement learning [1], have enjoyed vigorous growth recently, they still lack interpretability, robustness, and safety compared with classic planning-based methods [2], [3], [4]. Nonetheless, to make safe and smooth plans in complex traffic scenarios, the key is to accurately forecast the future trajectories of surrounding traffic participants [5], [6], [7], [8]. The current autonomous driving stack treats prediction and planning as separate and sequential parts [see Fig. 1(a)], which means the prediction module is independent of the planning module. The problem with this setting is that prediction and planning tasks are highly interrelated and interdependent tasks, especially when the AV interacts with other traffic participants in urban areas. Moreover, another challenge with the traditional method is how to specify the cost function that can properly evaluate future motion plans and achieve a delicate trade-off between different requirements, e.g., collision avoidance, travel efficiency, and ride comfort. Manually tuning the parameters of the cost function is laborious and time-consuming and might

only be applicable in certain scenarios. Although some methods have been proposed to learn the cost function from driving data, such as continuous inverse optimal control (CIOC) [9], [10], sampling-based maximum-entropy inverse reinforcement learning (IRL) [11], as well as max-margin method [12], they are not coupled with the prediction module and rely on the impractical assumption that the prediction results are perfect. Some other methods have turned to a pure data-driven setting [13], which utilizes a holistic model to directly output planned AV trajectory while predicting other agents' future trajectories [see Fig. 1(b)], implicitly handling the interactions between agents. However, such methods lack safety guarantees and reliability for the decision-making task in safety-critical applications.

In this article, we propose a novel framework called differentiable integrated prediction and planning (DIPP), as shown in Fig. 1(c). The objective of the DIPP framework is to make the prediction module aware of the downstream planning task, so as to deliver planning-aware prediction results to the motion planner, without altering the established structure of the prediction–planning process. The principle of the DIPP framework is to implicitly change the prediction of other agents' future behaviors in the training process, where the planning error of the AV could also influence the prediction results of other agents, enabling the planner to make better and human-like plans at test time. Specifically, we construct a Transformer-based neural network to forecast the future trajectories of surrounding agents simultaneously. Then, the prediction results are channeled into a differentiable motion planner (optimizer), which explicitly plans a trajectory for the AV. When training the framework, the AV's planned trajectory is compared against the human driving one, and the planning loss can be backpropagated to the prediction module and cost function, to alter the prediction results and automatically adjust the cost function with the objective of making human-like plans. Here, we assume the perception results (e.g., detecting and tracking of moving objects [14], localization, and mapping) are accurate and focus on the joint prediction and planning problem. The proposed DIPP framework is trained to match the human driving trajectories (interactions among humans) in the entire driving scene and tested in both open-loop and closed-looped manners. In open-loop evaluations, we compared the planned trajectories of our framework directly against human trajectories without executing the plan, while in closed-loop evaluations, we unrolled the planned trajectory in simulation to test its performance in deployment scenarios. The results show that our framework is able to deliver more planning-centric prediction results, enabling the system better adapt to the scenarios in which they are deployed.

In summary, the main contributions of this article are listed as follows.

1) We propose a fully differentiable structured learning framework that integrates prediction and planning modules for autonomous driving, enabling the prediction results better fit to the downstream planning task and the cost function learnable from real-world driving data.

2) We train the proposed framework with a large-scale real-world driving dataset that covers a wide variety of complex urban driving scenarios and validate it in both open-loop and closed-loop manners.

3) We demonstrate that the proposed framework outperforms the baseline methods in both open-loop and closed-loop tests and conduct an ablation study to investigate the importance of each component.

II. RELATED WORK

A. Multiagent Prediction

There has been a growing body of learning-based approaches for trajectory prediction to excellent effect, because deep neural networks can handle complex environments with multiple interacting agents and various road structures. Some works [5], [15], [16], [17] utilizing vector representation of scene context and Transformer-based networks have advanced the forecasting accuracy even further. However, most of the existing models are formulated to predict each agent's future trajectory independently, which is computationally inefficient and might produce inconsistent results. Thus, some approaches focus on multiagent joint prediction to generate future trajectories for all agents of interest in a consistent manner [7], [18], [19], [20]. Importantly, using the multiagent prediction method can allow the model to capture the interactions between agents and facilitate the planning task. Therefore, in our proposed framework, we employ the multiagent prediction setting and provide each agent with a local vectorized map that shows the possible lanes and nearby crosswalks. We utilize Transformer modules to model the interactions between agents and their relations to different elements on the local map.

B. Motion Planning

Motion planning is a long-researched area, and there are a variety of approaches, such as trajectory optimization, graph search, random sampling, and, more recently, learning-based methods. Learning-based methods employ neural networks as driving policies that generate actions or trajectories directly from sensor inputs or perception results, such as deep IL [21], [22] and reinforcement learning [23], [24], [25]. Though simple and efficient, the neural network-based motion planner suffers from poor interpretability and generalization capability and also lacks stability and safety guarantees. Therefore, we turn to classic motion planning methods [2], and a popular choice is an optimization-based method, as it is flexible and achieves maximal control of the trajectory. In particular, we employ a differentiable least-squares nonlinear optimizer as the motion planner, so that the optimizer can be integrated into the neural framework and its parameters (e.g., cost function weights) can be learned at the same time. In contrast to other planning methods that often work in static environments or assume a perfect prediction of surrounding obstacles, our proposed approach integrates the prediction of the surrounding agents and jointly trains the predictor and planner to achieve better performance of robustness.

C. Joint Prediction and Planning

Driving in dense and interactive traffic requires joint reasoning of other agents' future behaviors and the AV's plans [13], [26], [27] to make active and human-like decisions. One

promising framework is to generate a set of candidate trajectories for the AV and predict other agents' future trajectories conditioned on the AV's planned trajectories [28], and the best planning trajectory could be selected after evaluating these candidate plans and corresponding prediction results [26], [29], [30]. For example, an interactive prediction and planning framework is proposed in [27], which can predict other agents' future states according to a planned state sequence of the ego vehicle. The planner in their approach uses a cross-entropy method to sample a large number of action sequences and has to iteratively query the prediction model to get other agents' responses, which may significantly slow down the planning process. In contrast, our method enables the prediction model to output planning-aware prediction results and implicitly captures the influence of the ego vehicle's future actions during the training process. This allows the planner to generate human-like plans in response to the prediction results during test time without requiring iterative queries to the prediction model.

Another promising framework is the end-to-end holistic neural network model that implicitly captures the prediction–planning interactions in the latent space, such as SceneTransformer [13], which jointly outputs a motion plan for the AV and predicted trajectories for other agents. Although end-to-end models enjoy enhanced accuracy and simple inference, they cannot explicitly compute the feedback loop between planning and prediction from open loop and offline training and, thus, cannot guarantee safety. Another challenge of such methods is the distribution shift encountered when the predicted actions are rolled out in deployment. Therefore, some other methods, e.g., mixtures of affine time-varying systems (MATS) [31], have tried to couple the prediction model with classic optimization methods because of their interpretability and reliability. MATS predicts the parameters of a linear-affine dynamical structure, which are utilized by a model predictive controller (MPC) for motion planning. Though efficient due to a small number of parameters, MATS does not incorporate the scene context in forecasting, and its performance is compromised. In this article, we propose to output the trajectories of surrounding agents using a neural network and combine it with differentiable optimization steps that explicitly consider ride comfort, safety, and traffic rules, making the feedback between planning and prediction differentiable and enhancing both safety and human driving similarity.

III. METHOD

A. Problem Formulation

We formulate the driving scene with the AV and a varying number of diverse interacting traffic participants as a continuous-space discrete-time system (time discretization is uniform), where the AV is denoted as A_0 and other agents are denoted as $A_1, \ldots, A_N$. Each agent, including the AV, has a semantic class (i.e., vehicle, bicycle, or pedestrian), and its state at time t is denoted as $\mathbf{s}_t^i$, where i is the agent index. We also introduce the scene context, such as a vectorized high-definition map and traffic light signals, to the system and denote it as $\mathbf{M}$. Assuming the current time step is $t = 0$, given the historical states of all agents for the previous H timesteps $\mathbf{X} = \mathbf{s}_{-H:0}^{0:N}$ and the scene context $\mathbf{M}$, the predictor needs to

predict K possible joint future trajectories of all agents for the next T timesteps $\{\mathbf{Y}_k | k = 1, \ldots, K\}$, $\mathbf{Y}_k = \hat{\mathbf{s}}_{1:T}^{0:N}$, where $\hat{\mathbf{s}}_t^i$ is the predicted state of agent i at future timestep t, and the probability of each future $\{p_k | k = 1, \ldots, K\}$. The planner needs to further optimize the AV's future trajectory according to the initial trajectory of the AV $\hat{\mathbf{s}}_{1:T}^0$, the prediction results of other agents, and the cost function.

Accordingly, the proposed framework consists of two parts, as shown in Fig. 2. First, we build up a neural network predictor to embed the historical trajectories of agents and scene context into high-dimensional spaces. The hidden embeddings are used to generate the keys, values, and queries ($\mathbf{K}$, $\mathbf{V}$, and $\mathbf{Q}$) used in the attention mechanism [32] of the interaction modeling modules (i.e., agent–map interaction and agent–agent interaction). All agents' different future trajectories and their probabilities are finally decoded from the latent representation of the interaction relationships. Second, we employ a differentiable optimizer as a motion planner to explicitly plan a future trajectory for the AV according to the most-likely prediction result and initial motion plan. Specifically, we formulate the motion planning problem as a nonlinear least squares problem, where $\mathbf{u}$ is the optimization variables (action sequence), $\mathbf{u}^i \subset \mathbf{u}$ is a subset of the action sequence, $\hat{\mathbf{s}}$ is the predicted states of other agents, c_i is the individual cost, and ω_i is the weight of the cost term. The optimization objective (sum of squared cost terms) covers a variety of factors in driving, including travel efficiency, collision avoidance, and ride comfort. We employ a differentiable nonlinear optimizer to solve the optimization, and thus, the gradient from the planner (planning loss) can be backpropagated to the prediction module and cost function in the training stage, making the whole framework end-to-end learnable. The details of the neural network predictor and differentiable motion planner are given in Sections III-B and III-C, respectively.

B. Multiagent Prediction and Initial Plan

1) Input Representation: The neural network predictor receives two types of input data: historical agent states and scene context. For each agent, its historical state is a sequence of dynamic features for the past timesteps $H = 20$, including its 2-D position, heading angle, velocity, and bounding box size. We consider the nearest $N = 10$ agents around the AV, whose observation data are stored in a fixed-size tensor with the missing agents padded as zeros. For the scene context, we consider two types of map elements, which are lanes and crosswalks. Each map element is presented by a vector, which consists of a sequence of waypoints with different associated features. For each agent, we build a local scene context, which encompasses the possible lanes it might take and surrounding crosswalks. The features of the waypoint in the drivable lane include the positions and headings of the lane center and lane boundaries, as well as the speed limit and traffic signals (e.g., traffic light state and stop sign), and the features of the waypoint in the crosswalk are the positions of points enclosing the crosswalk area. All the agents' and map elements' positional attributes are translated to the AV's local coordinate system.

2) Encoding Agent History and Scene Context: To encode an agent's observed historical states, we feed the state sequence into a long short-term memory (LSTM) network.
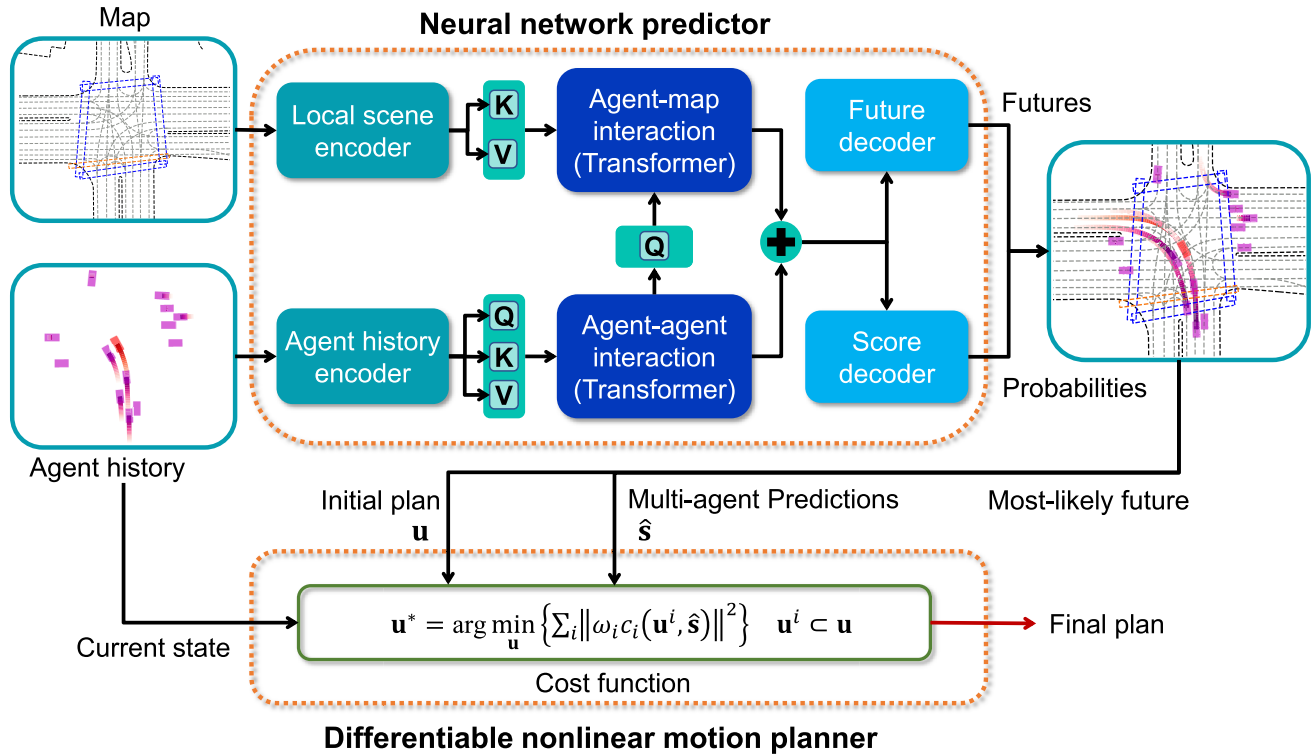
Fig. 2. Proposed DIPP framework. The neural network predictor is utilized to predict the future states of surrounding agents and the initial plan for the motion planner, and the differentiable motion planner with a learnable cost function is employed to explicitly plan an AV trajectory. All the components are connected and end-to-end differentiable.

LSTM is used, because we find it would bring better final prediction performance and computational efficiency than Transformers when dealing with short-term time series, and we can retrieve the encoded state feature at the last timestep from the LSTM as the node feature in the agent interaction graphs. Each type of agent shares an LSTM encoder, and all agents, including the AV, are stacked to form a tensor of agents' encoded historical states. The local scene context encoder consists of a lane encoder for processing the lanes and a crosswalk encoder for processing the crosswalk vectors. The lane encoder uses a multilayer perceptron (MLP) to encode numeric features (e.g., positions, directions, and speed limits) and embedding layers to encode discrete features (e.g., traffic light state, lane type, and stop sign), and the crosswalk encoder is another MLP to encode the numeric features (i.e., positions and headings). For each agent, we find its nearby six lanes and four crosswalks as map vectors, encode them separately, and concatenate the encoded map vectors to form a tensor of the agent's local scene context.

3) Modeling Agent–Agent and Agent–Scene Interactions: Capturing relationships or dependencies between agents and the environment is essential in ensuring prediction accuracy. To realize this, we follow the idea of [7] and construct two interaction graphs, which are the agent–agent interaction graph (all agents are fully connected in the graph) and the agent–map interaction graph (a target agent is connected to all local lanes and crosswalks polylines). We use a two-layer self-attention Transformer encoder as the agent–agent interaction encoder to process the graph, where the query, key, and value ($\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$) are the encoded agents' historical trajectory features. Here, the functionality of the Transformer is similar to the graph

attention network [7] but with improved structure and representation capability, and using a few layers of Transformer blocks would not significantly increase the computational cost. With each agent's encoded local scene context in hand, we employ two cross-attention Transformer encoders as the agent–map interaction encoder: one is for modeling the agent's attention on each lane vector (focusing on waypoints in the vector), and the other is for agent's attention on each crosswalk vector. We utilize an agent's interaction feature as query ($\mathbf{Q}$) and a single map vector (a sequence of encoded waypoints) as the key and value ($\mathbf{K}$ and $\mathbf{V}$). The operation is called multiple times to process all the map vectors from an agent, resulting in a sequence of agent–map vectors attention features. Then, we introduce a multimodal attention Transformer encoder [5], which is essentially an ensemble of cross-attention modules, to output multiple trajectories through the different attention modules, representing the different relations between the agent and map vectors. We still use an agent's interaction feature as query and agent–map vectors attention as key and value, yielding different encodings of the agent's relationship with the local scene context. Likewise, we apply the multimodal encoder to all agents to extract their possible interactions with the scene context and stack the results along the future axis.

4) Decoding Multimodal Future: For each agent, its agent–agent interaction encodings are repeated and concatenated with the multimodal agent–map interaction encoding to form a final feature vector. For the surrounding agents, we decode their possible future trajectories through a shared MLP from the final feature vector. For the AV, we decode its future control actions (i.e., acceleration and steering angle) from the feature vector through an MLP and translate the

action sequence to trajectory using a kinematic model (see Section III-C). We also output multiple possible trajectories for the AV to better model the interaction between agents. To predict the probability of each future (joint trajectories for all agents), we use max pooling to aggregate the information from all agents and map vectors and an MLP to decode the probability. The motion planner will take as input the future (i.e., the initial plan and other agent's prediction) with the highest probability.

C. Differentiable Motion Planning

1) Problem Statement: We consider an open-loop finite-horizon optimal planning problem, which is to find a sequence of control inputs $\mathbf{u} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_T\}$ to minimize the cost function in (1). The cost function is a sum of squared residual terms, each represented by a product of a weight ω_i and cost c_i [33], and the state of other surrounding agents $\hat{\mathbf{s}}$ is provided by the neural network predictor

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \frac{1}{2} \sum_i \left\| \omega_i c_i \left(\mathbf{u}^i, \hat{\mathbf{s}} \right) \right\|^2 \qquad (1)$$

where c_i is a function of subsets of the control sequence $\mathbf{u}^i \subset \mathbf{u}$ and potentially the predicted states of other surrounding agents $\hat{\mathbf{s}}$. In the training process, the gradient of the final planning error can be backpropagated to the predicted states of other agents and cost weights through the optimization problem [34], [35], to implicitly influence other agents' future behaviors and adjust the cost weights.

2) Kinematic Model: When calculating some costs, we need to convert the control action $\mathbf{u}_t = \{a_t, \delta_t\}$ (where a_t is acceleration and δ_t is steering angle) to state $\mathbf{s}_t = \{x_t, y_t, \theta_t, v_t\}$ (where (x_t, y_t) is 2-D position, θ_t is heading angle, and v_t is velocity). We adopt the kinematic bicycle model [36] shown in (2) to update the states of the AV

$$\begin{aligned} v_{t+1} &= v_t + a_t \Delta t \\ x_{t+1} &= x_t + v_t \cos(\theta_t) \Delta t \\ y_{t+1} &= y_t + v_t \sin(\theta_t) \Delta t \\ \theta_{t+1} &= \theta_t + \frac{v_t}{L} \tan(\delta_t) \Delta t \end{aligned} \qquad (2)$$

where L is the wheelbase of the vehicle and Δt is the time interval. The kinematic bicycle model is differentiable, thus permitting calculating gradients and Jacobians in the optimizer.

3) Cost Function: The cost function encompasses a variety of carefully crafted cost terms that encode different aspects of driving, including travel efficiency, ride comfort, traffic rules, and, most importantly, safety. The details of the different cost terms are given below.

a) Travel efficiency: We encourage the AV to reach the destination as fast as possible but not run above the speed limit. Therefore, the cost of travel efficiency is defined as the difference between the AV's speed v_t and speed limit

$$\mathbf{c}_t^{\text{speed}} = v_t - v_{\text{limit}} \qquad (3)$$

where v_{limit} is the speed limit of the lane.

b) Ride comfort: Human drivers prefer comfortable maneuvers, and we introduce four subcosts to represent the ride comfort factors the AV needs to optimize. They are longitudinal acceleration a_t and longitudinal jerk $\dot{a}_t$, as well as steering angle δ_t and steering change rate $\dot{\delta}_t$ for lateral stability and comfort

$$\begin{aligned} \mathbf{c}_t^{\text{acc}} &= a_t \\ \mathbf{c}_t^{\text{jerk}} &= \dot{a}_t \\ \mathbf{c}_t^{\text{steer}} &= \delta_t \\ \mathbf{c}_t^{\text{rate}} &= \dot{\delta}_t \end{aligned} \qquad (4)$$

where $\dot{a}_t = (\Delta a_t)/(\Delta t)$ and $\dot{\delta}_t = (\Delta \delta_t)/(\Delta t)$ are the discrete forms of the longitudinal jerk and steering rate, respectively.

c) Traffic rules: The AV is expected to adhere to the traffic rules and structure of the road. Thus, to promote staying close to the lane (route) centerline and following the lane direction, two subcosts are designed:

$$\begin{aligned} \mathbf{c}_t^{\text{pos}} &= p_t - p_{l,\perp} \\ \mathbf{c}_t^{\text{head}} &= \theta_t - \theta_{l,\perp} \end{aligned} \qquad (5)$$

where p_t and $p_{l,\perp}$ are the positions of the AV and closest point from the lane's centerline to the AV, respectively, and θ_t and $\theta_{l,\perp}$ are the heading angles of the AV and its closest point on the lane, respectively.

In addition, obeying traffic lights should be treated as a hard constraint for the AV. Here, we replace the hard constraint with a soft penalty term, which can be assigned with a large cost weight. We assume the AV runs along a predefined route, and its running distance is s_t, which is derived from $s_t = \sum_{t'=1}^{t} v_{t'} \Delta t$, and the stop line position (if encountering a red light) on the route is s_{stop}. We can formulate the cost of violating traffic lights using the hinge loss

$$\mathbf{c}_t^{\text{traffic}} = \begin{cases} s_t - s_{\text{stop}}, & s_t \geq s_{\text{stop}} \\ 0, & \text{otherwise.} \end{cases} \qquad (6)$$

It means that if the AV goes past the stop line at a red light, a large penalty will be induced, thus forcing the AV to stop near the stop point.

d) Safety: Keeping a safe distance from other traffic participants on the road to avoid collision is a fundamental requirement for AVs. However, optimizing the safe distances to all other agents in the scenario is unnecessary and time-consuming. Therefore, we take advantage of the Frenet frame [37], [38], which decouples the vehicle trajectory into the longitudinal direction along a predefined driving route and the lateral direction perpendicular to the route, to define the interactive agents. As illustrated in Fig. 3, all other agents' positions are projected to the Frenet frame with the AV's route as the reference path. At each future timestep, those agents whose predicted positions are within the route's conflict area are defined as the interactive agents. The calculation of safe distance at a specific timestep is given as follows:

$$d_{\text{safe}} = \min_i \left\| p_t - p_t^i \right\|_2 \qquad (7)$$

where p_t^i is the predicted position of the interactive agent i at future timestep t.

We also employ the hinge loss to formulate the safety cost to ensure the safe distance is large enough

$$\mathbf{c}_t^{\text{safety}} = \begin{cases} \epsilon - d_{\text{safe}}, & d_{\text{safe}} \leq \epsilon \\ 0, & \text{otherwise} \end{cases} \qquad (8)$$
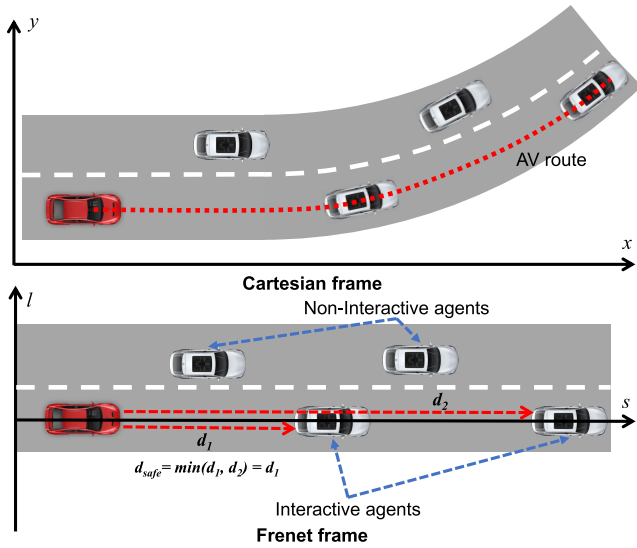
Fig. 3. Illustration of the calculation of safe distance. Other agents are first projected to the Frenet frame to find the interacting agents, and the calculation of distances is in the Cartesian frame.

where ϵ is the minimum safety distance requirement, which is defined as the sum of the lengths of two agents and a safety gap.

e) Weights: The cost terms in (1) consist of different features listed before at different timesteps. Here, we only consider the weights for different features, and the weights for individual features are the same at different timesteps. For features other than safety and traffic rules, we consider all the timesteps over the planning horizon. For the cost of traffic rule violation, we consider every other timestep across the future horizon to reduce computation costs. Because the computation for the safety cost is expensive, we only consider the trajectory waypoints at $[0.1, 0.3, 0.6, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0]$ s, which could significantly speed up the optimization without compromising safety. To ensure the constraints are satisfied, the weights for the constraints (i.e., safety and traffic light violations) in the cost function are set to a large enough value and fixed during training.

4) Differentiable Optimization: The Gauss–Newton algorithm is utilized to solve the nonlinear optimization problem [35]. It is an iterative least-squares optimization approach, where, at each iteration step, the nonlinear objective is first linearized around the current control variables to derive the linear system

$$\left(\sum_i J_i^\top J_i\right)\Delta\mathbf{u} = \sum_i J_i^\top \omega_i c_i \tag{9}$$

where $J_i = (\partial(\omega_i c_i))/(\partial\mathbf{u}^i)$ is the Jacobian of the cost with respect to the control variable.

We can explicitly solve the linear system using Cholesky decomposition of the normal equations to find the update $\Delta\mathbf{u}$ and eventually update the control variables

$$\mathbf{u} \leftarrow \mathbf{u} - \alpha\Delta\mathbf{u} \tag{10}$$

where $0 < \alpha \leq 1$ is the step size. In the original Gauss–Newton method, $\alpha = 1$, but it may cause the nonlinear objective not to decrease at every iteration. Therefore,

an improved version is setting $0 < \alpha < 1$ to employ a fraction of the update and, thus, mitigates the divergence issue.

Since the above procedure is fully differentiable, we can set up other differentiable components (e.g., prediction and cost function weights) and integrate them into the planner, enabling the whole architecture to be differentiable. Moreover, to start the optimization, one has to provide an initial guess for the control variables, which is crucial to the convergence to the global optimum, and we use the control actions given by the prediction network as the initial guess, which is also differentiable.

D. Learning Process

At a high level, we regard the motion planner as the primary component of our framework. During the forward pass, the motion planner takes as input the prediction results of other agents and the initial plan (with the highest probability), as well as the cost function, to start the trajectory optimization and iteratively updates the planned trajectory until passing a convergence check or reaching the maximum number of iterations. At the end of the optimization, the motion planner will output a trajectory for the AV to follow. In the backward pass, we evaluate the loss function on the planned trajectory, compute gradients with respect to the initial plan, cost function weights, as well as the predicted trajectories of other agents, and backpropagate the loss through the neural components and update the parameters, in order to generate better quality trajectories. The learning process of our framework is stable and efficient, because all the components (i.e., prediction, initial plan, and cost function) are learned in a supervised manner. Compared with generative adversarial IL (GAIL) [39], [40], which learns proxy reward signals through adversarial learning, our method only learns the weights of the cost function terms. However, the initialization of the cost function weights is important in our method, because this can affect the optimization process. The learning process of our framework is summarized in Algorithm 1.

Since all operations in the framework are differentiable, we can train the entire framework in an end-to-end fashion from real-world driving data, and we implement the framework using PyTorch and Theseus [33]. The loss function for training the framework encompasses four terms: prediction loss for all agents, score loss to accurately predict the probabilities of different futures, as well as imitation loss and planning cost for the AV. The overall loss is defined as follows:

$$\mathcal{L} = \lambda_1\mathcal{L}_{\text{prediction}} + \lambda_2\mathcal{L}_{\text{score}} + \lambda_3\mathcal{L}_{\text{imitation}} + \lambda_4\mathcal{L}_{\text{cost}} \tag{11}$$

where λ_1–λ_4 are the weights to scale these different loss terms. For prediction loss, we treat each future to be coherent individual futures across all agents and backpropagate the smooth $L1$ losses on the trajectories through the joint future that most closely matches the ground truth

$$\mathcal{L}_{\text{prediction}} = \sum_{k=1}^{K} \mathbb{1}(k = \hat{k}) \sum_{i=1}^{N} \text{smooth}L_1\left(\xi_i^k - \xi_i^{\text{gt}}\right) \tag{12}$$

where $\mathbb{1}$ is the indicator function, $\hat{k}$ is the index of the best-predicted joint future of multiple agents (the future with the smallest sum of displacement errors across all agents,

Algorithm 1 Learning of the Proposed DIPP Framework

Require: Neural network predictor f_θ, initial cost function weights $\{\omega_i\}$, differentiable planner g

1: Get the environment information (Agent tracks $\mathbf{X}$ and vectorized map $\mathbf{M}$)
2: Predict the AV and other agents' multi-modal joint trajectories $\{\xi_i^k\}_{k=1:K, i=0:N}$, $\{p_k\}_{k=1:K} \leftarrow f_\theta(\mathbf{X}, \mathbf{M})$
3: Calculate prediction loss $\mathcal{L}_{prediction}$ and score loss $\mathcal{L}_{score}$
4: Get the most-likely prediction result $\{\xi_i^{\hat{k}}\}_{i=0:N}$, $\hat{k} = \arg\min_k \sum_i \|\xi_i^k - \xi_i^{gt}\|_2$ ▷ Testing $\hat{k} = \arg\max\{p_k\}$
5: Plan the AV's trajectory using the differentiable planner $\xi_{AV}, \mathcal{L}_{cost} \leftarrow g(\xi_0^{\hat{k}}, \{\xi_i^{\hat{k}}\}_{i=1:N}, \{\omega_i\})$
6: Calculate imitation loss $\mathcal{L}_{imitation}$
7: Calculate total loss $\mathcal{L}$ according to Eq. (11)
8: Backpropagate loss and calculate gradients with respect to θ and $\{\omega_i\}$
9: Update f_θ and $\{\omega_i\}$ using Adam optimizer

$\hat{k} = \arg\min_k \sum_i \|\xi_i^k - \xi_i^{gt}\|_2$), and ξ_i^{gt} is the individual ground-truth trajectory. The scoring loss is defined as follows:

$$\mathcal{L}_{score} = \sum_{k=1}^{K} \mathbb{1}(k = \hat{k}) \log p_k \qquad (13)$$

where p_k is the probability of future k.

The motion planner would take as input the most-likely trajectories of all agents, where the AV's trajectory (original control actions) is used as planning initialization, and other agents' future trajectories are used in computing the safety cost. The output of the motion planner ξ_{AV} is compared against the ground-truth trajectory ξ_{AV}^{gt} to calculate the imitation loss

$$\mathcal{L}_{imitation} = \text{smooth}L_1(\xi_{AV} - \xi_{AV}^{gt}). \qquad (14)$$

IV. EXPERIMENTS

A. Dataset

We train and validate our framework on the Waymo Open Motion Dataset [41], a large-scale real-world driving dataset that focuses on urban driving scenarios with complex and diverse road structures and traffic interactions. The dataset contains 103 354 unique driving scenes, each 20 s in duration, with detailed map data and agent tracks. We randomly select 10% of the dataset, which corresponds to 100 data files out of 1000 files in total, resulting in a total of 10 156 different driving scenes. Since the driving scenes are randomly stored in the data files and the selected driving scenes are a random subset of the dataset, the data distribution of the reduced dataset remains the same as the original. To segment a 20-s driving scene into frames (each frame contains 7-s object tracks at 10 Hz), we set up a 7-s time window with a 2-s history observation horizon and a 5-s planning/prediction horizon. The window slides from the beginning of the scene with a stride of ten timesteps (1 s), resulting in 14 segments of 7-s frames. For training, we use 80% of the selected scenes, while the remaining 20% are used for validation and open-loop testing. The total number of frames used for training is 1 113 622, while 28 396 frames are used for validation and open-loop testing.

B. Evaluation

We evaluate the performance of the framework in both open-loop and closed-loop manners. In the open-loop testing, the motion planner plans a trajectory for the AV based on the current states and prediction results at each frame, and we compare the planned trajectory against the AV's ground-truth trajectory. In the closed-loop testing, we construct a log-replay simulator, where at each step, the AV will take the first action from its planned trajectory and update its state, while the neighboring agents will follow their record trajectories in the dataset. We set up the following metrics to evaluate the performance of the framework.

1) Safety: The collision rate is employed to measure the safety performance of the system. For open-loop testing, collision is calculated based on the AV's planned trajectory and the ground-truth future trajectories of other agents. If a collision is detected at any step in the plan, then the frame is deemed as a collision. For closed-loop testing, we check if the AV collides with other agents at every time step during the simulation.

2) Traffic Rule Violation: We consider deviating from the route and passing over the stop line at a right light as traffic rule violations, and we calculate the total number of frames where the AV violates the traffic rules.

3) Vehicle Dynamics: Three metrics are introduced to quantify rider comfort and feasibility of the planned trajectory, which are longitudinal acceleration and jerk, as well as lateral acceleration. They are absolute values averaged over time in a scene and will be compared against human driving metrics.

4) Human Driving Similarity: We use the position errors between the planned trajectory in open-loop testing or rollout trajectory in closed-loop testing and the ground truth to quantify the human likeness. We calculate the position errors at different timesteps (i.e., 1, 3, and 5 s in open-loop testing and 3, 5, and 10 s in closed-loop testing).

5) Prediction: We calculate the average displacement error (ADE) and final displacement error (FDE) of the multiagent joint trajectories from the most-likely future to reflect the performance of the prediction module. ADE computes the $L2$ norm between the ground-truth state $s_{a,t}$ and the most-likely joint prediction $\hat{s}_{a,t}^{\hat{k}}$: $1/(AT) \sum_a \sum_t \|s_{a,t} - \hat{s}_{a,t}^{\hat{k}}\|_2$, where $\hat{k} = \arg\max p_k$ is the index of the most-likely predicted joint future. FDE calculates the $L2$ distance between the ground-truth and predicted states at the final step T: $(1/A) \sum_a \|\hat{s}_{a,T} - \hat{s}_{a,T}^{\hat{k}}\|_2$.

C. Comparison Baselines

We compare our proposed method with the following baselines to reveal the effects and advantages of the proposed framework.

1) Vanilla IL: Based on the same scene context, we use only IL to train a network to directly generate the trajectory for the AV. The structure of the network is the same as the proposed method, but the prediction of other agents is removed and only one trajectory instead of multimodal trajectories is generated. The IL policy is trained on the same dataset with the same hyperparameters as the proposed method, which makes sure the comparison results are fair.

2) IL With Prediction Subtask: We train a multitask neural network with the main task of IL and the subtask of predicting other agents' future trajectories. This is equivalent to the proposed network without the differentiable motion planner, and the training settings and hyperparameters are the same as the proposed method.

3) Separated Prediction–Planning: Based on the same network, we separately train a prediction module without the integrated motion planner to output the initial trajectory for the AV and predicted trajectories for other agents, which is the same as the IL with the prediction model. In testing, a motion planner, which is the same as the proposed method, is utilized to perform trajectory planning for the AV with the trained prediction model and a prespecified cost function.

4) Conservative Q-Learning [42]: We utilize a widely used offline reinforcement learning algorithm that can learn to make decisions from offline datasets. We implement the conservative Q-learning (CQL) method with the d3rlpy offline RL library [43] using the default hyperparameters. The observation space is a three-channel red, green, blue (RGB) bird-eye-view image $256 \times 256 \times 3$ of the driving scene [44], the action is the target pose of the next step (Δx, Δy, and $\Delta \theta$) relative to the ego vehicle's current position, and the reward function is the distance traveled per step plus an extra reward for reaching the goal. The policy is trained on the same scenarios as the proposed method, and all transition steps (observations, actions, rewards, and terminals) in each scenario will be used. The trained policy is only evaluated in the closed-loop planning test.

5) Intelligent Driver Model [45]: We utilize a well-known model-based driving model, intelligent driver model (IDM), to control the longitudinal movement of the ego vehicle along the reference route. The IDM planner takes as input the current state of the ego vehicle (position and speed) and the state of the leading agent (position, speed, and length) under the coordinate of the reference route and outputs the acceleration of the ego, which is then translated to the coordinate in the scene. The IDM method is only used in the closed-loop planning test.

D. Implementation Details

1) Network: The network outputs three possible futures (joint trajectories of all agents) and their probabilities. For other agents, we predict the displacements relative to their current locations instead of their coordinates, which shows significant improvement in prediction accuracy. The learning of the cost function is also incorporated into the network and implemented by a small MLP, which takes a fixed dummy input and outputs the weights of the cost function. We set the cost weights for traffic light violations and collisions as large values (larger than two orders of magnitude) and unlearnable to ensure the constraints are satisfied.

2) Training: We use a batch size of 32 and an Adam optimizer with a learning rate that starts from $2e - 4$ and decays by a factor of 0.5 every four epochs. The total number of training epochs is 20, and we pretrain the framework for five epochs without the planner to get good initial prediction results and control actions. For speed considerations, we set the max number of iterations for the motion planner to 2, which will also encourage the network to produce high-quality

TABLE I
HYPERPARAMETERS OF THE MODEL AND TRAINING PROCESS

Parameter	Value
Number of neighbors N	10
Number of predicted futures K	3
Historical timesteps H	20
Future timesteps T	50
Step size α	0.4
Planning iterations	2
Initial learning rate	2e-4
Total training epochs	20
Pretraining epochs	5
Batch size	32
Weight for prediction loss λ_1	0.5
Weight for score loss λ_2	1
Weight for imitation loss λ_3	1
Weight for planning cost λ_4	1e-3

initial plans, and the step size for the Gauss–Newton update is $\alpha = 0.4$. The weights for the loss function are set to $\lambda_1 = 0.5$, $\lambda_2 = 1$, $\lambda_3 = 1$, and $\lambda_4 = 0.001$. The results shown in Sections V-A and V-B are based on this loss function setting, and we investigate the effects of the loss function weights in Section V-C. We clip the gradient norm of the network parameters with the max norm of the gradients as 5.

3) Testing: In the testing process, the max number of iterations for the motion planner is set to 50, the step size is set to 0.2, and the absolute error tolerance is set as $1e-2$, in order to plan a high-quality trajectory for the AV. Acceleration and jerk in the longitudinal and lateral directions are calculated based on the positions and headings on the trajectory. To check if a collision happens, we use a list of circles to approximate an object at each timestep. If the distance between any two circles' origins of the given two objects is lower than a threshold, it is considered as they collide with each other.

The main hyperparameters used in this article are summarized in Table I.

V. RESULTS AND DISCUSSION

A. Open-Loop Test

1) Multimodal Joint Prediction: Fig. 4 shows two representative scenarios where the neural network predictor makes multifuture predictions, as well as the probabilities of the predicted futures. In Scenario 1, the AV consistently chooses to stop behind the leading vehicle in all futures, while different interacting behaviors among the two agents in the intersection area are predicted. The future (joint trajectories) closest to ground truth is assigned with a higher probability. In Scenario 2, the predictor delivers different future predictions for both the AV and the other interacting agent at an unsignalized intersection. For example, the AV would choose to go if the other agent is predicted to turn left and stop if the other agent's trajectory is predicted to conflict with the AV's route. The close-to-ground-truth futures are also assigned with higher probabilities. These results indicate that the neural network predictor is able to provide accurate multimodal joint predictions for all agents in the scene and a good initial guess for the motion planner.

2) Qualitative Results: Fig. 5 displays some representative interactive scenarios, demonstrating the proposed framework's

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

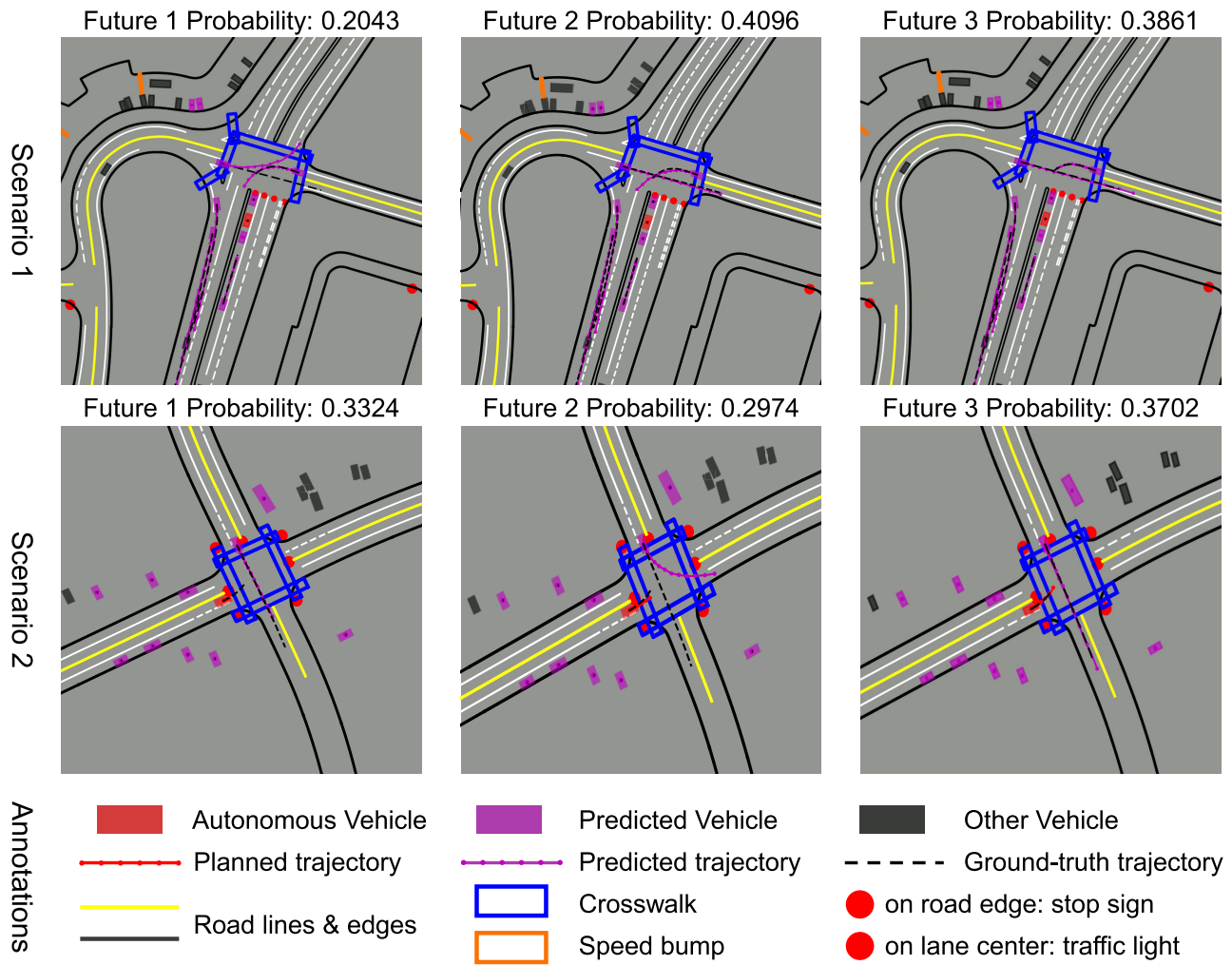HUANG et al.: DIFFERENTIABLE INTEGRATED MOTION PREDICTION AND PLANNING 9

Fig. 4. Multimodal predictions given by the neural network predictor. The trajectories of the ten nearest agents to the AV are predicted by the neural network.

ability to plan a safe, traffic rule-compliant trajectory for the AV, based on the prediction results of other surrounding agents' future states (including vehicles, cyclists, and pedestrians). The results reveal that the motion planner is capable of handling a variety of urban driving scenarios involving a mixture of interacting traffic participants, including making a smooth right turn, yielding to pedestrians on the crosswalk, stopping at the red light, and yielding to another vehicle at an unprotected left turn. In Fig. 6, we compare the proposed method with other baseline methods in two interactive scenarios. We can see that without explicit constraints, the neural network-planned trajectory gradually deviates from the lane centerline, while the motion planner's output trajectory adheres to the road structure. Our proposed method can generate planning-aware or planning-centric prediction results, which can help the planner deliver more close-to-human trajectories compared with using separate prediction results. For example, in the merging scenario, our method predicts that the interacting vehicle should yield to the AV at a stop sign, leading to more accurate prediction and planning results. In the interaction with a cyclist, our method predicts that the cyclist may move in front of the AV (however, not really happen in the data), and the motion planner generates a slow-down trajectory to avoid potential risks, which is in accordance

with the human-like trajectory. The result indicates that even if the prediction result is not entirely accurate, the planning module would benefit from such planning-centric results and produce human-like plans. The prediction of other agents in our framework is similar to the driver's belief rather than an exact reflection of the ground-truth dynamics.

3) Quantitative Results: Following the previously defined evaluation metrics, we list the quantitative results of our method and other baseline methods in Table II. The results of open-loop testing indicate that the proposed method significantly outperforms the IL-based methods in terms of collision and traffic rule violation rates. The IL-based methods have a much higher off-route rate due to the lack of explicit constraints on following the route, and without considering the ride comfort, they yield an unacceptable lateral acceleration, which is significantly worse than the planning-based methods and human driving. On the other hand, the planning-based methods deliver much lower acceleration and jerk, which ensures smoothness and ride comfort. Because the neural network is only trained to predict the trajectories by direct regression, IL with the multiagent prediction subtask method achieves the smallest planning error (compared with the ground-truth trajectories). However, since IL methods ignore other important factors, they perform very poorly in the
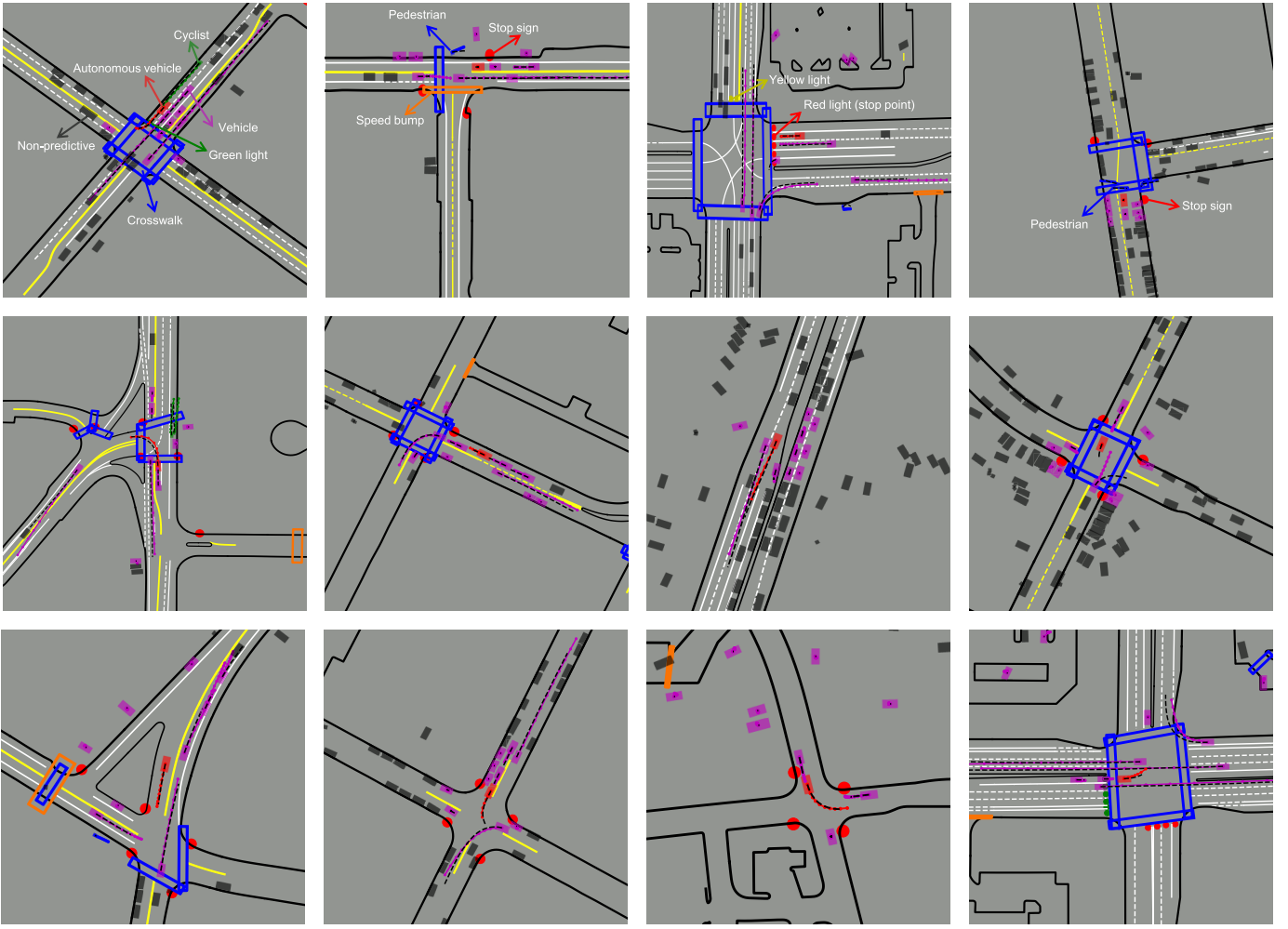
Fig. 5. Qualitative results of the proposed framework in open-loop testing. The colored solid lines are the planned or predicted trajectories for AV or surrounding agents, and black dotted lines are the ground-truth trajectories.

Vanilla imitation learning   Imitation learning + forecasting   Separate forecasting + planning   Ours
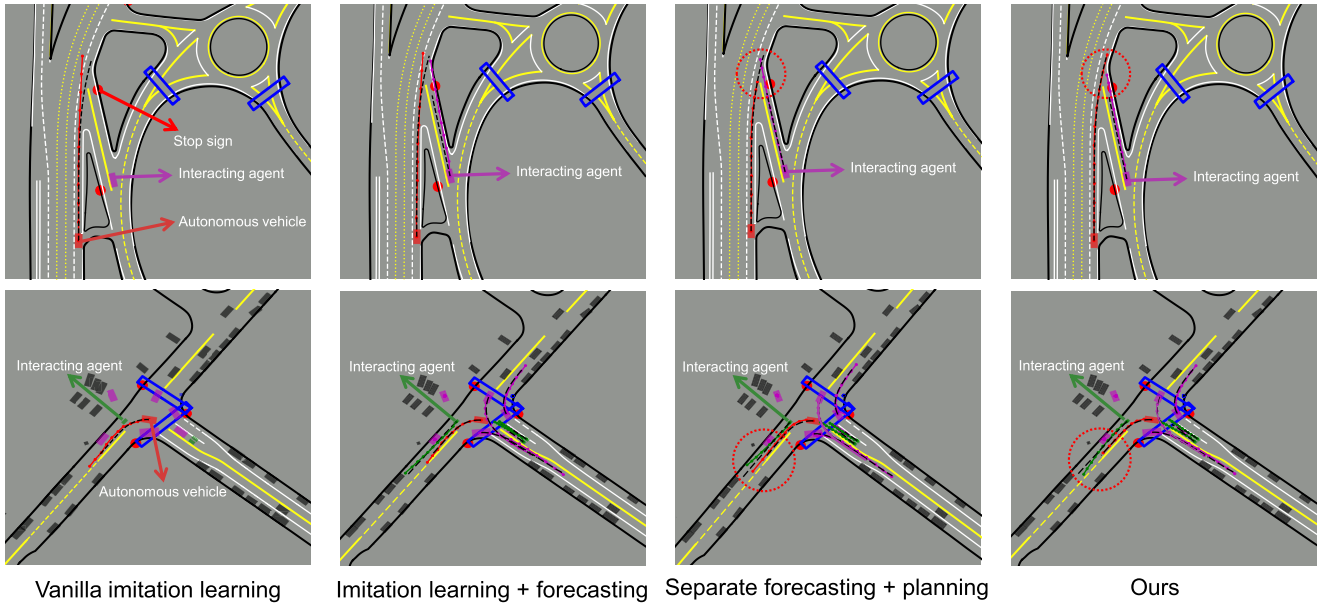
Fig. 6. Qualitative comparison between the proposed method and baseline methods in open-loop testing. Top: interacting with a vehicle. Bottom: interacting with a cyclist.

closed-loop test and barely finish a task (see Section V-B). One major conclusion from the results is that our proposed DIPP method, which jointly trains the predictor and planner, outperforms the planner with a separate trained prediction module, in terms of both planning and prediction errors. The improvements in DIPP may come from two possible ways:

TABLE II
OPEN-LOOP EVALUATION OF THE PROPOSED METHOD AGAINST BASELINE METHODS

Method	Collision (%)	Red light (%)	Off route (%)	Acc. (m/s^2)	Jerk (m/s^3)	Lat. Acc. (m/s^2)	Planning error (m) @1s	@3s	@5s	Prediction error (m) ADE	FDE
IL	5.469	2.772	4.816	0.546	2.881	3.151	0.175	1.416	4.194	–	–
IL+Prediction	3.930	1.670	4.542	0.672	4.004	3.939	**0.127**	**0.892**	**2.901**	0.773	1.916
Sep. Plan+Pred.	1.813	1.327	0.527	0.269	0.149	0.077	0.238	1.251	3.466	0.766	1.908
DIPP (ours)	**1.802**	**1.235**	**0.506**	0.269	0.150	0.079	0.227	1.187	3.335	**0.740**	**1.814**
Human	–	–	–	0.620	1.707	0.101	–	–	–	–	–

Sep. Plan+Pred.: separated plan + prediction, Acc.: acceleration, Lat. Acc.: Lateral acceleration
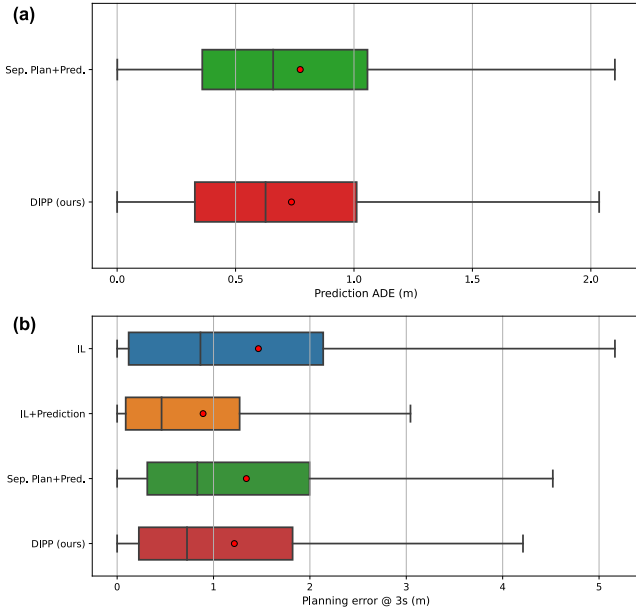

Fig. 7. Boxplots of the prediction and planning errors of the proposed method and other baselines in the open-loop planning test. (a) Prediction ADE. (b) Planning error @ 3 s.

1) the additional regularization from the planning module enables the prediction module to make fewer prediction errors that would negatively affect planning (e.g., large deviations from lane center and unnecessary stops) and 2) without significantly impacting raw prediction accuracy, the influence of the planning task enables the prediction module to make planning-centric results that would lead to better and human-like plans (e.g., react to possible cut-in of other agents). Another interesting finding is that though the vanilla IL method performs the worst, adding the prediction subtask can significantly reduce the collision rate and planning error, which shows the importance of multiagent joint prediction. Fig. 7 shows the boxplots of the prediction error of the proposed method compared against the separated prediction–planning method and the planning error of all methods in open-loop tests. The results further clarify the conclusion that differentiable and integrated training of the prediction and planning modules (DIPP) can improve both the prediction and planning metrics in open-loop tests.

B. Closed-Loop Test

1) Qualitative Results: To fully reveal the planning performance of our proposed method, we conduct a closed-loop test with 100 replayed scenes from the testing set. Specifically, we build a log-replay simulator to roll out the AV's planned trajectory, where other agents follow their original tracks in the dataset and only the AV's state gets updated. At each timestep, the AV plans a trajectory according to the prediction result and takes the first action from its planned trajectory and replans at the next time step. The simulation horizon is 15 s, and the interval is 0.1 s. To evaluate the closed-loop testing performance, we add a progress metric, which measures the running distance of the AV until it reaches the end of the scene, collides with other agents, or goes off route. We measure the position error between the AV's rollout trajectory and its ground-truth one at several time steps to reflect the similarity to the human driver. The results of the closed-loop test are given in Table III, and we provide Supplementary Videos[1] of our method navigating in a variety of urban scenarios for better visualization and evaluation of our method. It is important to note that the collision rate is just a lower bound of safety, since other agents do not react to the AV (e.g., the ego vehicle runs slower than the human driver in the data). The Supplementary Videos encompass various driving scenes where the proposed framework can handle different tasks, such as cruising, obeying traffic lights, making smooth turns, and, more importantly, interacting with other road users. We also add some scenarios where other vehicles would take adversarial actions toward the AV (not intentionally as the AV has deviated from its original track), but our planning framework can handle such emergencies and avoid collisions.

2) Qualitative Results: The results in Table III indicate that the IL-based methods cannot finish a single scene without causing a collision or going off route. Although the planning error of the IL + prediction method is the smallest in open-loop testing, the position errors of IL-based methods are significantly worse. The ride comfort is also compromised, as IL-based methods deliver much higher jerk and acceleration. This is because the IL-based methods suffer from distributional shifts, which means the compounding errors lead the AV to a situation that deviates from the training distribution and degrade its performance. Similar to IL-based methods, the CQL method performs poorly in the test and cannot finish a single task because of the difficulty of training and labeling the correct reward for human driving behaviors. On the other hand, the planning-based methods leveraging structural information and domain knowledge perform significantly better despite learning purely from an offline dataset. The motion planner with a separate trained prediction module shows similar yet worse performance compared with our proposed method,

[1]https://mczhi.github.io/DIPP/

TABLE III
CLOSED-LOOP EVALUATION OF THE PROPOSED METHOD AGAINST BASELINE METHODS

Method	Collision (%)	Off route (%)	Progress (m)	Acc. (m/s^2)	Jerk (m/s^3)	Lat. Acc. (m/s^2)	Position error (m) @3s	@5s	@10s
IL	40	60	8.23	1.857	3.249	1.872	11.29	19.42	43.63
IL+Prediction	42	58	11.05	1.345	1.977	0.813	11.03	19.13	42.25
CQL	50	50	16.38	3.456	8.798	2.154	12.85	20.18	45.10
IDM	28	0	49.44	0.885	2.236	0.081	3.895	9.568	30.32
Sep. Plan+Pred.	7	0	76.28	0.638	1.313	0.058	1.869	4.182	10.17
DIPP (Ours)	**5**	**0**	**77.57**	0.624	1.209	0.069	**1.726**	**3.913**	**9.365**
Human	–	–	–	0.621	2.067	0.105	–	–	–

Sep. Plan+Pred: separated plan + prediction, Acc.: acceleration, Lat. Acc.: Lateral acceleration

TABLE IV
ABLATION STUDY ON THE IMPORTANCE OF EACH COMPONENT
IN OPEN-LOOP TESTING

Model	Planning error (m)	Prediction error (m)
Base (ours)	1.583	0.740
Base w/ oracle prediction	1.388	–
No learnable cost function	1.834	0.740
No learnable initialization	2.672	0.740
No learnable prediction	1.811	1.871

TABLE V
ABLATION STUDY ON THE IMPORTANCE OF EACH
COMPONENT IN CLOSED-LOOP TESTING

Model	Failure (%)	Progress (m)	Position error (m)
Base (ours)	5	76.28	4.083
Base w/ oracle prediction	6	76.16	4.153
No learnable cost function	13	69.09	5.881
No learnable initialization	17	57.52	6.571
No learnable prediction	19	44.01	9.971

which highlights the advantages of integrated training of the planning and prediction modules. Aside from its simplicity and few parameters, the IDM method is more effective in closed-loop testing, since we only regulate the ego vehicle's longitudinal motion on the reference route. However, the performance of the IDM method is inferior to the planning-based approach, which is more flexible and can handle highly interactive and complex scenarios.

C. Ablation Study

1) Effects of Learnable Components: To unveil the importance and function of each key component in our proposed framework, an ablation study is conducted. We set up three baselines in which the learnable components (i.e., initial plan, cost function, and prediction) are dropped out one at a time. We also set up an oracle method that uses the ground-truth future trajectories of other agents as the prediction results to examine the capabilities of the proposed motion planner. For the non-learnable cost function baseline, the cost function used by the motion planner is manually tuned to balance the different cost terms. For the non-learnable initialization baseline, the initial guess to the motion planner is fixed as ($a_t = 0$ and $\delta_t = 0$) at all timesteps. For the non-learnable prediction baseline, the motion prediction module is a constant turn rate and velocity model. The results of the ablation study are summarized in Tables IV and V. In open-loop testing, we select the planning and prediction errors as the main metrics, and the planning error is averaged over three timesteps (1, 3, and 5 s), and ADE is used as the prediction error. In closed-loop testing, we employ the failure rate (sum of the collision and off-route rates), progress, and average position error (3, 5, and 10 s) as the main metrics.

The open-loop testing results in Table IV indicate that learnable initialization is the most important part of the framework to ensure the planned trajectory is close to the human-driving one. This suggests that a good initial plan for the planner

is critical to solving the optimization and learnable initialization could substantially improve the planning performance. Another important factor for planning is the cost function, and we demonstrate that learning the cost function from data can render the planned trajectories closer to humans. Surprisingly, using only a physics-based non-learnable prediction model would not significantly deteriorate the planning performance in an open-loop setting though the prediction error is notably higher. Using the oracle predictor can improve the motion planner's performance in open-loop testing, which suggests that the proposed motion planner performs well and could achieve even better performance with more training data and improved prediction accuracy. The closed-loop testing results in Table V reveal that the prediction module plays a more critical role in ensuring safety and human likeness. The gap between open-loop and closed-loop performance is due to that the closed-loop testing set contains many interactive scenarios, where accurately predicting other agents' future states is required. In contrast to open-loop testing, our method can deliver comparable performance to the oracle prediction method in closed-loop testing, marking the advantage of our method (planning-aware prediction) when deploying the system in practice. In addition, learnable initialization and cost function are necessary to stabilize the planning process (solving the optimization problem) and obtain better planning performance, which, otherwise, could lead to instability to find viable solutions and consequently worse safety and reliability. In summary, all three learnable components in our framework are integral to maintaining the final planning performance.

2) Effects of Weights of the Training Loss Function: We investigate the influence of the weights in the loss function (11) and test the trained models in the open-loop planning setting. To reveal the sensitivity of the framework performance to these values, we set up the following groups of loss function weights: 1) the default parameter setting in the experiment; 2) the weight for the prediction loss λ_1 is increased by 5×; 3) the weight for the prediction loss λ_1 is decreased to

TABLE VI
EFFECTS OF WEIGHTS OF THE LOSS FUNCTION IN OPEN-LOOP TESTING

No.	λ_1	λ_2	λ_3	Planning error (m)	Prediction error (m)
1	0.5	1	1	1.583	0.740
2	2.5	1	1	1.837	0.733
3	0.1	1	1	1.642	1.244
4	0.5	5	1	1.704	0.820

TABLE VII
EFFECTS OF THE PLANNER UPDATE STEP SIZE AND PLANNING
ITERATIONS IN OPEN-LOOP TESTING

step size α	planning iteration	Planning error (m)	Prediction error (m)
0.4	2	1.583	0.740
1.0	2	1.856	0.867
0.4	3	1.551	0.725
0.2	3	1.549	0.797

1/5 of the default value; and 4) the weight for the score loss λ_2 is increased by 5×. The weights for imitation loss λ_3 and planning cost λ_4 are fixed, and the same metrics in Table IV are used to evaluate the framework's performance.

The results in Table VI indicate that the default loss parameters in our experiments (No. 1) achieve the best planning performance. Although increasing the weight of prediction loss (No. 2) can slightly improve the prediction error, the planning performance is compromised, because the loss signal of the initial plan or cost function in motion planning is largely neglected in the total loss function. However, decreasing the weight of prediction loss (No. 3) still does not bring better planning performance, because the prediction results are required to be more accurate for both the ego vehicle and other agents. In addition, increasing the weight of score loss (No. 4) may not improve probability evaluation and even lead to worse planning or prediction performance. The results suggest that our multitask learning framework requires a trade-off between the planning and prediction tasks, and the default loss function weights can strike a better balance between the tasks.

3) Effects of the Step Size and Planning Iterations: We investigate the effects of the step size [α in (10)] and planning iterations in the motion planner on the final open-loop planning test. Note that the different parameters are only applied in the training process, while the parameters of the motion planner in the testing process are the same. We explore different parameter settings in Table VII, where the value of the step size α ranges from 0.2 to 1, and the maximum planning iterations are 3 for speed consideration. The same metrics for open-loop testing are used to evaluate the performance of the trained models.

The results in Table VII indicate that using the default planner setting provides satisfactory results in the final open-loop planning test. Setting the step size to the maximum value ($\alpha = 1$) could significantly impair the testing performance, because the large step size makes the optimization process unstable, which negatively impacts the learning process of the prediction module and the cost function weights. While increasing the number of planning iterations with the default step size ($\alpha = 0.4$) could slightly improve the planning and prediction metrics, this would also result in longer computing time for the planner, making it an unnecessary choice. On the other

hand, using a smaller step size ($\alpha = 0.2$) along with additional planning iterations could also achieve comparable performance to the default setting, but may still be an unnecessary choice. Overall, the default planner parameter setting provides a good balance between final performance and computational efficiency.

4) Results of Cost Function Learning: We report detailed results for the learned cost function and hand-tuned one in Table VIII. The traffic light violation and safety weights are fixed at 10, because they are considered constraints, and other parameters are learnable. The results indicate that using the learned cost function as opposed to the hand-tuned cost function significantly reduces planning errors in the open-loop planning test and the failure rate in the closed-loop test. In general, the learned weights are reasonable from the viewpoint of human drivers. Adhering to the driving route (for both positions and directions) is the most critical factor, and driving comfort (acceleration and change rate of steering) is also an important factor to optimize. The steering angle is not a significant factor in the cost, since the vehicle primarily needs to follow the route, but speed and jerk cost terms are assigned with smaller weights due to their relatively large scales. It is important to note that the cost terms have different scales, and the weights are learned to balance the scales to facilitate the optimization process. This means that the learned cost function weights may not reflect their actual weights in human driving and are, thereby, unsuitable for other frameworks.

D. Discussions

1) Strengths: The direct benefit of our method is to integrate structural information and domain knowledge into machine learning models, or from a different perspective, to apply machine learning models to some hard-to-solve points in traditional optimization problems. We apply our method to the challenging autonomous driving task and focus on solving real-world problems (e.g., how to specify the cost function and how other agents' actions would affect the ego) by learning from real-world data. The integration of prediction, cost function, and planning modules in our framework leads to end-to-end, decision-focused learning, that trains the whole pipeline to optimize the final planning performance directly. Experiments demonstrate that the proposed method has excellent open-loop and closed-loop performance and robustness against distributional shifts and adversarial agents, despite only learning from offline data. In addition, the proposed method outperforms the separated planning and prediction method in both planning and prediction performance, which underscores the advantage of our method to integrate the prediction and planning modules in an end-to-end differentiable way.

2) Limitations and Future Work: Some limitations of this work should be acknowledged. First is the computation time of the framework. The average runtime of the prediction + planning step is 0.012 s per sample at training (with NVIDIA RTX 3080 GPU, two iterations) and 1.78 s at testing (with AMD 3900X CPU, max 50 iterations). The bottleneck in the framework is the computation of Jacobin in solving the least-squares optimization. However, we need to note that this work is just a prototype of the proposed framework, and the running efficiency has not been optimized. We plan to further improve the efficiency of the computation pipeline to reduce the

TABLE VIII
DETAILED RESULTS FOR THE LEARNED PLANNING COST

Cost Weights	ω_1 ($\mathbf{c}^{speed}$)	ω_2 ($\mathbf{c}^{acc}$)	ω_3 ($\mathbf{c}^{jerk}$)	ω_4 ($\mathbf{c}^{steer}$)	ω_5 ($\mathbf{c}^{rate}$)	ω_6 ($\mathbf{c}^{pos}$)	ω_7 ($\mathbf{c}^{head}$)	ω_8 ($\mathbf{c}^{traffic}$)	ω_9 ($\mathbf{c}^{safety}$)	Planning error (m)	Failure (%)
Hand-tuned	0.1	0.5	0.1	0.01	0.5	0.5	5	10	10	1.834	13
Learned	0.055	0.29	0.13	0.014	0.46	0.31	3.23	10	10	**1.583**	**5**

runtime in solving the optimization, potentially enabling the framework to run near real time. Another limitation is that we do not thoroughly consider the uncertainty or multimodality of human behavior prediction, only taking the most probable one as the planner input. In future work, the planner can take all modalities and uncertainties along the timesteps into account to make more informed decisions or perform contingency planning.

VI. CONCLUSION

We propose a novel differentiable integrated multi-agent interactive prediction and motion planning framework, which is trained end-to-end from real-world driving data. A Transformer-based predictor is established to predict joint future trajectories of surrounding agents and provide an initial guess for the planner. The predicted trajectories, initial plan, and learnable cost function are channeled to an optimization-based differentiable motion planner, allowing every component in the framework to be differentiable. The framework is validated on a large-scale urban driving dataset in both open-loop and closed-loop testing. The open-loop testing results reveal that our proposed method outperforms traditional pipeline methods and IL-based methods in terms of planning and prediction performance, ride comfort, and safety metrics. The closed-loop testing results indicate that planning-based methods significantly outperform IL-based methods (though have better similarity to human trajectories in open-loop testing), which suggests that our method overcomes the distributional shift problem common in offline learning. Moreover, the proposed method outperforms the pipeline method in closed-loop testing, emphasizing the benefit of joint training of prediction and planning modules. In the ablation study, we find that all learnable components in the framework are integral to maintaining the stability of the optimizer and desired planning performance. The proposed framework has the potential to accelerate the development of autonomous driving decision-making systems by enabling all components to be learned from real-world data.

REFERENCES

[1] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, "A survey of end-to-end driving: Architectures and training methods," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 4, pp. 1364–1384, Apr. 2022.

[2] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016.

[3] J. Ma, Z. Cheng, X. Zhang, Z. Lin, F. L. Lewis, and T. H. Lee, "Local learning enabled iterative linear quadratic regulator for constrained trajectory planning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, May 2, 2022, doi: 10.1109/TNNLS.2022.3165846.

[4] P. Hang, C. Lv, C. Huang, J. Cai, Z. Hu, and Y. Xing, "An integrated framework of decision making and motion planning for autonomous vehicles considering social behaviors," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14458–14469, Dec. 2020.

[5] Z. Huang, X. Mo, and C. Lv, "Multi-modal motion prediction with transformer-based neural network for autonomous driving," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 2605–2611.

[6] H. Gao, Y. Qin, C. Hu, Y. Liu, and K. Li, "An interacting multiple model for trajectory prediction of intelligent vehicles in typical road traffic scenario," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 31, 2021, doi: 10.1109/TNNLS.2021.3136866.

[7] X. Mo, Z. Huang, Y. Xing, and C. Lv, "Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9554–9567, Jul. 2022.

[8] S. Chen and J. Wang, "Heterogeneous interaction modeling with reduced accumulated error for multiagent trajectory prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Nov. 28, 2022, doi: 10.1109/TNNLS.2022.3224007.

[9] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," in *Proc. 29th Int. Conf. Mach. Learn.*, 2012, pp. 475–482.

[10] Y. Xu, J. Xie, T. Zhao, C. Baker, Y. Zhao, and Y. N. Wu, "Energy-based continuous inverse optimal control," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, May 5, 2022, doi: 10.1109/TNNLS.2022.3168795.

[11] Z. Huang, J. Wu, and C. Lv, "Driving behavior modeling using naturalistic human driving data with inverse reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 10239–10251, Aug. 2022.

[12] A. Sadat, M. Ren, A. Pokrovsky, Y. Lin, E. Yumer, and R. Urtasun, "Jointly learnable behavior and trajectory planning for self-driving vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 3949–3956.

[13] J. Ngiam et al., "Scene transformer: A unified architecture for predicting future trajectories of multiple agents," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–25.

[14] Á. Llamazares, E. J. Molinos, and M. Ocaña, "Detection and tracking of moving obstacles (DATMO): A review," *Robotica*, vol. 38, no. 5, pp. 761–774, May 2020.

[15] J. Gao et al., "VectorNet: Encoding HD maps and agent dynamics from vectorized representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11522–11530.

[16] B. Varadarajan et al., "MultiPath++: Efficient information fusion and trajectory aggregation for behavior prediction," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 7814–7821.

[17] J. Gu, C. Sun, and H. Zhao, "DenseTNT: End-to-end trajectory prediction from dense goal sets," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 15283–15292.

[18] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "THOMAS: Trajectory heatmap output with learned multi-agent sampling," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–18.

[19] Z. Zhou, L. Ye, J. Wang, K. Wu, and K. Lu, "HiVT: Hierarchical vector transformer for multi-agent motion prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2022, pp. 8823–8833.

[20] X. Jia, L. Sun, H. Zhao, M. Tomizuka, and W. Zhan, "Multi-agent trajectory prediction by combining egocentric and allocentric views," in *Proc. 5th Annu. Conf. Robot Learn.*, 2021, pp. 1434–1443. [Online]. Available: https://openreview.net/forum?id=lAtePxetBNb

[21] L. Chen et al., "What data do we need for training an AV motion planner?" in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 1066–1072.

[22] Z. Huang, C. Lv, Y. Xing, and J. Wu, "Multi-modal sensor fusion-based deep neural network for end-to-end autonomous driving with scene understanding," *IEEE Sensors J.*, vol. 21, no. 10, pp. 11781–11790, May 2021.

[23] Z. Huang, J. Wu, and C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jan. 26, 2022, doi: 10.1109/TNNLS.2022.3142822.

[24] A. Kendall et al., "Learning to drive in a day," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8248–8254.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

HUANG et al.: DIFFERENTIABLE INTEGRATED MOTION PREDICTION AND PLANNING   15

[25] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 740–759, Feb. 2022.

[26] A. Cui, S. Casas, A. Sadat, R. Liao, and R. Urtasun, "LookOut: Diverse multi-future prediction and planning for self-driving," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 16087–16096.

[27] J. L. V. Espinoza, A. Liniger, W. Schwarting, D. Rus, and L. Van Gool, "Deep interactive motion prediction and planning: Playing games with motion prediction models," in *Proc. Learn. Dyn. Control Conf.*, 2022, pp. 1006–1019.

[28] H. Song, W. Ding, Y. Chen, S. Shen, M. Y. Wang, and Q. Chen, "PiP: Planning-informed trajectory prediction for autonomous driving," in *Proc. Eur. Conf. Comput. Vis.* Glasgow, U.K.: Springer, 2020, pp. 598–614.

[29] J. Liu, W. Zeng, R. Urtasun, and E. Yumer, "Deep structured reactive planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 4897–4904.

[30] S. Casas, A. Sadat, and R. Urtasun, "MP3: A unified model to map, perceive, predict and plan," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 14398–14407.

[31] B. Ivanovic, A. Elhafsi, G. Rosman, A. Gaidon, and M. Pavone, "MATS: An interpretable trajectory forecasting representation for planning and control," in *Proc. 4th Annu. Conf. Robot Learn.*, 2020, pp. 1–14.

[32] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[33] L. Pineda et al., "Theseus: A library for differentiable nonlinear optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022, pp. 3801–3818.

[34] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for end-to-end planning and control," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–12.

[35] M. Bhardwaj, B. Boots, and M. Mukadam, "Differentiable Gaussian process motion planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 10598–10604.

[36] R. Rajamani, "Lateral vehicle dynamics," in *Vehicle Dynamics and Control*. New York, NY, USA: Springer, 2012, pp. 15–46.

[37] Y. Zhang, H. Sun, J. Zhou, J. Hu, and J. Miao, "Optimal trajectory generation for autonomous vehicles under centripetal acceleration constraints for in-lane driving scenarios," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 3619–3626.

[38] Y. Zhang, H. Sun, J. Zhou, J. Pan, J. Hu, and J. Miao, "Optimal vehicle path planning using quadratic optimization for Baidu Apollo open platform," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 978–984.

[39] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.

[40] N. Baram, O. Anschel, I. Caspi, and S. Mannor, "End-to-end differentiable adversarial imitation learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 390–399.

[41] S. Ettinger et al., "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9690–9699.

[42] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1179–1191.

[43] T. Seno and M. Imai, "d3rlpy: An offline deep reinforcement learning library," *J. Mach. Learn. Res.*, vol. 23, no. 315, pp. 1–20, 2022. [Online]. Available: http://jmlr.org/papers/v23/22-0017.html

[44] Z. Huang, X. Mo, and C. Lv, "ReCoAt: A deep learning-based framework for multi-modal motion prediction in autonomous driving application," in *Proc. IEEE 25th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2022, pp. 988–993.

[45] C. Zhang and L. Sun, "Bayesian calibration of the intelligent driver model," 2022, *arXiv:2210.03571*.
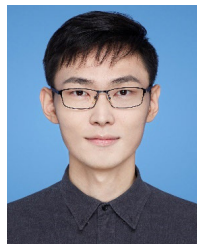
Zhiyu Huang (Graduate Student Member, IEEE) received the B.E. degree from the School of Automobile Engineering, Chongqing University, Chongqing, China, in 2019. He is currently pursuing the Ph.D. degree with the School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore.

His current research interests include behavior prediction and decision-making with machine learning-based methods for autonomous driving.
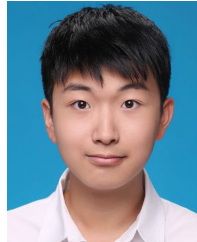
Haochen Liu (Graduate Student Member, IEEE) received the B.E. degree from the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China, in 2021. He is currently pursuing the Ph.D. degree with the School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore.

His current research interests include deep learning-enabled motion prediction and decision-making.

Jingda Wu (Graduate Student Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from the Beijing Institute of Technology, Beijing, China, in 2016 and 2019, respectively. He is currently pursuing the Ph.D. degree with the School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore.

His research interests include human guidance-based reinforcement learning algorithms, human–artificial intelligence (AI) collaborated driving strategy design, decision-making of autonomous vehicles, and data-driven energy management of electrified vehicles.
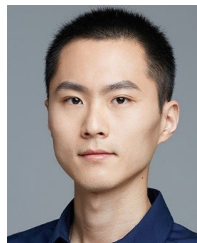
Chen Lv (Senior Member, IEEE) received the Ph.D. degree from the Department of Automotive Engineering, Tsinghua University, Beijing, China, in 2016.

He was a joint Ph.D. Researcher with the Electrical Engineering and Computer Sciences (EECS) Department, University of California at Berkeley, Berkeley, CA, USA, from 2014 to 2015, and a Research Fellow with the Advanced Vehicle Engineering Center, Cranfield University, Cranfield, U.K., in 2016 and 2018. He is currently an Assistant Professor with Nanyang Technology University, Singapore. His research interests include advanced vehicles and human–machine systems, where he has contributed over 100 papers and obtained 12 granted patents.

Dr. Lv received the Highly Commended Paper Award of Institution of Mechanical Engineers (IMechE), U.K., in 2012, the NSK Outstanding Mechanical Engineering Paper Award in 2014, the China Society of Automotive Engineering (CSAE) Outstanding Paper Award in 2015, the Tsinghua University Outstanding Doctoral Thesis Award in 2016, the CSAE Outstanding Doctoral Thesis Award, and the IEEE IV Best Workshop/Special Session Paper Award in 2018. He serves as a Guest Editor for the IEEE/ASME TRANSACTIONS ON MECHATRONICS (TMECH), *IEEE Intelligent Transportation Systems* (ITS) *Magazine*, and *Applied Energy*. He serves as an Associate Editor/Editorial Board Member for the *International Journal of Vehicle Autonomous Systems*, *Frontiers in Mechanical Engineering*, and *Vehicles*.