# Gen-Drive: Enhancing Diffusion Generative Driving Policies with Reward Modeling and Reinforcement Learning Fine-tuning

Zhiyu Huang<sup>1</sup>, Xinshuo Weng<sup>2</sup>, Maximilian Igl<sup>2</sup>, Yuxiao Chen<sup>2</sup>, Yulong Cao<sup>2</sup>, Boris Ivanovic<sup>2</sup>, Marco Pavone<sup>2,3</sup>, Chen Lv<sup>1</sup>

Abstract—Autonomous driving necessitates the ability to reason about future interactions between traffic agents and to make informed evaluations for planning. This paper introduces the Gen-Drive framework, which shifts from the traditional prediction and deterministic planning framework to a generation-then-evaluation planning paradigm. The framework employs a behavior diffusion model as a scene generator to produce diverse possible future scenarios, thereby enhancing the capability for joint interaction reasoning. To facilitate decision-making, we propose a scene evaluator (reward) model, trained with pairwise preference data collected through VLM assistance, thereby reducing human workload and enhancing scalability. Furthermore, we utilize an RL fine-tuning framework to improve the generation quality of the diffusion model, rendering it more effective for planning tasks. We conduct training and closed-loop planning tests on the nuPlan dataset, and the results demonstrate that employing such a generationthen-evaluation strategy outperforms other learning-based approaches. Additionally, the fine-tuned generative driving policy shows significant enhancements in planning performance. We further demonstrate that utilizing our learned reward model for evaluation or RL fine-tuning leads to better planning performance compared to relying on human-designed rewards. Project website: https://mczhi.github.io/GenDrive.

### I. Introduction

Navigating complex environments requires autonomous driving agents to adeptly anticipate future scenarios (e.g., the behaviors of other agents) while making informed decisions [1], [2]. Conventional predictive and deterministic planning approaches often separate the prediction and planning processes [3], [4], which isolates the ego vehicle from the social context and often results in behaviors that do not comply with social driving norms. Although integrated prediction-planning frameworks [5]-[9] have been proposed to address this issue, they still rely on deterministic planning, which poses challenges in addressing the uncertainties, multi-modality, and mutual interactive dynamics of agent behaviors. To overcome these challenges, we propose the adoption of generation-evaluation methods for the planning task. The key idea is that our approach integrates the ego agent into the social interaction context, generates a range of possible outcomes for all agents in the entire scene, and employs a learned scene evaluator to guide decision-making.

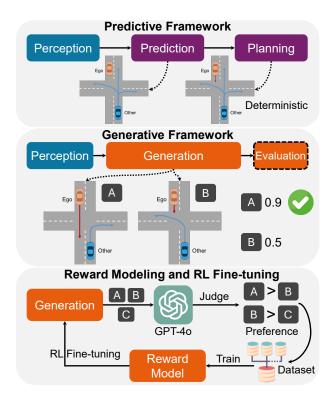


Fig. 1. *Gen-Drive* represents a paradigm shift from conventional prediction and deterministic planning approaches to a generation-then-evaluation framework. In this framework, different joint future scenes for both the ego agent and other agents are generated, followed by a selection process through a scene evaluation (reward) model. To train an effective reward model, we sample generation outcomes and employ a VLM-assisted pipeline to curate a pairwise preference dataset. The trained reward model can be utilized to make informed decisions and fine-tune the generation model via RL, further enhancing its planning performance.

While generative models, particularly diffusion models, have seen extensive use in simulation and prediction tasks in autonomous driving [10]–[14], their application in decision-making tasks has been relatively limited.

Two primary limitations hinder the application of generative models in the planning task. First, it is complicated to evaluate generated scenarios and select the optimal one for decision-making that aligns with human expectations and values. To resolve this, we introduce a scene evaluation (reward) model trained on preference data derived from VLM feedback, enabling better decision-making. Second, unlike simulation or scenario generation tasks that benefit from a diversity of samples, planning with generative models requires producing more probable future scenarios with fewer samples to minimize computational overhead and runtime delays. We address this by introducing a reinforcement

 $<sup>^1</sup>School$  of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore. E-mail: <code>zhiyu001@e.ntu.edu.sg</code>, <code>lyuchen@ntu.edu.sg</code>

<sup>&</sup>lt;sup>2</sup>NVIDIA Research, NVIDIA Corporation, Santa Clara, CA, USA. E-mail: {xweng, migl, yuxiaoc, yulongc, bivanovic, mpavone}@nvidia.com

<sup>&</sup>lt;sup>3</sup>Department of Aeronautics and Astronautics, Stanford University, USA.

learning (RL) fine-tuning framework [15] that enhances the quality of diffusion generation based on the obtained reward model. The results demonstrate significant improvements in the model's planning performance after fine-tuning.

In this paper, we introduce Gen-Drive, a diffusion generative driving policy, along with its training framework. The policy model comprises a query-centric scene context encoder in vector space [16], a diffusion-based scene generator, and a scene evaluator to assess the quality of the generated scenes for planning. The training process involves three stages. Initially, we utilize a large volume of real-world driving data to train the base diffusion model, consisting of the scene encoder and generator. Subsequently, we curate a dataset of pairwise preference data on scenarios generated by the base diffusion model, employing a hybrid labeling pipeline assisted by vision language models (VLMs), and then train a scene evaluator based on the dataset. Finally, we employ reinforcement learning from AI feedback [17] to fine-tune the diffusion generator, enhancing its efficacy in planning tasks. The comprehensive Gen-Drive framework and its training process are illustrated in Fig. 1. The contributions of this paper are summarized as follows:

- We develop a multi-agent trajectory diffusion model to reason about the interactions among all agents and generate diverse, scene-consistent future scenarios.
- We train a reward model from a curated VLMfeedback preference dataset that is able to evaluate the goodness of the scenarios generated by the model.
- We build an RL fine-tuning pipeline to enhance the performance of the diffusion driving policy based on the learned reward model.

The base model is trained using the nuPlan dataset, and evaluated on the nuPlan closed-loop planning benchmark [18]. The results reveal that our diffusion driving policy achieves favorable performance, particularly after fine-tuning.

# II. RELATED WORK

# A. Prediction and Planning for Autonomous Driving

Prediction-planning frameworks form the cornerstone of decision-making in autonomous driving, which predicts the behaviors of other agents and then plans for the ego agent. Recent advancements in deep learning have significantly enhanced the accuracy of prediction models [19]-[21]. Nonetheless, simply enhancing prediction accuracy may not necessarily translate to better planning performance [8], [22]. To address this, integrated prediction and planning (IPP) methods have been proposed [5], [6], [8], [23], aiming to optimize overall planning performance. Although IPP approaches can directly enhance planning performance, they still isolate the ego vehicle from social interaction contexts and struggle to handle the inherent multi-modality of the prediction results. Moreover, the complexity of designing IPP methods and their complicated training requirements present challenges to their practical deployment. Consequently, we propose a shift from predictive to generative approaches, by developing a model capable of generating diverse future scenarios for all agents, including the ego vehicle. This is intended to ensure scene consistency, better capture interaction dynamics, and enhance final planning performance.

### B. Reward Modeling for Autonomous Driving

Reward modeling or scene/planning evaluation is a crucial yet challenging task. Traditional evaluations primarily rely on metrics and functions crafted by humans, such as nuPlan score [18] and Predictive Driving Model (PDM) score [24]. These metrics, while useful, may not reflect human values accurately across different scenarios and can result in planning models trained with these rewards diverging from human-like behaviors. A promising direction of reward modeling is to learn from human driving data using inverse reinforcement learning (IRL) [7], [8], [25]. However, it requires assumptions about the structure of the reward function that may not reflect actual human preferences under varying conditions. Recently, leveraging pairwise human preference data to train reward models has gained popularity [26], which can further be applied to fine-tune generative models [17]. This approach shows promise in delivering human-aligned evaluations and enhancing human-like driving capabilities.

# C. Generative Models for Autonomous Driving

Generative models, such as diffusion and auto-regressive Transformer models, have been increasingly utilized in traffic simulation and trajectory prediction tasks [10], [20], [27]-[32]. These models excel in capturing complex and multimodal distributions of multi-agent joint behaviors, leading to exceptional performance in scenario generation performance [10], [27], [28], [32]. Although auto-regressive generation models, such as BehaviorGPT [27] and SMART [28], are adept at generating interactive behaviors, they fall short in diversity and controllability compared to diffusion models. Consequently, diffusion models represent a promising alternative. Notable examples in diffusion-based scenario generation include CTG++ [12] and VBD [10], which form the foundation of our framework. A particular application of diffusion models in planning is Diffusion-ES [33], but it is mainly used as a trajectory optimization method for the ego agent instead of generating multi-agent future scenarios. In our proposed Gen-Drive model, we aim to improve planning performance with proper reward modeling and to refine the diffusion model for planning. Training diffusion models with RL, previously successful in image generation [34], [35], is employed to improve the model's generation quality by directing outcomes towards high-rewarding plans/scenarios and reducing computational cost.

### III. METHOD

We employ a generative (diffusion) model to replace the prediction-planning models in the conventional paradigm. The critical distinction is that the ego agent is not isolated from the scene; rather, it is considered an integral part, with all agents' behaviors being deeply interdependent. To leverage this generative model for planning, we design a scene evaluation (reward) model. This model is trained using

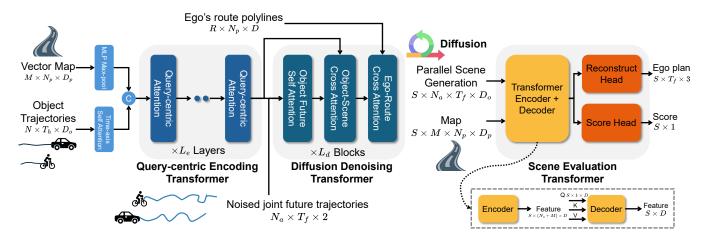


Fig. 2. Neural network structure of the *Gen-Drive* model. The query-centric encoding Transformer encodes all scene elements in local coordinates while preserving relative information in attention calculations. The diffusion denoising Transformer comprises multiple attention blocks that iteratively attend to noised object futures, future-scene, and ego-route interactions. During diffusion generation, different scenes can be produced in parallel and then be fed into the scene evaluation Transformer. The evaluation model utilizes a Transformer encoder-decoder to fuse information from the future scene and map, and two MLP heads are used to reconstruct the ego plan and output a score for the scene/plan.

a curated dataset of pairwise human preferences, enabling it to directly score the generated scenarios (plans) and facilitating the selection of optimal and contextually aligned decisions. Furthermore, we utilize the reward model to fine-tune the diffusion generation process, steering it towards generating high-rewarding plans. The RL fine-tuning step can enhance overall planning performance and reduce the need for extensive sampling. The neural network structure of the *Gen-Drive* model is illustrated in Fig. 2.

# A. Scene Generator

For an initial driving scene at the current timestep, we consider N objects (including the ego) and M map elements, tracking the historical trajectories of these objects over  $T_h$  timesteps. The current scene inputs to the encoder consist of object trajectories  $\mathcal{O} \in \mathbb{R}^{N \times T_h \times D_o}$  and map polylines  $\mathcal{M} \in \mathbb{R}^{M \times N_p \times D_p}$ , where  $N_p$  is the number of waypoints and  $D_o$  and  $D_p$  are the dimensional features of each point.

**Encoder**. The current scene inputs are initially encoded through a time-axis self-attention Transformer layer for the object trajectories, yielding  $\mathcal{O}^e \in \mathbb{R}^{N \times D}$ , and through an MLP with max-pooling for the map data, resulting in  $\mathcal{M}^e \in \mathbb{R}^{M \times D}$ . They are concatenated to form an initial encoding. We employ a query-centric Transformer encoder [10], [16], [21] to fuse the features of the scene elements and produce a comprehensive scene condition encoding  $\mathcal{C} \in \mathbb{R}^{(N+M) \times D}$ .

**Denoiser.** The diffusion process operates in the joint action space a of all objects of interest  $N_a$ , and action consists of acceleration and yaw rate. The noise is directly added to the action sequence. Given the noise input  $\mathbf{a}_k \in \mathbb{R}^{N_a \times T_f \times 2}$ , where k is the noise level,  $T_f$  is the future timesteps, as well as the scene condition  $\mathcal{C}$ , we employ a denoising Transformer with self-attention and cross-attention layers to predict the denoised action sequence  $\hat{\mathbf{a}}_0$ . For the ego agent, additional route information is provided and an additional cross-attention layer is employed to model egoroute relations. Further details on the diffusion process and model structure can be found in our previous work [10].

**Generation**. Future scenarios (joint object actions) are generated starting from random Gaussian noise  $\mathbf{a}_K \sim \mathcal{N}(0, \mathbf{I})$ , where K is the total number of diffusion steps. Subsequently, each diffusion step k involves sampling from the transition dynamics specified below [36]–[38]:

$$\mu_k := \frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1 - \bar{\alpha}_k} \hat{\mathbf{a}}_0 + \frac{\sqrt{\alpha_k}(1 - \bar{\alpha}_{k-1})}{1 - \bar{\alpha}_k} \mathbf{a}_k, \tag{1}$$

$$p(\mathbf{a}_{k-1}|\mathbf{a}_k) = \mathcal{N}\left(\mathbf{a}_{k-1}; \mu_k, \frac{1 - \bar{\alpha}_{k-1}}{1 - \bar{\alpha}_k} \beta_k \mathbf{I}\right),$$
 (2)

where  $\alpha_k$ ,  $\bar{\alpha}_k$ , and  $\beta_k$  are derived from a predetermined noise schedule. By iteratively reversing the diffusion step, we obtain the final denoised joint action output  $\mathbf{a}_0$ . Subsequently, states are derived using a dynamics model f to translate object actions into states  $\mathbf{x}_0 = f(\mathbf{a}_0)$ . The state encompasses x/y coordinates, heading, and velocity of the object.

### B. Scene Evaluator

The scene evaluator takes as input S future scenarios produced by the diffusion generator, which can be generated in parallel by initiating from a batch of Gaussian noise. These generated scenarios are structured as  $\mathcal{S} \in$  $\mathbb{R}^{S \times N_a \times T_f \times D_o}$ , and another input to the evaluator is the vector map  $\mathcal{M}$ . These future scenes are encoded using a query-centric Transformer encoder, similar to the encoding of historical scenes, resulting in a scene feature representation  $\mathcal{S}^e \in \mathbb{R}^{S \times (N_a + M) \times D}$ . Subsequently, we employ the ego agent's future or planning encoding  $\mathcal{A}^e \in \mathbb{R}^{S \times \tilde{1} \times \tilde{D}}$  extracted from the scene encoding  $S^e$  as query, and the scene encoding as key and value in a Transformer decoder, to derive the planning-centric feature of the future scenes  $A \in \mathbb{R}^{S \times D}$ . Note that the Transformer decoder attends to the  $(N_a + M)$ elements within each scene individually. Two MLP heads are appended to this feature tensor to reconstruct the planning trajectories of the ego agent and output scores for different generated scenes (planning trajectories), respectively. The ego planning reconstruction head is added as an auxiliary task to enhance stability and effectiveness.

# C. Training Base Diffusion Model

The base diffusion model is trained to recover clean trajectories from noised joint trajectory inputs under various noise levels and scene conditions. At each training step, a noise level k and Gaussian noise are sampled to perturb the original action trajectories. Since the model predicts scenelevel joint trajectories, all object trajectories are affected by the same noise level. The training loss function for the base diffusion model can be formulated as:

$$\mathcal{L}_{\mathcal{G}} = \mathbb{E}_{(\mathcal{O}, \mathcal{M}) \sim D, k \sim \mathcal{U}(0, K)} \left[ \mathcal{SL}_{1} \left( f \left( \mathcal{G}(\mathcal{O}, \mathcal{M}, \mathbf{a}_{k}, k) \right) - \mathbf{x} \right) \right],$$
(3)

where D is the dataset,  $\mathcal{SL}_1$  is the Smooth L1 loss, f is the dynamics model, x is the ground-truth future states of the objects.  $\mathcal{G}$  denotes the diffusion model that predicts clean trajectories given noised trajectories  $a_k$  and scene conditions.

# D. Training Reward Model

Pairwise preference data collection. To build an effective reward model, it is essential to curate a comprehensive dataset. One approach involves utilizing humandesigned metrics such as the PDM score [24]. However, relying on such metrics presents significant limitations, as they may not accurately reflect actual human values across diverse scenarios. Additionally, accurately labeling scenes with reward values is challenging even for human evaluators. Alternatively, we can engage human annotators to perform pairwise comparisons, determining which scenarios more align with human preferences. Nevertheless, curating a largescale reward dataset imposes a substantial workload on human annotators. To address this, we employ VLMs to enhance the efficiency and scalability of the process.

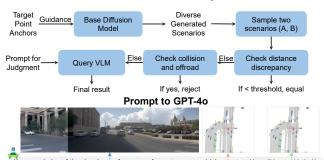
The VLM-assisted reward labeling pipeline is illustrated in Fig. 3. To increase the diversity of planning trajectories, we first extract 32 5-second anchor goals from data utilizing the K-means algorithm and employ guided diffusion [10], [12] to generate 32 diverse planning trajectories for the ego agent along with reactive behaviors of other objects in the scene by the model. Subsequently, we conduct pairwise sampling of these scenarios. We first compute discrepancies between the planned trajectories, and then check collisions and off-road to filter out obvious failure cases. If these measures are insufficient for distinction, we utilize GPT-40 to provide a conclusive evaluation. As illustrated in Fig. 3, GPT-40 provides reasonable evaluations of the two generated scenarios based on the current scene context.

**Training process.** At each training step, we sample a batch of pairwise comparison results, i.e., accepted (positive) scenes  $S_a$  and rejected (negative) scenes  $S_r$  from the same initial conditions. The loss function for training the scene evaluation model is formulated as follows [17], [39]:

$$\mathcal{L}_{\mathcal{R}} = \mathbb{E}_{(\mathcal{S}_a, \mathcal{S}_r) \sim D_r} \left[ -\log \sigma \left( \mathcal{R}(\mathcal{S}_a) - \mathcal{R}(\mathcal{S}_r) \right) \right], \quad (4)$$

where  $D_r$  is the pairwise preference reward dataset, and  $\mathcal{R}$  denotes the reward model that predicts the score of the generated scene. Note that the reconstruction loss is omitted for brevity. Some examples of the reward model outputs

#### Preference Annotation Pipeline



You are a judge of the planning performance of an autonomous driving system. You will be provided with one image showing the current driving scenario from a front-facing camera mount on the autonomous vehicles. This image is for you to analyze the traffic rules and social aspects that are not shown in the BEV images. Here is the front-facing image of the current driving scenario, (front facing camera image). Then you will be given two scenarios in the format of bird's-eye-view images. Scenario A: (bev\_image\_a), Scenario B: (bev\_image\_b). Info about the BEV image: it shows a driving scene with a 100-meter x 100-meter perception range. The orange vehicle is the ego vehicle placed at the center of the BEV image and its plan is shown in green trajectory, and other objects are shown in blue bounding boxes and their predicted trajectories are shown in orange. Different parts of the BEV image are maps of the scene, e.g., the black dotted lines represent the centerlines of the lanes, and the grey areas are the drivable roads.

Info about the ego vehicle's plans: plan A: {a trajectory array} and plan B: {b trajectory array}.

waypoint is the format of (x, y, heading, velocity x, velocity y). The planning horizon is 5 seconds and the frequency is 2Hz. The origin is the ego vehicle's current position, which is placed at the center of the BEV -axis is the vertical direction of the BEV image, and Y-axis is the horizontal direction of the BEV ir ed on your knowledge of human driving behaviors and your awareness of the situation, judge which one of the given scenarios is better or equivalent to the other in driving. You should first directly give your in the format of [A is better] or [B is better] or [A and B are equally good] and then justify your choice You should consider different factors, especially interactions with surrounding objects and traffic rules

#### Result

(B) IB is better]. Analyzing the two BEV (Bird's Eye View) scenarios where orange represents the ego vehicle and In Scenario A, the trajectory seems consistent and straightforward initially, with gradual curvature adjustments. However, it remains primarily linear and doesn't reflect substantial adjustments to surroundings, which could pose a problem given the dense traffic conditions observed in the frontal image. Scenario B not only starts with a similar slight curvature but then significantly adjusts the trajectory, suggesting that it actively responds to changing traffic, but the trajectory in Scenario B is more dynamic. There is a notable adjustment in speed and turning as seen in the greater spread and the curvature of waypoints, reflecting agile responsiveness to traffic conditions, a critical consideration in busy urban settings ......

In conclusion, considering the more realistic and responsive trajectory planning in interacting with

Scenario B stands out as the better planning approach for the presented autonomous driving system

Fig. 3. Pipeline for collecting planning preference data using VLM.

are illustrated in Fig. 4, and the results demonstrate that our trained reward model produces reasonable scores for the generated plans and scenes.

# E. Fine-tuning Generation Model

To enhance the efficacy of diffusion generation for planning tasks, we propose fine-tuning the diffusion model using the trained reward model and RL. We can formulate the diffusion denoising process as a multi-step MDP [15], [35], where the denoiser functions as a policy conditioned on the noise input at each step. The trajectory comprises K timesteps, with a reward signal emitted at the end of the diffusion process. The RL objective is to maximize the cumulative reward along the trajectory, and we can utilize denoising diffusion policy optimization (DDPO) [35] to refine the generative policy. The fine-tuning loss is formulated as:

$$\mathcal{L}_F = \sum_{k=K}^{0} \left[ \mathcal{L}_{policy} + \alpha \mathcal{S} \mathcal{L}_1(\hat{\mathbf{x}}_0 - \mathbf{x}) \right], \tag{5}$$

$$\mathcal{L}_{policy} = -\min \left\{ r(\mathbf{x}_0) \frac{p(\mathbf{x}_{k-1}^e | \mathbf{x}_k^e; \theta)}{p(\mathbf{x}_{k-1}^e | \mathbf{x}_k^e; \theta_{old})}, \right.$$

$$r(\mathbf{x}_0) \operatorname{Clip} \left( \frac{p(\mathbf{x}_{k-1}^e | \mathbf{x}_k^e; \theta)}{p(\mathbf{x}_{k-1}^e | \mathbf{x}_k^e; \theta_{old})}, 1 - \epsilon, 1 + \epsilon \right) \right\},$$

$$(6)$$

where  $\hat{\mathbf{x}}_0$  is the denoised state trajectories for all objects,  $\mathbf{x}$ is the ground-truth trajectories,  $\mathbf{x}_k^e$  is the state trajectory for the ego agent;  $r(\mathbf{x}_0)$  is the reward of the generated scene;  $\alpha$ is a regularization parameter, and  $\epsilon$  is the clipping parameter.

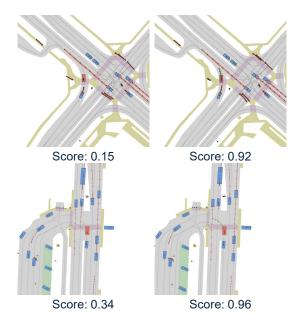


Fig. 4. Examples of scene evaluation (reward model) outputs; plans that ensure safety, interactivity, and making progress receive higher scores.

A regression term is added to ensure the denoised trajectory does not deviate from the original one.

Note that the fine-tuning loss is accumulated over the entire diffusion trajectory, and only the denoiser is learnable while the encoder is fixed during fine-tuning. The RL finetuning algorithm with DDPO is illustrated in Algorithm 1.

# Algorithm 1 Fine-tuning of generation with DDPO

```
Require: Pre-trained generation model \mathcal{G}_{\theta_{pre}}, trained reward
      model \mathcal{R}, sample size m, learning rate \gamma, fine-tuning
      steps T, update iterations I
  1: Initialize: \mathcal{G}_{\theta_1} \leftarrow \mathcal{G}_{\theta_{pre}}
```

```
2: for t \leftarrow 1 to T do
```

Generate m samples of generation sequence  $\begin{aligned} \{\mathbf{a}_k^{(i)}\}_{k=0}^K & \text{ from the current diffusion model } \mathcal{G}_{\theta} \\ & \text{Collect } m \text{ samples of scene reward } \{r^{(i)}\} \text{ using } \mathcal{R} \\ & \text{Normalize rewards } \{r^{(i)}\} = \frac{r^{(i)} - mean(\{r^{(i)}\})}{std(\{r^{(i)}\})} \end{aligned}$ 

5:

6:

Compute loss function  $\mathcal{J}(\theta) = \frac{1}{m} \sum_{M} \mathcal{L}_{F}$ Refer to Eq. (5)

Update  $\theta_{t+1} \leftarrow \theta_t - \gamma \nabla_{\theta} \mathcal{J}(\theta)$ 

end for 9:

end for

4:

8:

11: **Output:** Fine-tuned generation model  $\mathcal{G}_{\theta_T}$ 

# F. Implementation Details

**Model Parameters.** The scene context consists of N =100 objects, each with a 2-second historical trajectory, sampled at 0.5-second intervals ( $T_h = 4$ ). The vector map contains M=350 elements, with each comprising  $N_p=20$ waypoints, and additional R=30 route polylines for the ego vehicle. The base diffusion model consists of 6 encoding layers with query-centric attention and 6 Transformer decoding layers, with a hidden dimension of D=256. The model generates future scenes for  $N_a = 50$  objects closest to the ego agent over a 5-second horizon, with 0.5-second intervals ( $T_f = 10$ ). We employ K = 10 diffusion steps and a cosine noise schedule. The scene evaluator comprises 3 query-centric attention layers in the Transformer encoder and 3 cross-attention layers in the Transformer decoder.

**Training Pipeline.** For training the base diffusion model, we employ a batch size of 16 per GPU and 20 training epochs. AdamW optimizer is used, starting with a learning rate of 2e-4, which decays 0.95 every 1k steps. For the reward model, we collect 5k scenarios, each with 50 pairwise comparisons, and train it using a batch size of 32 per GPU across 50 epochs. In the fine-tuning phase, the parameters are set: regularization  $\alpha = 10$ , clip parameter  $\epsilon = 0.01$ , sample size m=32 per GPU, learning rate  $\gamma=1e-5$ , fine-tuning steps T = 1000, and update iterations per step I = 5. All training jobs are conducted on 8 NVIDIA A100 GPUs.

# IV. EXPERIMENTS

# A. Experimental Setup

Dataset. We employ the OpenScene dataset [24], which is a compact subset of the nuPlan dataset sampled at 2Hz. Only the vectorized data, including trajectories and maps, were employed to train the base behavior diffusion model. The training set comprises 450k segments, each comprising a 2second history and a 5-second future. No data augmentation methods are used. For reward model training and generation fine-tuning, we select 5k scenarios from the test split.

Closed-loop Planning Test. We test the model's performance on the nuPlan closed-loop planning task with nonreactive agents [18]. The planning frequency is set at 1 Hz, and the trajectory controller is set to iLQR. Evaluation metrics include the overall planning score, collision score, and progress score (higher scores indicate better performance). We adopt the reduced Val14 set as the benchmark [40].

# B. Main Results

The closed-loop planning results for different models are presented in Table I. Furthermore, Fig. 5 illustrates some typical scenarios from the generation process, and the finetuned policy shows better planning performance. Closed-loop planning results can be found on the project website.

TABLE I CLOSED-LOOP PLANNING RESULTS ON NUPLAN REDUCED VAL14 BENCHMARK

Method	Score	Collision	Progress
PDM-Closed [40]	0.9121	0.9701	0.9268
GameFormer [1]	0.8376	0.9473	0.8812
DTPP [8]	0.8243	0.9292	0.8323
PlanTF [41]	0.8366	0.9402	0.9267
Gen-Drive (Single)	0.8260	0.9160	0.8435
Gen-Drive (Multiple + PDM)	0.8087	0.9253	0.8587
Gen-Drive (Multiple + Learned)	0.8512	0.9365	0.8664
Gen-Drive (Fine-tuned, Single)	0.8633	0.9414	0.9012
Gen-Drive (Fine-tuned, Multiple)	0.8753	0.9572	0.8994
Gen-Drive (PDM fine-tuned, Single)	0.8550	0.9455	0.8655

Generation and evaluation outperforms single-sample **inference**. The multiple-sample planning method generates

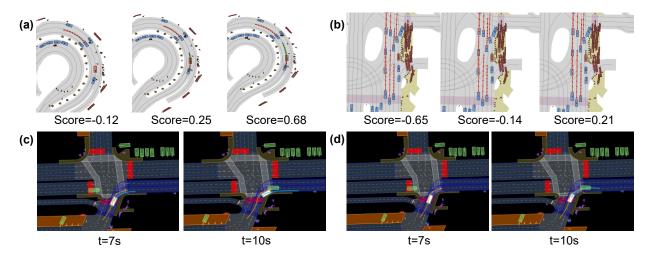


Fig. 5. Illustration of the planning process. (a) and (b) show two scenarios where our generative planner produces diverse plans and interaction outcomes. (c) and (d) compare the performance of the base policy with the fine-tuned policy, showing that the fine-tuned one learns to yield and avoid off-road plans.

16 scenes in parallel through batch processing and selects the optimal scenario using a learned reward model. This method enhances the diversity of generated plans, thereby improving the overall planning score. Additionally, the generative planner with our learned reward model outperforms the PDM score-based evaluator in planning. The average inference time for single-sample planning is 282.5 ms and 483.6 ms for multiple-sample planning on an RTX 4090 GPU, maintaining real-time performance requirements.

**Fine-tuning enhances performance**. Planning efficacy still largely depends on generation quality, and we demonstrate that RL fine-tuning can substantially enhance quality and performance. Notably, even with the single-sample method, the overall planning score of the fine-tuned policy outperforms the multiple-sample approach without fine-tuning. Moreover, using our learned reward model for fine-tuning performs better than using the PDM-based scorer.

Comparison with predictive methods. Compared to learning-based predictive planners (PlanTF, GameFormer, and DTPP), our model exhibits superior performance by using a generation-then-evaluation method. However, the PDM-Closed planner, which utilizes a rule-based trajectory generator and scorer, achieves the highest scores. It is important to note that it is optimized for nuPlan metrics, which may lack human likeness and adaptability to real-world scenarios.

### C. Ablation Studies

Influence of the number of modeling objects. We investigate the effect of the number of objects in the scene generator, ranging from 1 (where only the ego vehicle's plan is generated) to 100, and present the outcomes in Table II. We adjust the number of modeling objects in model training and employ a single-sample generation in testing. The results indicate that generating only the ego vehicle's plan results in inferior performance, primarily due to lack of movement in some cases. Conversely, an excessive number of modeling objects (e.g., 100) also results in diminished performance and runtime efficiency. Therefore, modeling 50 ego and surrounding objects performs best while maintaining runtime efficiency.

TABLE II INFLUENCE OF THE NUMBER OF MODELING OBJECTS

Number of objects	Score	Collision	Progress
1 (ego only)	0.8118	0.9384	0.7990
20	0.8188	0.9160	0.8383
50	0.8260	0.9160	0.8435
100	0.8079	0.9253	0.8166

Influence of number of fine-tuning steps. We examine the effect of training steps in the RL fine-tuning phase, employing a multiple-sample generation and scoring method in testing. The results shown in Table III reveal that 1000 fine-tuning steps achieve the best planning metrics, beyond which the performance of the fine-tuned policy tends to decline. This is a common issue in RLHF frameworks, as the policy might exploit the reward function and generate unreasonable behaviors. Therefore, we confine the RL fine-tuning phase to 1000 steps to prevent performance degradation.

TABLE III INFLUENCE OF THE NUMBER OF RL FINE-TUNING STEPS

Fine-tuning steps	Score	Collision	Progress
500	0.8441	0.9354	0.8459
1000	0.8753	0.9572	0.8994
1500	0.8457	0.9324	0.8431
2000	0.8123	0.9211	0.8345

### V. CONCLUSIONS

We propose the *Gen-Drive* framework, marking a paradigm shift to generation-evaluation in decision-making for autonomous driving. The framework integrates a behavior diffusion model as the core scene generator to model multiagent joint interactions, coupled with a scene evaluator learned using VLM-assisted preference data. Moreover, we apply RL fine-tuning to diffusion generation to further enhance its efficacy in planning tasks. Experimental results demonstrate the superior performance of our model with proper reward modeling compared to other learning-based planning methods and further enhancements through RL fine-tuning. Future research will aim to incorporate a perception module from sensor inputs within this generative framework to establish a fully end-to-end learnable driving system.

### REFERENCES

- 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 Interna*tional Conference on Computer Vision, 2023, pp. 3903–3913.
- [2] Y. Chen, S. Tonkens, and M. Pavone, "Categorical traffic transformer: Interpretable and diverse behavior prediction with tokenized latent," arXiv preprint arXiv:2311.18307, 2023.
- [3] Z. Huang, C. Tang, C. Lv, M. Tomizuka, and W. Zhan, "Learning online belief prediction for efficient pomdp planning in autonomous driving," *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 7023–7030, 2024.
- [4] Y. Chen, S. Veer, P. Karkus, and M. Pavone, "Interactive joint planning for autonomous vehicles," *IEEE Robotics and Automation Letters*, 2023.
- [5] 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 Sys*tems, 2023.
- [6] P. Karkus, B. Ivanovic, S. Mannor, and M. Pavone, "Diffstack: A differentiable and modular control stack for autonomous vehicles," in *Conference on robot learning*. PMLR, 2023, pp. 2170–2180.
- [7] Z. Huang, H. Liu, J. Wu, and C. Lv, "Conditional predictive behavior planning with inverse reinforcement learning for human-like autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, 2023.
- [8] Z. Huang, P. Karkus, B. Ivanovic, Y. Chen, M. Pavone, and C. Lv, "DTPP: Differentiable joint conditional prediction and cost evaluation for tree policy planning in autonomous driving," in 2024 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2024, pp. 6806–6812.
- [9] B. Ivanovic, A. Elhafsi, G. Rosman, A. Gaidon, and M. Pavone, "MATS: An interpretable trajectory forecasting representation for planning and control," arXiv preprint arXiv:2009.07517, 2020.
- [10] Z. Huang, Z. Zhang, A. Vaidya, Y. Chen, C. Lv, and J. F. Fisac, "Versatile scene-consistent traffic scenario generation as optimization with diffusion," arXiv preprint arXiv:2404.02524, 2024.
- [11] Z. Zhong, D. Rempe, D. Xu, Y. Chen, S. Veer, T. Che, B. Ray, and M. Pavone, "Guided conditional diffusion for controllable traffic simulation," in 2023 IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 3560–3566.
- [12] Z. Zhong, D. Rempe, Y. Chen, B. Ivanovic, Y. Cao, D. Xu, M. Pavone, and B. Ray, "Language-guided traffic simulation via scene-level diffusion," in *Conference on Robot Learning*. PMLR, 2023, pp. 144–177.
- [13] C. Jiang, A. Cornman, C. Park, B. Sapp, Y. Zhou, D. Anguelov, et al., "Motiondiffuser: Controllable multi-agent motion prediction using diffusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9644–9653.
- [14] C. Xu, D. Zhao, A. Sangiovanni-Vincentelli, and B. Li, "Diffscene: Diffusion-based safety-critical scenario generation for autonomous vehicles," in *The Second Workshop on New Frontiers in Adversarial Machine Learning*, 2023.
- [15] M. Uehara, Y. Zhao, T. Biancalani, and S. Levine, "Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review," arXiv preprint arXiv:2407.13734, 2024.
- [16] 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. 17863–17873.
- [17] Y. Cao, B. Ivanovic, C. Xiao, and M. Pavone, "Reinforcement learning with human feedback for realistic traffic simulation," arXiv preprint arXiv:2309.00709, 2023.
- [18] N. Karnchanachari, D. Geromichalos, K. S. Tan, N. Li, C. Eriksen, S. Yaghoubi, N. Mehdipour, G. Bernasconi, W. K. Fong, Y. Guo, et al., "Towards learning-based planning: The nuplan benchmark for real-world autonomous driving," arXiv preprint arXiv:2403.04133, 2024.
- [19] Z. Huang, X. Mo, and C. Lv, "Multi-modal motion prediction with transformer-based neural network for autonomous driving," in 2022 International Conference on Robotics and Automation (ICRA). IEEE, 2022, pp. 2605–2611.
- [20] A. Seff, B. Cera, D. Chen, M. Ng, A. Zhou, N. Nayakanti, K. S. Refaat, R. Al-Rfou, and B. Sapp, "Motionlm: Multi-agent motion forecasting as language modeling," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 8579–8590.

- [21] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [22] P. Tran, H. Wu, C. Yu, P. Cai, S. Zheng, and D. Hsu, "What truly matters in trajectory prediction for autonomous driving?" arXiv preprint arXiv:2306.15136, 2023.
- [23] S. Hagedorn, M. Hallgarten, M. Stoll, and A. Condurache, "Rethinking integration of prediction and planning in deep learning-based automated driving systems: a review," arXiv preprint arXiv:2308.05731, 2023
- [24] D. Dauner, M. Hallgarten, T. Li, X. Weng, Z. Huang, Z. Yang, H. Li, I. Gilitschenski, B. Ivanovic, M. Pavone, et al., "Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking," arXiv preprint arXiv:2406.15349, 2024.
- [25] Z. Huang, J. Wu, and C. Lv, "Driving behavior modeling using naturalistic human driving data with inverse reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, 2021.
- [26] T. Kaufmann, P. Weng, V. Bengs, and E. Hüllermeier, "A survey of reinforcement learning from human feedback," arXiv preprint arXiv:2312.14925, 2023.
- [27] Z. Zhou, H. Hu, X. Chen, J. Wang, N. Guan, K. Wu, Y.-H. Li, Y.-K. Huang, and C. J. Xue, "Behaviorgpt: Smart agent simulation for autonomous driving with next-patch prediction," arXiv preprint arXiv:2405.17372, 2024.
- [28] W. Wu, X. Feng, Z. Gao, and Y. Kan, "Smart: Scalable multiagent real-time simulation via next-token prediction," arXiv preprint arXiv:2405.15677, 2024.
- [29] Y. Hu, S. Chai, Z. Yang, J. Qian, K. Li, W. Shao, H. Zhang, W. Xu, and Q. Liu, "Solving motion planning tasks with a scalable generative model," arXiv preprint arXiv:2407.02797, 2024.
- [30] J. Philion, X. B. Peng, and S. Fidler, "Trajeglish: Learning the language of driving scenarios," arXiv preprint arXiv:2312.04535, 2023.
- [31] Z. Guo, X. Gao, J. Zhou, X. Cai, and B. Shi, "Scenedm: Scene-level multi-agent trajectory generation with consistent diffusion models," arXiv preprint arXiv:2311.15736, 2023.
- [32] M. Niedoba, J. Lavington, Y. Liu, V. Lioutas, J. Sefas, X. Liang, D. Green, S. Dabiri, B. Zwartsenberg, A. Scibior, et al., "A diffusionmodel of joint interactive navigation," Advances in Neural Information Processing Systems, vol. 36, 2024.
- [33] B. Yang, H. Su, N. Gkanatsios, T.-W. Ke, A. Jain, J. Schneider, and K. Fragkiadaki, "Diffusion-es: Gradient-free planning with diffusion for autonomous and instruction-guided driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 15 342–15 353.
- [34] Y. Zhang, E. Tzeng, Y. Du, and D. Kislyuk, "Large-scale reinforcement learning for diffusion models," arXiv preprint arXiv:2401.12244, 2024.
- [35] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine, "Training diffusion models with reinforcement learning," arXiv preprint arXiv:2305.13301, 2023.
- [36] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.
- [37] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International conference on machine learning*. PMLR, 2021, pp. 8162–8171.
- [38] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," arXiv preprint arXiv:2205.09991, 2022.
- [39] J. Abramson, A. Ahuja, F. Carnevale, P. Georgiev, A. Goldin, A. Hung, J. Landon, J. Lhotka, T. Lillicrap, A. Muldal, et al., "Improving multimodal interactive agents with reinforcement learning from human feedback," arXiv preprint arXiv:2211.11602, 2022.
- [40] D. Dauner, M. Hallgarten, A. Geiger, and K. Chitta, "Parting with misconceptions about learning-based vehicle motion planning," in *Conference on Robot Learning*. PMLR, 2023, pp. 1268–1281.
- [41] J. Cheng, Y. Chen, X. Mei, B. Yang, B. Li, and M. Liu, "Rethinking imitation-based planners for autonomous driving," in 2024 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2024, pp. 14123–14130.