MAPLE: Latent Multi-Agent Play for End-to-End Autonomous Driving

Rajeev Yasarla[1✉] **Deepti Hegde**[1⋆] **Hsin-Pai Cheng**[1⋆] **Shizhong Han**[1⋆]
Yunxiao Shi[1] **Meysam Sadeghigooghari**[2] **Hanno Ackermann**[1] **Litian Liu**[1]
Pranav Desai[2] **Fatih Porikli**[1] **Mohammad Ghavamzadeh**[1] **Hong Cai**[1✉]

[1]Qualcomm AI Research* [2]Qualcomm Technologies, Inc

✉{ryasarla,hongcai}@qti.qualcomm.com

⋆*Equal Contribution*

Abstract

Vision-language-action (VLA) models are effective as end-to-end motion planners, but can be brittle when evaluated in closed-loop settings due to being trained under traditional imitation learning framework. Existing closed-loop supervision approaches lack scalability and fail to completely model a reactive environment. We propose MAPLE, a novel framework for reactive, multi-agent rollout of a dynamic driving scenario in the latent space of the VLA model. The ego vehicle and nearby traffic agents are independently controlled over multi-step horizons, while being reactive to other agents in the scene, enabling closed-loop training. MAPLE consists of two training stages: (1) supervised fine-tuning on the latent rollouts based on ground-truth trajectories, followed by (2) reinforcement learning with global and agent-specific rewards that encourage safety, progress, and interaction realism. We further propose diversity rewards that encourage the model to generate planning behaviors that may not be present in logged driving data. Notably, our closed-loop training framework is scalable and does not require external simulators, which can be computationally expensive to run and have limited visual fidelity to the real-world. MAPLE achieves state-of-the-art driving performance on Bench2Drive and demonstrates scalable, closed-loop multi-agent play for robust E2E autonomous driving systems.

1 Introduction

End-to-end (E2E) autonomous driving has emerged as a promising paradigm that unifies perception, prediction, and planning into a single learned model. Recent E2E planners, such as UniAD [15] and VAD [23], demonstrate strong performance under supervised learning, but often struggle to generalize to long-tail and interactive driving scenarios. Vision-language-action (VLA) models further advance this line of work by grounding driving behavior in language and leveraging the reasoning capabilities of large multimodal models. Approaches such as GPT-Driver [33], EMMA [17], DriveVLM [45], and Senna [24] show improved robustness in complex driving situations, while ORION [10] and SimLingo [36] strengthen alignment between language and motion planning.

A central limitation of existing VLA planners is that training is primarily performed in an open-loop manner using supervised fine-tuning on large-scale logged driving data [10, 57, 36]. As a result, these models do not explicitly model closed-loop interactions between the ego vehicle and other traffic participants, and typically treat surrounding agents as non-reactive during training. This gap becomes critical at deployment time, where driving is inherently closed-loop and multi-agent,

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

requiring continuous adaptation to the evolving behavior of the other agents. Consequently, these models suffer from covariate shift, where small deviations from demonstrated trajectories compound over time and lead to planning failures [6, 26]. Additionally, ground-truth trajectory labels extracted from driving logs are very sparse, *i.e.*, they only capture a very small number of possible trajectories in the action space, which makes it challenging to learn a robust policy from them alone [26]. These limitations highlight the need for training mechanisms that explicitly support closed-loop, multi-agent interaction, ideally without relying on simulation pipelines that are costly and challenging to scale.

Reinforcement learning (RL) offers a natural way to perform closed-loop training by optimizing policies through interaction. Prior work has shown that large-scale RL can yield highly robust motion planners when trained with self-play and reactive agents in simulation [5, 18]. However, these approaches rely on external simulators that operate on symbolic representations, such as maps, bounding boxes, and vehicle states [13, 12, 43, 4, 5]. While these symbolic simulators enable scalable RL, they are not directly applicable to end-to-end VLA models that operate on raw sensory inputs. Extending RL to E2E planners requires simulating reactive agents and future observations at the pixel level, which is computationally expensive and difficult to scale using graphics engines, splatting methods, or diffusion-based image generation [8, 16]. Moreover, pixel-level simulators often exhibit limited visual fidelity and distribution mismatch with real-world data, further hindering transfer. These challenges motivate our approach of enabling RL directly in the latent space of a VLA, avoiding pixel-level simulation while still supporting closed-loop, multi-agent interaction learning.

Our goal is to enable this form of scalable closed-loop training for end-to-end motion planners. Towards this, we propose MAPLE, a framework that simulates the behavior of the ego-vehicle and multiple reactive agents in the latent space of a VLA model. This addresses the challenge of simulating sensory data while also training an E2E planner in a closed-loop environment in the presence of reactive agents. Diversity rewards encourage out-of-distribution behavior of surrounding agents at training time, providing richer feedback signals.

Our contributions are as follows.

- **Multi-agent scenario rollout in latent space across SFT and RL.** We introduce a simulator-free multi-agent rollout mechanism that jointly evolves the ego vehicle and surrounding reactive agents through autoregressive rollouts in the latent space of a VLA model. This enables closed-loop, multi-step agent interactions during training entirely in the latent space, significantly reducing computational cost compared to pixel-level training, while remaining removable at inference time without additional overhead.
- **Diversity-aware RL for multi-agent play.** To encourage heterogeneous and realistic interactions beyond logged driving data, we introduce a diversity-aware reward in the RL stage that explicitly promotes distinct agent behaviors. This leads to richer multi-agent interactions and more robust closed-loop driving policies.
- **State-of-the-art performance on closed-loop autonomous driving benchmarks.** Extensive experiments and ablation studies on the Bench2Drive benchmark show that the proposed multi-agent rollout mechanism and diversity-aware RL significantly improve driving performance, boosting the driving score by 25+%, setting the new state-of-the-art, and enhancing robustness across complex multi-ability driving scenarios.

2 Related Work

End-to-end and VLA models for autonomous driving. Early E2E planners such as UniAD [15] and VAD [23] unify perception, prediction, and planning in a single pipeline but struggle with long-tail generalization. VLA models address this by grounding motion in language. GPT-Driver [33] casts planning as text generation with chain-of-thought reasoning [47]. EMMA [17] scales this to large multi-modal corpora. DriveVLM [45] and Senna [24] pair a VLM with a conventional planner or meta-action head. ORION [10] and SimLingo [36] further tighten the language-trajectory alignment. ReCogDrive [29] and DiffRefiner [52] improve closed-loop robustness through recognition-driven planning and diffusion-based refinement, yet still rely on static logged trajectories for training. GeRo [51] introduces language-conditioned latent rollouts with ego planning and surrounding-agent motion prediction. Despite these advancements, all these methods treat surrounding agents as non-reactive, leaving closed-loop multi-agent dynamics underexplored. MAPLE closes this gap by enabling reactive co-planning of ego vehicle and nearby agents entirely within a latent token space.

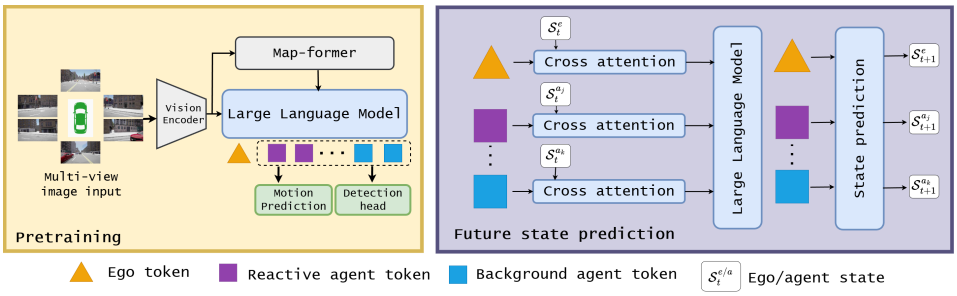

Figure 1: **MAPLE pretraining and future state prediction. Left:** Pretraining the VLA backbone with auxiliary supervision (e.g., map learning, detection, and motion prediction). **Right:** State-transition pretraining that predicts next-step ego/agent states over a horizon T to stabilize the token space.

Multi-agent simulation and self-play. Trajectory forecasting methods [14, 38, 56] model joint agent futures from fixed observations but do not support interactive planning. Self-play has produced strong policies in games [39, 46] and manipulation [1]. Recently, self-play has been applied to autonomous driving to obtain robust policies [5]. However, this is limited to planning in the symbolic space with bounding boxes, maps, and vehicle states. Adversarial scenario generation [35, 53] synthesizes safety-critical behaviors but still relies on symbolic planners, not learned VLA policies. Extending self-play to E2E planners requires simulating future observations at the pixel level. Graphics engines [8], Gaussian splatting, and diffusion-based generators [16] can produce such inputs but suffer from high computational cost, limited visual fidelity, and poor scalability for interactive multi-agent rollouts. MAPLE provides a novel framework that enables multi-agent interactive play directly in the latent space of a VLA model, requiring no external simulator or image rendering/generation.

Diverse action generation and RL-based policy optimization. Diffusion-based planners, including DiffusionDrive [30], DiffusionPlanner [55], and GoalFlow [50], capture multimodal trajectory distributions through stochastic sampling but do not explicitly encourage behavioral diversity during training, leaving them prone to mode collapse under RL fine-tuning [27]. GRPO [37] has been adapted for driving in AlphaDrive [25], R2SE [31], and TrajHF [28], improving policy stability via rule-based rewards, though each assigns a single planner per agent. Sun et al. [41] use latent token sequences and diversity-seeking rewards, focusing more on reasoning and not closed-loop driving. GenDrive [16] adds reward modeling to a diffusion policy but still relies on a simulator. MAPLE addresses both gaps: it assigns multiple discrete planners per agent and penalizes behavioral overlap with diversity-aware GRPO rewards, preventing collapse and generating rich, safety-critical long-tail scenarios without any external simulator.

3 Proposed Approach

We present MAPLE, a novel framework for end-to-end motion planning that performs closed-loop multi-agent rollout in the latent space of a VLA model. The ego vehicle and neighboring agents are represented as compact latent tokens, encoding velocity, acceleration, location, map labels, and traffic status. Given a latent ego vehicle token, latent agent tokens, and a high-level scenario description, MAPLE rolls out future scenarios by predicting future tokens in an autoregressive manner. This allows us to model the interaction of agents in the scene. Action planning and motion prediction heads are supervised with a series of rewards to encourage safe and diverse driving behavior. An overview of the multi-agent rollout mechanism in our proposed MAPLE framework can be seen in Figures 2 and 3.

3.1 Problem Setup

Given multi-view camera images up to time t, our goal is to predict a safe ego-vehicle trajectory $\tau^e_{t:t+T}$, where T is the planning horizon, while accounting for the future evolution of surrounding traffic participants. Unlike ego-only planners that model the behavior of the neighboring agents with ground-truth trajectory playback, we perform decision-making with *reactive* agents, where the ego vehicle's plan is conditioned on the anticipated behavior of agents controlled by learned policies.

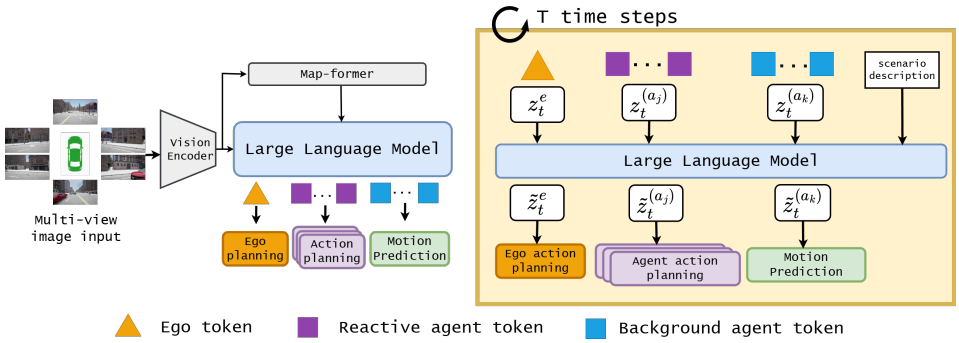

Figure 2: **MAPLE supervised fine-tuning (SFT) stage. Left:** Single-step supervision and inference. The VLA backbone encodes multi-view images (and map features) into ego and agent tokens, which are decoded by an ego planner, reactive-agent planners, and a motion head. **Right:** The same model unrolled for T steps during imitation-learning-based scenario rollouts. Predicted tokens/trajectories are fed back autoregressively to supervise ego and reactive-agent planning over the horizon while forecasting background-agent motion.

We divide the agents into (i) a set of *reactive agents* ($\mathbf{Z}_t^{\text{react}}$) that are likely to interact with the ego vehicle, and (ii) *background agents* ($\mathbf{Z}_t^{\text{bg}}$), whose behaviors are less likely to influence the ego vehicle. The future states of the ego and reactive agents are rolled out in an autoregressive manner, while the future trajectories of background agents are directly predicted through regression.

3.2 MAPLE Architecture

MAPLE is a closed-loop training framework for VLA models. The framework consists of a vision encoder, a map encoder, an LLM, and several task-specific heads related to object detection, map segmentation, motion prediction, and action planning. The action planning head is a variational auto-encoder (VAE) that decodes ego and reactive agent tokens into their respective trajectory waypoints [10]. The motion prediction head is a multi-layer perceptron that regresses future trajectories from background agent tokens. An overview of the architecture can be seen in Figure 2.

3.3 Pretraining

In order to learn a stable tokenization of ego and agent dynamics for consistent autoregressive scenario generation, the VLA model is first pretrained to perform the auxiliary tasks of 3D object detection, map segmentation, agent motion prediction, and traffic light segmentation. Detailed definitions of all auxiliary pretraining losses and their weighting are provided in Appendix G.1. These tasks encourage the shared backbone to encode geometry, semantics, and traffic context into latent agent tokens that can be reliably decoded over time. The model is also trained to predict the future states of the ego vehicle and the surrounding agents. An overview of the MAPLE future state prediction is shown in Figure 1.

Future state prediction. The state of a traffic participant at time t is represented by four ordered tokens: $S_t = \big\langle \texttt{<DYN>}, \texttt{<TYPE>}, \texttt{<MS>}, \texttt{<TS>} \big\rangle_t$, where <DYN> encodes agent dynamics (e.g., position, heading, velocity, acceleration), <TYPE> is a categorical token in $\{\texttt{car}, \texttt{truck}, \texttt{pedestrian}, \texttt{cyclist}\}$, <MS> is the index of the map segment where the agent resides, and <TS> encodes traffic light or traffic status information related to the agent and associated to the selected map context. Given the current state S_t (and the corresponding ego/agent token from VLA encoder), the model predicts the next state $\hat{S}_{t+1}$.

We define the future state prediction loss $\mathcal{L}_{\text{state}}$ as a weighted combination of (i) cross-entropy losses for the discrete state labels <MS> and <TS>, and (ii) an ℓ_1 regression loss for the continuous dynamics token <DYN>:

$$\mathcal{L}_{\text{state}} = \lambda_{\text{dyn}} \big\| \widehat{\texttt{<DYN>}} - \texttt{<DYN>} \big\|_1 + \lambda_{\text{ms}} \, \text{CE}\big(\widehat{\texttt{<MS>}}, \texttt{<MS>}\big) + \lambda_{\text{ts}} \, \text{CE}\big(\widehat{\texttt{<TS>}}, \texttt{<TS>}\big), \tag{1}$$

where $\widehat{\cdot}$ and $\text{CE}(\cdot, \cdot)$ denote model predictions and cross-entropy loss, and λ_{dyn}, λ_{ms}, and λ_{ts} are the weights for the loss terms. The overall pretraining objective consists of minimizing the sum of the respective losses on 3D object detection, map segmentation, motion prediction, and future state prediction.

3.4 MAPLE Rollout and Training Framework

The closed-loop framework enables language-conditioned scenario rollouts for the ego vehicle and a subset of *reactive* agents. The objective is to learn a compact token space in which the VLA backbone can roll out future traffic states in an autoregressive manner and support behavior-aware action planning over a horizon T. Compared to ego-only planners, MAPLE explicitly models multi-step interactions by (i) jointly rolling out ego and reactive-agent states, and (ii) encouraging diverse driving behaviors via safety and diversity rewards.

3.4.1 Multi-Agent Rollout and Supervised Fine-Tuning

Given a set of historical and current multi-view images at time t and a high-level scenario description, MAPLE encodes the scene into a latent ego token z_t^e , a set of latent reactive agent tokens $\mathbf{Z}_t^{\text{react}}$ indexed by a_j, and a set of latent background agent tokens $\mathbf{Z}_t^{\text{bg}}$ indexed by a_k (see Figure 2). Auto-regressive rollout is performed for T steps. At each rollout step $\Delta = \{0, \dots, T-1\}$, the VLA model predicts the next-step tokens $\tilde{z}_{t+\Delta+1}$ conditioned on the current tokens $\tilde{z}_{t+\Delta}$ and the scenario description. Here $\tilde{z}_t$ denotes latent tokens generated by auto-regressive rollout, distinguished from ground-truth tokens z_t obtained from logged data.

The predicted tokens are decoded into trajectories and fed back as input for the next rollout step, yielding a rollout from t to $t+T$. The planning heads produce waypoint trajectories for the ego vehicle and reactive agents: the ego planner P^e decodes $\tilde{z}^e_{t+\Delta+1}$ into trajectory $\tilde{\tau}^e_{t+\Delta+1}$, and agent planners $\{P^{(a_j)}\}_{a_j \in \mathbf{Z}_t^{\text{react}}}$ decode reactive-agent tokens into trajectories $\{\tilde{\tau}^{(a_j)}_{t+\Delta+1}\}$. Let $\mathcal{P} = \{P^e\} \cup \{P^j\}_{j=1}^{N_R}$ denote the set of action planners, where N_R denotes the number of reactive-agent planners. At the start of each rollout, we use P^e as the ego action planner, and for each reactive agent we select an action planner based on the agent's behavior.

The motion head predicts future trajectories for background agents indexed by a_k. Background agents participate in the latent rollout, however, they are *not controlled* by planning heads. Instead, their trajectories $\{\tilde{\tau}^{(a_k)}_{t+\Delta+1}\}$ are generated by a motion prediction head conditioned on the current scene tokens. This design models passive scene dynamics while reserving explicit decision-making for the ego vehicle and reactive agents.

Supervised Fine-Tuning (SFT). In the SFT stage, we perform imitation-learning-based scenario rollouts for T steps, supervising the ego vehicle and a set of reactive agents, while predicting motion for background agents. We supervise each step using ground-truth trajectories, providing accurate targets for the rollout horizon. The SFT objective loss is defined as the sum of per-step planning and motion prediction losses over the rollout horizon.

$$\mathcal{L}_{\text{SFT}} = \sum_{\Delta=1}^{T} \left[\mathcal{L}_{\text{plan}}\big(\tilde{\tau}^e_{t+\Delta}, \tau^e_{t+\Delta}\big) + \sum_{a_j \in \mathbf{Z}_t^{\text{react}}} \mathcal{L}_{\text{plan}}\big(\tilde{\tau}^{(a_j)}_{t+\Delta}, \tau^{(a_j)}_{t+\Delta}\big) + \sum_{a_k \in \mathbf{Z}_t^{\text{bg}}} \mathcal{L}_{\text{mot}}\big(\tilde{\tau}^{(a_k)}_{t+\Delta}, \tau^{(a_k)}_{t+\Delta}\big) \right]. \quad (2)$$

Here, by $\mathcal{L}_{\text{plan}}(\cdot,\cdot)$, we denote the planning loss used for supervising ego and reactive-agent trajectories, and by $\mathcal{L}_{\text{mot}}(\cdot,\cdot)$, the motion prediction loss applied to background agents. The detailed formulation of the planning loss is provided in Appendix G.2.

3.4.2 Post-Training Reinforcement Learning

The main objective of MAPLE is to learn *diverse* and *robust* driving behaviors while performing multi-step scenario rollouts that induce a wide range of environmental dynamics and ego–agent interactions. In such settings, expert demonstrations are often multi-modal (e.g., brake vs. yield vs. merge), and pure imitation learning can be sub-optimal or even conflictual under distribution shift. Thus, relying solely on SFT (Eq. 2) is insufficient to handle long-tail events and novel reactive interactions. To mitigate these issues and encourage safe, high-fidelity, and behaviorally diverse rollouts, we introduce an RL fine-tuning stage using the Group Relative Policy Optimization (GRPO) algorithm [37] with a structured reward design.

Reward structure. At each rollout step $t + \Delta$, we compute three types of rewards: (i) a **global reward** that captures scene-level safety and stability, (ii) a **vehicle-specific reward** that encourages

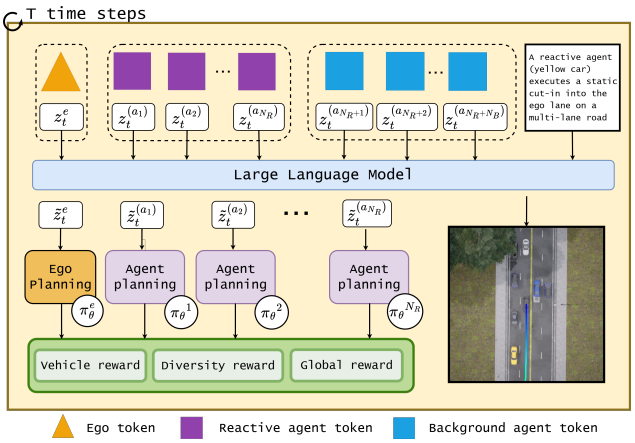

Figure 3: **MAPLE RL fine-tuning stage.** Starting from the SFT model, we optimize multi-step rollouts over T steps using RL with safety-aware and interaction-aware rewards (e.g., collision avoidance and TTC).

progress and safe driving for each controlled agent, and (iii) a **diversity reward** that promotes distinct behaviors across different planners/policies.

At a time t, we define the total rollout reward as

$$R_t = G_t + D_t + r_t^e + \sum_{a_j \in \mathbf{Z}^{\text{react}}} r_t^{a_j}, \tag{3}$$

where G_t and D_t are global and diversity rewards, r_t^e is the vehicle-specific reward for the ego vehicle, and $r_t^{a_j}$ is the vehicle-specific reward for the reactive agent $a_j \in \mathbf{Z}^{\text{react}}$.

Global reward. The global reward encourages long collision-free rollouts and penalizes scene-level infractions. Let l be the number of steps completed without collision (early termination yields smaller l) in a T-step rollout. We define the global reward G_t at time t as

$$G_t = \frac{l}{T} - \sum_{\Delta=1}^{T} \text{Col}\big(\tilde{\tau}_{t+\Delta}^{e}\big) - \sum_{a_j \in \mathbf{Z}^{\text{react}}} \sum_{\Delta=1}^{T} \text{Col}\big(\tilde{\tau}_{t+\Delta}^{a_j}\big), \tag{4}$$

where $\text{Col}(\cdot)$ is the collision penalty calculated from the predicted trajectory $\tilde{\tau}_{t+\Delta}$.

Vehicle-specific reward. Vehicle-specific rewards promote route completion/progress while discouraging unsafe behaviors, such as low time-to-collision (TTC). We use a step-wise reward and sum it over the rollout horizon T:

$$r_t = \sum_{\Delta=1}^{T} \Big[\underbrace{RC_{t+\Delta}}_{\text{route completion}} - \underbrace{\phi(\text{TTC}_{t+\Delta})}_{\text{TTC penalty}} - \underbrace{\mathcal{L}_{\text{prog}}(\tilde{\tau}_{t+\Delta})}_{\text{progress loss}} \Big]. \tag{5}$$

Here $RC_{t+\Delta}$ measures incremental progress along the reference route, $\mathcal{L}_{\text{prog}}$ penalizes lack of forward progress or deviation from a desired trajectory, and $\phi(\text{TTC})$ is a monotone penalty function that is activated when TTC falls below a safety threshold, e.g., $\phi(\text{TTC}) = \max(0, \text{TTC}_{\max} - \text{TTC})$.

Diversity reward. Safety and progress objectives can favor similar policies, so multiple planning heads alone do not ensure heterogeneous agent responses. The diversity reward D_t discourages behavioral collapse by rewarding separation in driving-quality descriptors. This exposes the ego policy to a wider range of nearby-agent responses during rollout, helping it to learn decisions that accommodate different yielding, compliance, and comfort behaviors. Global collision and vehicle-specific TTC/progress rewards provide complementary safety and task feedback.

We characterize the behavior of the ego and each of the N_R reactive agents $a_j \in \mathbf{Z}^{\text{react}}$ using a compact descriptor formed from NAVSIM driving-quality metrics [7], which captures salient properties, such as mean acceleration, jerk, minimum time-to-collision (TTC), lane centering error, and lane-change timing. We denote these compact descriptors by $\Gamma^e(\cdot)$, for the ego, and by $\{\Gamma^{a_j}(\cdot)\}_{j=1}^{N_R}$, for the reactive agents. Note that if two agents belong to the same behavior category, then they will have the same compact descriptor.

At the start of the rollouts, the ego vehicle is controlled by a policy/planner π_θ^e, and the reactive agents are controlled by policies $\{\pi_\theta^{a_j}\}_{j=1}^{N_R}$. For each of these agents, we generate a rollout/trajectory over the horizon T using its own policy, i.e., $\{\tilde{\tau}_{t+\Delta}^{a_j}\}_{\Delta=1}^{T} \sim \pi_\theta^{a_j}$. The diversity reward for the ego (and similarly for each reactive agent) is computed as

$$D_t = \frac{1}{N_R} \sum_{j=1}^{N_R} \sum_{\Delta=1}^{T} \left\| \Gamma^e\big(\tilde{\tau}_{t+\Delta}^{e}\big) - \Gamma^{a_j}\big(\tilde{\tau}_{t+\Delta}^{a_j}\big) \right\|_1, \tag{6}$$

which assigns higher rewards when different planner selections induce measurably distinct (yet safe) behaviors. Additional details on the descriptor $\Gamma(\cdot)$, behavior category, and the diversity reward are provided in Appendix B.

GRPO objective. Given a group of G rollouts at time t with returns $\{R_t^{(g)}\}_{g=1}^{G}$, we compute a group-relative baseline $b_t = \frac{1}{G}\sum_{g=1}^{G} R_t^{(g)}$ and advantages $A_t^{(g)} = R_t^{(g)} - b_t$. We then optimize the policy using the GRPO loss:

$$\mathcal{L}_{\text{RL}} = -\frac{1}{G} \sum_{g=1}^{G} \min\Big(r_\theta^{(g)} A_t^{(g)},\ \text{clip}\big(r_\theta^{(g)}, 1-\epsilon, 1+\epsilon\big) A_t^{(g)} \Big), \tag{7}$$

where $r_\theta^{(g)} = \frac{\pi_\theta\left(\{\tilde{\tau}_{t+\Delta}^{(g)}\}\right)}{\pi_{\theta_{\text{old}}}\left(\{\tilde{\tau}_{t+\Delta}^{(g)}\}\right)}$ and the clip parameter is set to $\epsilon = 0.1$.

4 Experimental Setup

Datasets. We conduct both training and evaluation of MAPLE using the Bench2Drive benchmark [21], a closed-loop end-to-end autonomous driving suite built on the CARLA simulator [8]. Following the official benchmark setup, the dataset consists of 1,000 driving clips, among which 950 are used for training and 50 are reserved for open-loop validation. Each clip spans roughly 150 meters and captures a wide range of traffic conditions and interaction patterns. Closed-loop performance is assessed using the standard Bench2Drive protocol, which evaluates agents over 220 routes covering 44 diverse interactive scenarios. We additionally evaluate on the in-distribution and generalization splits of Fail2Drive [11] for closed-loop driving under distribution shifts.

Evaluation Metrics. On Bench2Drive, we adopt the official closed-loop evaluation criteria, including Driving Score (DS), Success Rate (SR), Efficiency, Comfort, and Multi-Ability [21]. Driving Score reflects overall route completion while accounting for traffic violations. Success Rate reports the percentage of routes finished without failure. Efficiency and Comfort quantify the agent's driving speed and smoothness, respectively. The Multi-Ability metric further measures performance across five complex urban driving behaviors.

Model Architecture. We instantiate MAPLE on the multimodal LLM backbone Qwen2.5-1.5B-VL [2] for end-to-end multi-agent planning. The Qwen2.5 text encoder and LLM generate scenario-level text tokens and ego/agent-specific latent tokens. For visual inputs, MAPLE employs an EVA-pretrained vision transformer (ViT) [9] with a Q-Former module [10] to extract image tokens from multi-view observations. The detection and motion prediction heads each consist of a multi-head attention module followed by three MLP layers. The future state prediction head adopts the same architecture. Each action-planning head in the planner set $\mathcal{P}$ is implemented as a variational autoencoder (VAE), following existing VLA planning heads [10]. Unless stated otherwise, MAPLE predicts trajectories conditioned only on Navigation Command (NC), without explicit lane-center targets, such as Target Point (TP). MAPLE adopts an anchor-free design and predicts six trajectory modes aligned with Bench2Drive. Scenario rollouts are used *only during training* and are disabled at inference time.

Training. All experiments are conducted on 8 NVIDIA H100 GPUs. MAPLE is trained in three stages. (1) Pretraining: 18 epochs with auxiliary supervision, including detection, map segmentation, motion prediction, and future state prediction, to learn robust latent token representations. (2) Supervised Fine-Tuning (SFT): 24 epochs of scenario rollouts with imitation learning. Autoregressive rollouts take T steps, supervising ego-vehicle and reactive-agent planning and background-agent motion prediction. (3) Post-Training Reinforcement Learning: 6 epochs using the proposed reward

Table 1: Comparison of closed-loop planning and multi-ability performance on the Bench2Drive. Avg. L2 denotes the average trajectory error over 2 seconds at 2 Hz. Here Cond. is the applied condition of NC = Navigation Command or TP = Target Point. † indicates models trained with geometric path waypoint supervision and evaluated using the PID controller from SimLingo.

Method	Cond.	Closed-Loop Metrics				Multi-Ability Test (%) ↑					
		DS↑	SR (%)↑	Efficiency↑	Comfort↑	Merging	Overtaking	E-Brake	Give Way	T.Sign	Mean
TCP-traj [49]	TP	59.9	30.0	76.5	18.1	28.8	24.3	51.7	40.0	46.3	34.2
ThinkTwice [20]	TP	62.4	31.2	69.3	16.2	27.4	18.4	35.8	50.0	54.2	37.2
DriveAdapter [19]	TP	64.2	33.1	70.2	16.0	28.4	28.4	47.5	50.0	56.4	42.1
SimLingo† [36]	TP	85.1	67.3	**259.2**	33.7	54.0	57.0	88.3	53.3	82.5	67.0
UniAD-Base [15]	NC	45.8	16.4	129.2	43.6	8.9	9.3	20.0	20.0	14.2	14.5
VAD [23]	NC	42.4	15.0	157.9	46.0	11.4	11.4	18.6	20.0	19.2	18.1
MomAD [40]	NC	44.5	16.7	170.2	**48.6**	-	-	-	-	-	-
DriveTransformer-L [22]	NC	63.5	35.0	100.6	20.8	17.6	35.0	48.4	40.0	52.1	38.6
HiP-AD [44]†	NC	86.8	69.1	203.1	19.4	50.0	84.4	83.3	40.0	50.5	65.9
Qwen2.5 [3]	NC	63.9	31.6	119.3	10.1	14.3	28.9	30.1	30.0	24.7	25.6
ORION [10]	NC	77.7	54.6	151.5	17.4	25.0	71.1	78.3	30.0	69.2	54.7
ReCogDrive [29]	NC	71.4	45.5	138.2	17.5	29.7	20.0	69.1	20.0	71.3	42.0
GeRo [51]	NC	81.9	60.1	176.5	40.2	40.1	78.2	87.3	50.0	76.8	66.5
MAPLE (ours)	NC	85.2	67.5	184.3	37.9	46.3	80.7	88.1	60.0	78.1	70.6
MAPLE † (ours)	NC	**88.3**	**70.5**	210.3	39.0	49.4	**86.2**	**90.0**	**60.0**	80.4	**73.2**

mechanism (Section 3.4.2), with T-step autoregressive rollouts optimizing ego and reactive-agent planning via agent-specific, global, and diversity rewards. We use 4 input frames, set $N_R = 8$ for reactive-agent planners, and use a rollout horizon of $T = 8$. To increase temporal coverage, we sample rollouts with multiple temporal strides corresponding to effective rates of 0.5Hz, 1Hz, 1.5Hz, 2Hz, 5Hz, and 10Hz, covering up to 12 seconds into the future. All Qwen2.5-1.5B parameters are updated across stages. Optimization uses AdamW with cosine annealing, a learning rate of 2×10^{-4}, and weight decay 0.01.

5 Results

We perform closed-loop evaluation and multi-ability test of MAPLE on Bench2Drive and Fail2Drive, compare with recent state-of-the-art methods, and evaluate the latent rollout realism of MAPLE. We further conduct ablation studies on different aspects of our proposed approach. More results can be found in the appendix, including inference computation analysis (Section C), open-loop evaluation (Section D), and additional qualitative visualization results and comparison (Section F).

5.1 Closed-Loop Evaluation

Quantitative Results. Table 1 summarizes quantitative comparisons with state-of-the-art methods on Bench2Drive, including closed-loop evaluation over 220 test routes and multi-ability assessments across five challenging urban driving behaviors: merging, overtaking, emergency braking, giving way, and traffic sign handling. Our method outperforms all existing approaches and achieves state-of-the-art performance across key metrics. Notably, the diverse multi-agent driving behaviors enabled by MAPLE lead to consistent improvements on most multi-ability metrics, highlighting its robustness and effectiveness in complex interactive driving scenarios. We also report open-loop evaluation results in Appendix Section D.

Standard MAPLE uses temporal waypoint supervision at 0.5 s intervals and the Bench2Drive controller. MAPLE † adds geometric path waypoints at 2 m spacing alongside temporal waypoints and uses the SimLingo PID controller during closed-loop evaluation. This variant aligns waypoint supervision and controller choices with SimLingo and HiP-AD while using the original Bench2Drive 950 training videos. Note that SimLingo trains on a different, larger set of driving scenarios (approximately 650K samples) and a stronger PDM-Lite expert; these training-data differences remain.

Qualitative Results. Figure 4 illustrates the closed-loop performance of the proposed MAPLE across diverse environments and interaction scenarios. In adverse weather conditions with reduced visibility (top row), the policy demonstrates cautious and stable behavior by smoothly reducing speed and applying mild braking when encountering pedestrians and roadside obstacles. In clear suburban settings with dynamic agents such as vehicles, pedestrians, and cyclists (bottom row), MAPLE adapts its planned trajectory to safely yield, overtake, and negotiate interactions through gradual steering and acceleration adjustments. Across all scenarios, the planned ego trajectories remain smooth and consistent, indicating coherent action selection and stable closed-loop control under varying environmental conditions. Additional qualitative results can be found in Appendix Section F.

Table 2: Closed-loop evaluation on Fail2Drive, in terms of Driving Score (DS) and Success Rate (SR, in %); both are higher the better. † uses geometric path supervision and the SimLingo PID controller.

Method	In-distribution		Generalization	
	DS↑	SR↑	DS↑	SR↑
ORION	53.0	52.0	51.2	46.0
SimLingo	80.6	79.3	71.7	55.0
HiP-AD	74.1	70.7	67.1	56.7
MAPLE	85.3	82.1	76.1	63.7
MAPLE †	86.7	82.9	79.5	65.8

Table 3: Realism of latent rollouts on Bench2Drive using adapted Waymo Sim Agents metrics. Higher values indicate better agreement with the reference distribution.

Method	Meta	Kinematic	Interactive
ORION [10]	0.391	0.263	0.518
SimLingo [36]	0.421	0.368	0.573
DeepSight [54]	0.490	0.413	0.608
GeRo [51]	0.553	0.454	0.620
MAPLE	0.572	0.476	0.703

Generalization on Fail2Drive Fail2Drive [11] tests closed-loop generalization using paired CARLA routes: each route containing a distribution shift is matched with an in-distribution counterpart. Its 17 new scenario classes cover appearance, layout, behavioral, and robustness shifts. Reporting both splits therefore examines adaptation to changed driving conditions beyond the standard Bench2Drive scenarios. Table 2 shows that MAPLE maintains 85.3 DS on the in-distribution split, close to its 85.2 DS on Bench2Drive (Table 1). On the generalization split, despite trained on less data, MAPLE reaches 76.1 DS and 63.7% SR, outperforming SimLingo by 4.4 DS and 8.7 SR points and HiP-AD by 9.0 DS and 7.0 SR points. MAPLE † further improves these scores to 79.5/65.8. Nevertheless, MAPLE's SR falls from 82.1% to 63.7% across the two splits, showing that substantial generalization challenges remain.

Latent Rollout Realism Closed-loop driving scores measure the deployed policy's performance, but do not directly assess the fidelity of the latent futures used during training. We therefore evaluate decoded rollout trajectories using metrics adapted from the Waymo Open Sim Agents Challenge [34], with ground-truth Bench2Drive trajectories as the reference distribution. All methods are evaluated with the same protocol using their best checkpoint; DeepSight [54] provides a latent world-modeling comparison.

The kinematic metric averages likelihoods of linear speed, linear acceleration, angular speed, and angular acceleration, assessing agreement with reference motion statistics. The interactive metric averages likelihoods of distance to the nearest object, collision indication, and time-to-collision, assessing proximity and collision-risk statistics. The realism meta score linearly aggregates these measures. We omit the map component because distance-to-center values are unavailable for some methods. Thus, the evaluation measures trajectory-distribution agreement rather than latent-token reconstruction or exact matching of a single future.

Table 3 shows gains in both kinematic and interactive agreement, with MAPLE reaching 0.476 and 0.703, respectively, and a meta score of 0.572. The interactive improvement is particularly relevant because surrounding-agent behavior shapes the safety and progress feedback used in rollout RL. These results complement the driving evaluation by checking the generated trajectories from the latent model. Appendix E.1 gives further evaluation protocol details and discussion.

5.2 Ablation Study

Ego-Agent Rollout Setting. We perform an ablation study on Bench2Drive closed-loop evaluation to analyze the impact of ego-agent rollout at different training stages as shown in Table 4. The baseline model does not incorporate future state estimation or rollout. Introducing single-step future state estimation during pretraining improves Driving Score (DS) by 2.3 points over the baseline. Enabling multi-step ego-agent rollout yields further gains, outperforming single-step estimation by 4.2 points in DS. Incorporating rollout during supervised fine-tuning (SFT) leads to additional improvements, and the best performance is achieved when rollout is applied consistently across pretraining, SFT, and reinforcement learning (RL) stages. Compared to the baseline, this full rollout strategy improves DS by 12.3 points and increases the success rate by 7.9

Single- vs. Multi-Planner As shown in Table 5, we ablate on single-planner, multi-planner, and multi-planner with diverse action modeling under different rollout settings. Under the ego-only setting, rollout is applied only to the ego agent, while performance is further improved by adopting a multi-planner strategy. During inference, the final planner is selected based on the reward likelihood

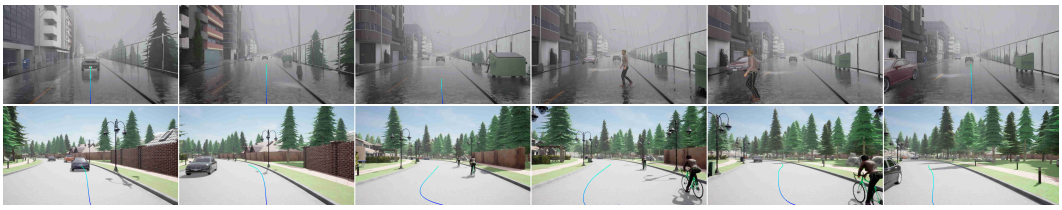

Figure 4: **Qualitative examples of closed-loop driving on Bench2Drive using MAPLE.** We show representative trajectories in diverse scenarios, including adverse-weather scenes with limited visibility and sudden pedestrian crossings (top row), and clear suburban traffic with dynamic agents such as cyclists and surrounding vehicles (bottom row). Blue curves denote the planned ego-vehicle trajectory, highlighting smooth and adaptive behavior during interactive maneuvers.

Table 4: Closed-loop ablation study of ego-only rollout at different training stages on Bench2Drive. *Roll* indicates whether ego-agent future rollout is enabled during training. SFT and RL denote supervised fine-tuning and reinforcement learning.

Method	Ego	$\mathcal{A}^{\text{react}}$	Roll	DS ↑	SR (%) ↑
Baseline	✓	✗	✗	68.1	51.9
+ state estimation pretraining	✓	✗	✗	70.4	54.3
+ rollout SFT	✓	✗	✓	77.5	57.4
+ rollout RL for ego-only	✓	✗	✓	**80.4**	**59.8**

Table 5: Closed-loop ablation study of rollout-based action planning strategies on Bench2Drive. Single: all agents share one planner. Multi: agents are assigned distinct planners.

Rollout Type	Planner	Diversity Reg.	Reactive Agents	DS ↑	SR (%) ↑
Ego Only	Single	✗	✗	80.4	59.8
	Multi	✗	✗	81.8	61.5
	Multi	✓	✗	82.6	62.7
Self-Play	Single	✗	✓	82.1	62.0
	Single	✓	✓	84.1	63.9
Multi-Agent Play	Multi	✗	✓	83.4	66.9
	Multi	✓	✓	**85.2**	**67.5**

learned during the reinforcement learning stage. The self-play setting applies rollout to both the ego and reactive agents using a single shared planner P^e. Multi-agent play extends this design by assigning different planners to different reactive agents, drawing from a pool $\{P_k\}_{k=1}^{N_R}$. The results consistently show that multi-planner approaches achieve higher Driving Scores than single-planner baselines. Moreover, multi-planner configurations with diverse action modeling outperform those that share a single planner across agents, and enabling reactive-agent rollout further improves performance compared to ego-only training. Most importantly, the multi-planner setup with diverse action modeling achieves the best overall results, improving Driving Score by an additional 4.8 points and Success Rate by 7.7 points.

Additional Results. We have conducted more ablation studies to analyze the impact of several design choices, including the number of reactive agents, reward components, and supervised loss terms in SFT. which can be found in Appendix Section A.

6 Conclusion

We introduced MAPLE, a simulator-free multi-agent training framework that enables closed-loop, interactive planning through latent-space rollouts in vision–language–action models. By extending rollouts to reactive agents and incorporating diversity-aware reinforcement learning, MAPLE generates richer interactions and improves robustness in data-scarce settings. Extensive experiments on Bench2Drive demonstrate state-of-the-art closed-loop performance, highlighting MAPLE as a scalable and effective approach for advancing end-to-end autonomous driving.

Limitations. Despite its strong performance, MAPLE has a few limitations. First, multi-agent interactions are modeled through latent rollouts rather than physical simulation. This enables fast and scalable training but can potentially have limited fidelity in scenarios that requires very high-precision dynamics. Second, our experiments were primarily conducted on Bench2Drive. As part of future work, we will extend our experiments to real-world data when robust, accurate real-data-based closed-loop simulators become available.

Broader Impact. MAPLE provides a simulator-free, multi-agent closed-loop training framework that improves the robustness of end-to-end autonomous driving models. Our work can facilitate the development of autonomous driving systems that have improved safety, adaptability, and performance in complex driving scenarios, particularly in data-scarce or long-tail situations.

References

[1] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[2] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.

[3] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[4] Luca Bergamini, Yawei Ye, Oliver Scheel, Long Chen, Chih Hu, Luca Del Pero, Błażej Osiński, Hugo Grimmett, and Peter Ondruska. SimNet: Learning reactive self-driving simulations from real-world observations. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5119–5125, 2021.

[5] Marco Cusumano-Towner, David Hafner, Alexander Hertzberg, Brody Huval, Aleksei Petrenko, Eugene Vinitsky, Erik Wijmans, Taylor W. Killian, Stuart Bowers, Ozan Sener, Philipp Kraehenbuehl, and Vladlen Koltun. Robust autonomy emerges from self-play. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

[6] Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning. In *Conference on Robot Learning*, pages 1268–1281. PMLR, 2023.

[7] Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. NAVSIM: Data-driven non-reactive autonomous vehicle simulation and benchmarking. *arXiv preprint arXiv:2406.15349*, 2024.

[8] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[9] Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva: Exploring the limits of masked visual representation learning at scale. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19358–19369, 2023.

[10] Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Dingkang Liang, Chong Zhang, Dingyuan Zhang, Hongwei Xie, Bing Wang, and Xiang Bai. Orion: A holistic end-to-end autonomous driving framework by vision-language instructed action generation. *arXiv preprint arXiv:2503.19755*, 2025.

[11] Simon Gerstenecker, Andreas Geiger, and Katrin Renz. Fail2Drive: Benchmarking closed-loop driving generalization. *arXiv preprint arXiv:2604.08535*, 2026.

[12] Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinlei Pan, Yan Wang, Xiangyu Chen, John D. Co-Reyes, Rishabh Agarwal, Rebecca Roelofs, Yao Lu, Nico Montali, Paul Mougin, Zoey Yang, Brandyn White, Aleksandra Faust, Rowan McAllister, Dragomir Anguelov, and Benjamin Sapp. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

[13] K. Tan et al. H. Caesar, J. Kabzan. Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. In *CVPR ADP3 workshop*, 2021.

[14] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5): 4282–4286, 1995.

[15] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17853–17862, 2023.

[16] Zhiyu Huang, Xinshuo Weng, Maximilian Igl, Yuxiao Chen, Yulong Cao, Boris Ivanovic, Marco Pavone, and Chen Lv. Gen-drive: Enhancing diffusion generative driving policies with reward modeling and reinforcement learning fine-tuning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3445–3451. IEEE, 2025.

[17] Jyh-Jing Hwang, Runsheng Xu, Hubert Lin, Wei-Chih Hung, Jingwei Ji, Kristy Choi, Di Huang, Tong He, Paul Covington, Benjamin Sapp, Yin Zhou, James Guo, Dragomir Anguelov, and Mingxing Tan. Emma: End-to-end multimodal model for autonomous driving. *arXiv preprint arXiv:2410.23262*, 2024.

[18] Bernhard Jaeger, Daniel Dauner, Jens Beißwenger, Simon Gerstenecker, Kashyap Chitta, and Andreas Geiger. Carl: Learning scalable planning policies with simple rewards. *arXiv preprint arXiv:2504.17838*, 2025.

[19] Xiaosong Jia, Yulu Gao, Li Chen, Junchi Yan, Patrick Langechuan Liu, and Hongyang Li. Driveadapter: Breaking the coupling barrier of perception and planning in end-to-end autonomous driving. In *ICCV*, 2023.

[20] Xiaosong Jia, Penghao Wu, Li Chen, Jiangwei Xie, Conghui He, Junchi Yan, and Hongyang Li. Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In *CVPR*, 2023.

[21] Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. In *NeurIPS 2024 Datasets and Benchmarks Track*, 2024.

[22] Xiaosong Jia, Junqi You, Zhiyuan Zhang, and Junchi Yan. Drivetransformer: Unified transformer for scalable end-to-end autonomous driving. In *The Thirteenth International Conference on Learning Representations*, 2025.

[23] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8350, 2023.

[24] Bo Jiang, Shaoyu Chen, Bencheng Liao, Xingyu Zhang, Wei Yin, Qian Zhang, Chang Huang, Wenyu Liu, and Xinggang Wang. Senna: Bridging large vision-language models and end-to-end autonomous driving. *arXiv preprint arXiv:2410.22313*, 2024.

[25] Bo Jiang, Shaoyu Chen, Qian Zhang, Wenyu Liu, and Xinggang Wang. Alphadrive: Unleashing the power of vlms in autonomous driving via reinforcement learning and reasoning. *arXiv preprint arXiv:2503.07608*, 2025.

[26] Peter Karkus, Maximilian Igl, Yuxiao Chen, Kashyap Chitta, Boris Ivanovic, and Marco Pavone. Beyond behavior cloning in autonomous driving: a survey of closed-loop training techniques. Technical report, NVIDIA Research, 2025.

[27] Roberta Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*, 2023.

[28] Derun Li, Jianwei Ren, Yue Wang, Xin Wen, Pengxiang Li, Leimeng Xu, Kun Zhan, Zhongpu Xia, Peng Jia, Xianpeng Lang, et al. Finetuning generative trajectory model with reinforcement learning from human feedback. *arXiv preprint arXiv:2503.10434*, 2025.

[29] Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, et al. Recogdrive: A reinforced cognitive framework for end-to-end autonomous driving. In *International Conference on Learning Representations (ICLR)*, 2026.

[30] Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, et al. Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 12037–12047, 2025.

[31] Haochen Liu, Tianyu Li, Haohan Yang, Li Chen, Caojun Wang, Ke Guo, Haochen Tian, Hongchen Li, Hongyang Li, and Chen Lv. Reinforced refinement with self-aware expansion for end-to-end autonomous driving. *arXiv preprint arXiv:2506.09800*, 2025.

[32] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[33] Jiageng Mao, Yuxi Qian, Junjie Ye, Hang Zhao, and Yue Wang. Gpt-driver: Learning to drive with gpt. *arXiv preprint arXiv:2310.01415*, 2023.

[34] Nico Montali, John Lambert, Paul Mougin, Alex Kuefler, Nick Rhinehart, Michelle Li, Cole Gulino, Tristan Emrich, Zoey Yang, Shimon Whiteson, Brandyn White, and Dragomir Anguelov. The Waymo Open Sim Agents Challenge. *arXiv preprint arXiv:2305.12032*, 2023.

[35] Davis Rempe, Jonah Philion, Leonidas J Guibas, Sanja Fidler, and Or Litany. Generating useful accident-prone driving scenarios via a learned traffic prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[36] Katrin Renz, Long Chen, Elahe Arani, and Oleg Sinavski. Simlingo: Vision-only closed-loop autonomous driving with language-action alignment. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 11993–12003, 2025.

[37] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[38] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. *arXiv preprint arXiv:2209.13508*, 2022.

[39] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550:354–359, 2017.

[40] Ziying Song, Caiyan Jia, Lin Liu, Hongyu Pan, Yongchang Zhang, Junming Wang, Xingyu Zhang, Shaoqing Xu, Lei Yang, and Yadan Luo. Don't shake the wheel: Momentum-aware planning in end-to-end autonomous driving. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22432–22441, 2025.

[41] Guohao Sun, Hang Hua, Jian Wang, Jiebo Luo, Sohail Dianat, Majid Rabbani, Raghuveer Rao, and Zhiqiang Tao. Latent chain-of-thought for visual reasoning. *arXiv preprint arXiv:2510.23925*, 2025.

[42] Wenchao Sun, Xuewu Lin, Yining Shi, Chuang Zhang, Haoran Wu, and Sifa Zheng. Sparsedrive: End-to-end autonomous driving via sparse scene representation. *arXiv preprint arXiv:2405.19620*, 2024.

[43] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. TrafficSim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10400–10409, 2021.

[44] Yingqi Tang, Zhuoran Xu, Zhaotie Meng, and Erkang Cheng. Hip-ad: Hierarchical and multi-granularity planning with deformable attention for autonomous driving in a single decoder. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 25605–25615, 2025.

[45] Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Chenxu Hu, Yang Wang, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models. *arXiv preprint arXiv:2402.12289*, 2024.

[46] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575:350–354, 2019.

[47] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[48] Xinshuo Weng, Boris Ivanovic, Yan Wang, Yue Wang, and Marco Pavone. Para-drive: Parallelized architecture for real-time autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15449–15458, 2024.

[49] Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. *Advances in Neural Information Processing Systems*, 35:6119–6132, 2022.

[50] Zebin Xing, Xingyu Zhang, Yang Hu, Bo Jiang, Tong He, Qian Zhang, Xiaoxiao Long, and Wei Yin. Goalflow: Goal-driven flow matching for multimodal trajectories generation in end-to-end autonomous driving. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1602–1611, 2025.

[51] Rajeev Yasarla, Deepti Hegde, Shizhong Han, Hsin-Pai Cheng, Yunxiao Shi, Meysam Sadeghigooghari, Shweta Mahajan, Apratim Bhattacharyya, Litian Liu, Risheek Garrepalli, Thomas Svantesson, Fatih Porikli, and Hong Cai. Generative scenario rollouts for end-to-end autonomous driving. *arXiv preprint arXiv:2601.11475*, 2026.

[52] Liuhan Yin, Runkun Ju, Guodong Guo, and Erkang Cheng. Diffrefiner: Coarse to fine trajectory planning via diffusion refinement with semantic interaction for end to end autonomous driving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12009–12017, 2026.

[53] Linrui Zhang, Zhenghao Peng, Quanyi Li, and Bolei Zhou. CAT: Closed-loop adversarial training for safe end-to-end driving. In *Conference on Robot Learning*, 2023.

[54] Lingjun Zhang, Changjie Wu, Linzhe Shi, Jiangyang Li, Jiaxin Liu, Lei Yang, Hang Zhang, Mu Xu, and Hong Wang. DeepSight: Long-horizon world modeling via latent states prediction for end-to-end autonomous driving. *arXiv preprint arXiv:2605.10564*, 2026.

[55] Yinan Zheng, Ruiming Liang, Kexin Zheng, Jinliang Zheng, Liyuan Mao, Jianxiong Li, Weihao Gu, Rui Ai, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Diffusion-based planning for autonomous driving with flexible guidance. In *The Thirteenth International Conference on Learning Representations*, 2025.

[56] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17863–17873, 2023.

[57] Zewei Zhou, Tianhui Cai, Seth Z Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. Autovla: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. *arXiv preprint arXiv:2506.13757*, 2025.

A Ablation Study

A.1 Number of Reactive Agents

Agent Distribution in Bench2Drive. To contextualize our choice of reactive agent count, we analyze the distribution of traffic agents in the Bench2Drive validation set across 12,806 frames. As reported in Table 6, each frame contains on average 11.91 vehicles (std: 9.59, median: 8, max: 41), 0.06 pedestrians, and 0.03 cyclists. Vehicles account for over 99% of all annotated agents. The high variance and gap between mean and median indicate that scene density varies significantly across routes, with some frames containing up to 41 vehicles. These statistics motivate our investigation into the number of reactive agents and confirm that 8 reactive agents provides sufficient coverage for the large majority of evaluation scenarios.

Table 6: Per-frame agent statistics in the Bench2Drive validation set (12,806 frames).

Category	Mean	Std	Min	Max	Median
Vehicles	11.91	9.59	0	41	8.0
Pedestrians	0.06	0.33	0	3	0.0
Cyclists	0.03	0.18	0	1	0.0
Total agents	12.00	9.57	0	41	8.0

Ablation on the Number of Reactive Agents. Reactive agents are selected as those most relevant to the current driving scenario. We study the effect of the number of reactive agents by varying this number among 1, 2, 4, and 8, with agents selected using a KNN-based strategy. As shown in Table 7, the baseline uses no reactive agents with multi-planner and diversity regularization. Increasing the number of reactive agents consistently improves both Driving Score and Success Rate, yielding gains of up to +2.6 DS and +4.8 SR when using 8 reactive agents. Since 8 reactive agents already capture most influential interactions even in dense driving scenarios, further increasing this number is likely to introduce additional computational overhead with limited benefit; therefore, we select 8 reactive agents in all experiments.

Table 7: Ablation study on the number of reactive agents under closed-loop evaluation on Bench2Drive (B2D). All results are obtained using multi-planner with diversity regularization. All Δ values denote improvement *w.r.t. the setting with 0 reactive agents*.

# Reactive Agents	DS ↑	ΔDS	SR (%) ↑	ΔSR
0	82.6	–	62.7	–
1	82.8	+0.2	63.4	+0.7
2	83.9	+1.3	65.4	+2.7
4	84.5	+1.9	66.6	+3.9
8	**85.2**	**+2.6**	**67.5**	**+4.8**

A.2 Losses in SFT

Table 8 analyzes the effect of applying supervised losses to different entities during SFT. Starting from the setting without explicit SFT supervision (ID 1), adding the ego planning loss (ID 2) provides a clear gain in both Driving Score and Success Rate, indicating improved ego control under closed-loop evaluation. Supervising background-agent motion prediction in addition to ego (ID 3) further improves performance, suggesting that more accurate scene evolution during rollouts benefits ego planning. Finally, adding supervision for reactive-agent planning (ID 4) yields the best results, demonstrating that explicitly training interactive agents alongside the ego leads to stronger closed-loop robustness.

A.3 Rewards in RL

Table 9 presents the performance improvements of each reward term in our RL fine-tuning objective. Starting from the base setting SFT model (*i.e.*, without additional rewards), adding the global

Table 8: Ablation of supervised losses used during SFT. "Ego" denotes applying the planning loss to the ego trajectory, $\mathbf{Z}^{\text{react}}$ denotes supervising reactive-agent planning, and $\mathbf{Z}^{\text{bg}}$ denotes supervising background-agent motion prediction.

ID	Ego	$\mathbf{Z}^{\text{react}}$	$\mathbf{Z}^{\text{bg}}$	DS ↑	SR (%) ↑
(1)	✗	✗	✗	70.4	54.3
(2)	✓	✗	✗	75.2	56.1
(3)	✓	✗	✓	77.5	57.4
(4)	✓	✓	✓	79.6	59.1

reward G_t improves both Driving Score (DS) and Success Rate (SR), indicating that scene-level safety feedback provides useful training signal. Incorporating the agent-specific reward r_t^i yields a larger gain, improving DS to 83.4 and SR to 66.9, reflecting better progress and interaction-aware behavior at the agent level. Finally, enabling the diversity reward D_t further boosts performance to the best overall results (DS 85.2, SR 67.5), suggesting that diversity regularization complements safety/progress rewards by mitigating mode collapse and encouraging distinct yet effective behaviors during rollout-based training.

Table 9: Ablation on reward components. G_t denotes the global reward, r_t^i the agent-specific reward, and D_t the diversity reward.

ID	G_t	r_t^i	D_t	DS ↑	SR (%) ↑
(5)	✗	✗	✗	79.6	59.1
(6)	✓	✗	✗	81.1	62.2
(7)	✓	✓	✗	83.4	66.9
(8)	✓	✓	✓	**85.2**	**67.5**

B Diversity Reward

To encourage heterogeneous driving styles (e.g., conservative vs. assertive merges) and to mitigate mode collapse during RL fine-tuning, we introduce a diversity reward that promotes behavioral separation among a *group* of sampled rollouts. Concretely, we maintain a pool of reactive-agent policies/planners $\mathcal{P} = \{\pi_\theta^e\} \cup \{\pi_\theta^{a_j}\}_{j=1}^{N_R}$, where π^e is the ego policy/planner and $\{\pi_\theta^{a_j}\}$ are behavior-specialized policies/planners used to generate trajectories for N_R reactive agents. In all experiments, we set $N_R = 8$.

Behavior Descriptor. We summarize each rolled-out trajectory segment $\tilde{\tau}_{t:t+T}$ using a descriptor $\Gamma(\cdot)$ formed from the NAVSIM driving-quality metrics [7]: No at-fault Collisions (NC), Drivable Area Compliance (DAC), Driving Direction Compliance (DDC), Traffic Light Compliance (TLC), Ego Progress (EP), Time to Collision (TTC) within bound, Lane Keeping (LK), History Comfort (HC), and Extended Comfort (EC). These measures describe safety, rule compliance, progress, and comfort.

Planner Assignment and Rollout Sampling. For each reactive agent $a_j \in \mathbf{Z}_t^{\text{react}}$, we assign a behavior category (e.g., *aggressive*, *cautious*, *rule-compliant*, *high-comfort* from the NAVSIM driving-quality metrics) and compute the diversity reward D_t for agent a_j accordingly. For example, if agent a_7 is categorized as aggressive and agent a_2 as safe/comfortable, $\pi_\theta(a_7)$ and $\pi_\theta(a_2)$ generate their respective predicted trajectories $\tilde{\tau}^{a_2}$ and $\tilde{\tau}^{a_3}$. D_t for a_7 is computed using the $\Gamma(\cdot)$ aggressive descriptor and D_t for a_2 is computed using the $\Gamma(\cdot)$ safe/comfortable descriptor. This yields predicted trajectories for all reactive agents in the rollout.

Diversity Reward. We define the diversity reward as the average pairwise ℓ_1 distance between behavior descriptors across the different behavior metrics:

$$D_t = \frac{1}{N_R} \sum_{j=1}^{N_R} \sum_{\Delta=1}^{T} \left\| \Gamma^e\left(\tilde{\tau}^e_{t+\Delta}\right) - \Gamma^{a_j}\left(\tilde{\tau}^{a_j}_{t+\Delta}\right) \right\|_1, \tag{8}$$

This reward assigns higher values when different planner selections induce measurably distinct (yet safe and rule-compliant) behaviors, thereby encouraging the policy to maintain a diverse set of plausible interaction patterns during training.

C Inference Computation Analysis

Table 10 reports the per-frame inference latency of MAPLE measured on a single NVIDIA H100 (80GB) GPU with batch size 1, using 10 warmup batches followed by 50 timed batches with `torch.cuda.synchronize()` for accurate GPU timing. Each timed forward pass encompasses the full inference pipeline: (1) a multi-camera image backbone and neck that extract per-frame visual features from surrounding cameras; (2) a BEV encoder that lifts and fuses perspective features into a bird's-eye-view representation; (3) a Qwen2.5-1.5B language model head that encodes scene context and generates agent-conditioned queries; (4) a map head for local map prediction; and (5) per-agent decoders (comprising present/future distribution encoders, GRU-based trajectory predictors, and ego future decoders) instantiated independently for each of the 8 reactive planners. Rollout is disabled at inference, so no environment simulation or autoregressive steps are performed. With 8 reactive-agent planners, MAPLE requires 12,988 GFLOPs per inference and achieves a mean latency of 487.7 ms (2.05 FPS), with a P90 of 529.1 ms. The compute cost is dominated by the Qwen2.5-1.5B language model, which accounts for over 99% of total FLOPs, while the image backbone contributes only 15 GFLOPs.

Table 10: Computational cost and per-frame inference latency of MAPLE on a single NVIDIA H100 GPU (batch size 1). Rollout is disabled at inference.

Method	**#Planners**	**GFLOPs**	**Mean (ms)**	**P50 (ms)**	**P90 (ms)**	**Inferences Per Second**
MAPLE	8	12,988	487.7	491.6	529.1	2.05

D Open-Loop Evaluation on Bench2Drive

In Table 11, we present open-loop assessment on Bench2Drive, we additionally report the average L2 error between predicted and ground-truth trajectories over a 2-second horizon at 2Hz.

Table 11: Unified comparison of closed-loop, open-loop, and multi-ability performance on the Bench2Drive base set. Avg. L2 denotes the average trajectory error over 2 seconds at 2 Hz. NC = Navigation Command, TP = Target Point. † indicates models trained with geometric path waypoint supervision and evaluated using the PID controller from SimLingo.

Method	Condition	Closed-loop Metric				Open-loop	Multi-Ability Test(%) ↑					
		DS↑	SR(%)↑	Efficiency↑	Comfort↑	Avg. L2↓	Merging	Overtaking	E-Brake	Give Way	T.Sign	Mean
TCP-traj [49]	TP	59.9	30.0	76.5	18.1	1.70	28.8	24.3	51.7	40.0	46.3	34.2
TCP-traj w/o distillation [49]	TP	49.3	20.5	78.8	22.9	1.96	28.7	28.7	48.3	40.0	28.7	34.1
ThinkTwice [20]	TP	62.4	31.2	69.3	16.2	0.95	27.4	18.4	35.8	50.0	54.2	37.2
DriveAdapter [19]	TP	64.2	33.1	70.2	16.0	1.01	28.4	28.4	47.5	50.0	56.4	42.1
SimLingo [36]†	TP	85.1	67.3	259.2	33.7	-	54.0	57.0	88.3	53.3	82.5	67.0
UniAD-Base [15]	NC	45.8	16.4	129.2	43.6	0.73	8.9	9.3	20.0	20.0	14.2	14.5
VAD [23]	NC	42.4	15.0	157.9	46.0	0.91	11.4	11.4	18.6	20.0	19.2	18.1
MomAD [40]	NC	44.5	16.7	170.2	48.6	0.87	-	-	-	-	-	-
DriveTransformer-Large [22]	NC	63.5	35.0	100.6	20.8	0.62	17.6	35.0	48.4	40.0	52.1	38.6
HiP-AD [44]†	NC	86.8	69.1	203.1	19.4	0.69	50.0	84.4	83.3	40.0	50.5	65.9
Qwen2.5 [3]	NC	63.9	31.6	119.3	10.1	0.87	14.3	28.9	30.1	30.0	24.7	25.6
ORION [10]	NC	77.7	54.6	151.5	17.4	0.68	25.0	71.1	78.3	30.0	69.2	54.7
ReCogDrive [29]	NC	71.4	45.5	138.2	17.5	-	29.7	20.0	69.1	20.0	71.3	42.0
GeRo [51]	NC	81.9	60.1	176.5	40.2	0.57	40.1	78.2	87.3	50.0	76.8	66.5
MAPLE (ours)	NC	85.2	67.5	184.3	37.9	0.54	46.3	80.7	88.1	60.0	78.1	70.6
MAPLE (ours)†	NC	88.3	70.5	210.3	39.0	0.53	49.4	86.2	90.0	60.0	80.4	73.2

E Rollout Realism and Broader Discussion

E.1 Rollout Realism

We evaluate predicted trajectories using metrics adapted from the Waymo Open Sim Agents Challenge [34], with ground-truth Bench2Drive trajectories as the reference distribution. The kinematic metric averages likelihoods of linear speed, linear acceleration, angular speed, and angular acceleration. The interactive metric averages likelihoods of distance to the nearest object, collision indication, and time-to-collision. The realism meta metric is a linear aggregation of these metrics. We omit the map metric because distance-to-center values are unavailable for some methods. All compared checkpoints are evaluated under the same protocol using the best checkpoint from each method.

Table 3 in the main paper reports the resulting scores. MAPLE improves all three metrics, including the realism meta score from 0.391 for ORION to 0.572. DeepSight provides a relevant latent world-modeling comparison. These metrics directly assess trajectory-level agreement with reference motion and interaction statistics; they do not guarantee exact physical dynamics for every generated future.

E.2 Latent Exploration, Model Scale, and Pretrained Knowledge

Imperfect rewards can favor unrealistic latent futures. MAPLE mitigates this through auxiliary state-estimation pretraining and rollout SFT before RL, together with collision, TTC, progress, and behavior-based rewards. Diversity is measured through driving-quality descriptors rather than arbitrary directions in latent space. The realism evaluation above provides complementary evidence about the resulting trajectories.

Larger VLM backbones may offer stronger semantic priors for rare scenes, but would also increase inference cost. Their effect on rollout realism and long-tail performance has not been evaluated here. The reported inference cost in Appendix C characterizes the current backbone.

The curriculum improves the controlled ego-only ablation from 68.1/51.9 DS/SR to 70.4/54.3 after state-estimation pretraining, 77.5/57.4 after rollout SFT, and 80.4/59.8 after rollout RL (see Table 4). These task-specific improvements and the Fail2Drive results indicate useful driving adaptation, but do not directly establish that general pretrained Qwen knowledge is preserved. A dedicated evaluation of forgetting remains future work.

F Qualitative Results

F.1 Qualitative Comparison with ReCogDrive

Figures 5 and 6 qualitatively compare MAPLE with ReCogDrive [29], which is a very recent work that leverages ego-only, single-step RL to train E2E VLA planners, under closed-loop evaluation on Bench2Drive for the same route. Both methods make progress through an interactive scene containing dynamic agents (e.g., two cyclists traveling along the road) and nearby traffic; however, their interaction outcomes differ. ReCogDrive exhibits a failure case around the 5th column of Figure 6: the planned trajectory deviates abruptly and results in a collision/infraction event, which is also reflected in the BEV visualization in Figure 5. This substantially reduces the composed score (score_penalty $= 0.4225$, score_composed $= 42.25$), despite the route being marked as completed. In contrast, MAPLE completes the same route without collision infractions (Figures 5 and 6) and achieves a perfect composed score (score_penalty $= 1.0$, score_composed $= 100$). Visually, MAPLE produces smoother and more consistent planned ego trajectories (blue/cyan) with gradual curvature changes while maintaining safe separation from the cyclists and surrounding vehicles, whereas ReCogDrive shows sharper trajectory changes preceding the infraction. Overall, this example highlights that MAPLE yields more robust closed-loop behavior in interactive scenarios, avoiding safety-critical failures that can occur even when route completion is achieved.

F.2 Additional Qualitative Results

Figures 7 and 8 provide additional qualitative closed-loop driving results on the Bench2Drive benchmark using MAPLE, covering a broader range of weather, lighting, and interaction patterns. For each example, we show multiple time steps from the ego view with the planned ego trajectory

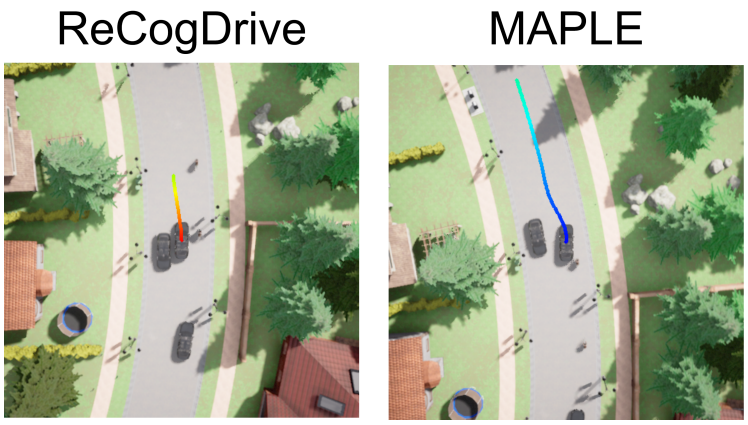

Figure 5: **BEV qualitative comparison on Bench2Drive (closed-loop).** Bird's-eye-view visualization for the same route/scenario (`RouteScenario_25951_rep0`, `HazardAtSideLaneTwoWays_1`, `weather_id=7`). Left: ReCogDrive [29]. Right: MAPLE (ours). The planned ego trajectory is overlaid, illustrating different interaction outcomes in the same context.

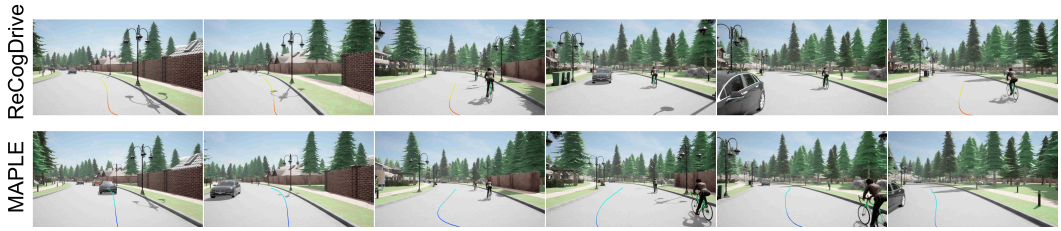

Figure 6: **Closed-loop rollout comparison on Bench2Drive.** Multi-frame qualitative rollouts for the same route/scenario (`RouteScenario_25951_rep0`, `HazardAtSideLaneTwoWays_1`, `weather_id=7`). Top row: ReCogDrive. Bottom row: MAPLE. Colored curves denote the planned ego trajectory across time, highlighting differences in closed-loop interaction behavior.

overlaid (blue/cyan). Figure 7 highlights particularly challenging perception and interaction regimes: (i) low-light/night driving on wet roads, where the ego plan remains centered and stable even when a pedestrian suddenly enters near the roadway and nearby vehicles occupy adjacent lanes; (ii) dense fog on a multi-lane road, which substantially reduces visibility and shortens the effective planning horizon, where MAPLE maintains a smooth, lane-consistent trajectory; and (iii) adverse-weather urban driving at intersections with complex lane markings and cross traffic, where MAPLE executes a left turn with gradual curvature and without abrupt corrections. Figure 8 further includes suburban/rural scenes with oncoming traffic, vehicles moving at varying speeds, and gentle-to-moderate road curvature, as well as nighttime intersection scenarios under wet-road illumination. Across these diverse settings, MAPLE consistently produces temporally coherent plans with smooth curvature changes and stable lane positioning, while adapting the trajectory to dynamic agents and roadway structure.

F.3 Failure Case

Figure 9 shows a failure case where MAPLE becomes overly cautious during a unprotected left turn and steers toward the road boundary to avoid a potential collision. Resulting in a brief deviation outside the route lanes forabout 1.0 meters (1.29% of the completed route).

G Losses

This section details the loss functions used in the pretraining and supervised fine-tuning (SFT) stages of MAPLE. We first pretrain the VLA backbone with auxiliary perception tasks to obtain stable scene representations, and then optimize planning and prediction objectives with rollout-based SFT.

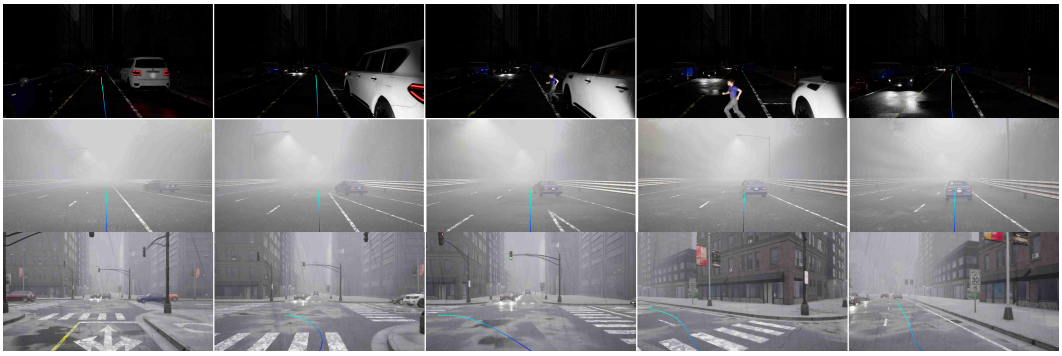

Figure 7: **Additional qualitative closed-loop driving examples on Bench2Drive using MAPLE.** These examples includes challenging conditions, like low-light/night driving with sudden pedestrian appearances and wet-road reflections, dense fog/highway driving with reduced visibility, and urban scenes with adverse weather. Blue/cyan curves denote the planned ego trajectory.



Figure 8: **Additional qualitative closed-loop driving examples on Bench2Drive using MAPLE.** More examples covering suburban/rural traffic with oncoming vehicles and lane curvature, as well as nighttime intersection scenarios with wet-road conditions and surrounding traffic. Blue/cyan curves denote the planned ego trajectory.

G.1 Pretraining

During pretraining, the VLA model is trained with auxiliary tasks including 3D object detection, map segmentation, agent motion prediction, traffic-light/traffic-state prediction, and future state prediction. These losses encourage the backbone to encode geometry, semantics, and traffic context into latent tokens, stabilizing subsequent rollout training.

3D Object Detection. For detection (via the proposed query-based detector), we use a standard classification + regression objective:

$$\mathcal{L}_{\text{det}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{reg}}, \tag{9}$$

where $\mathcal{L}_{\text{cls}}$ is a focal loss and $\mathcal{L}_{\text{reg}}$ is an ℓ_1 regression loss.

Traffic-State Prediction. For traffic-state (e.g., traffic light / traffic status) prediction, we apply a focal loss:

$$\mathcal{L}_{\text{tra}} = \mathcal{L}_{\text{focal}}. \tag{10}$$

Agent Motion Prediction. For multi-agent motion prediction, we use a classification + regression loss:

$$\mathcal{L}_{\text{mot}} = \mathcal{L}_{\text{mcls}} + \mathcal{L}_{\text{mreg}}, \tag{11}$$

where $\mathcal{L}_{\text{mcls}}$ is a focal loss over motion modes and $\mathcal{L}_{\text{mreg}}$ is an ℓ_1 regression loss on the predicted trajectories.

Map Segmentation / Vectorized Map Learning. For map learning, we use a classification loss (focal loss) and a regression loss based on Manhattan (ℓ_1) distance between predicted map points and ground-truth map points:

$$\mathcal{L}_{\text{map}} = \mathcal{L}_{\text{map-cls}} + \lambda_{\text{map}}\mathcal{L}_{\text{map-reg}}, \qquad \mathcal{L}_{\text{map-reg}} = \|\hat{\mathbf{m}} - \mathbf{m}\|_1. \tag{12}$$

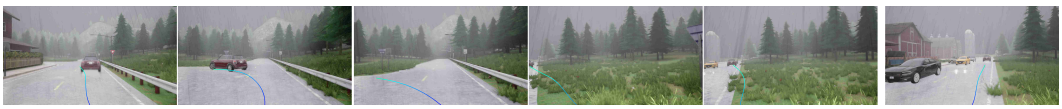

Figure 9: **Failure case: over-cautious avoidance leading to lane departure (Bench2Drive, closed-loop).** In this scenario, MAPLE performs an overly conservative unprotected left turn to avoid a potential collision, resulting in a brief deviation outside the route lanes for *about 1.0 meters* (*1.29%* of the full route). The car quickly moves back to the lane after this brief deviation. Blue/cyan curves denote the planned ego trajectory over time.

This follows the practice for vectorized map prediction used in VAD [23].

Future State Prediction. We additionally supervise future state prediction using the loss in Eq. 1, denoted as $\mathcal{L}_{\text{state}}$.

Overall Pretraining Objective. The full pretraining loss is the weighted sum of all components:

$$\mathcal{L}_{\text{pre}} = \lambda_{\text{det}}\mathcal{L}_{\text{det}} + \lambda_{\text{tra}}\mathcal{L}_{\text{tra}} + \lambda_{\text{mot}}\mathcal{L}_{\text{mot}} + \lambda_{\text{state}}\mathcal{L}_{\text{state}} + \lambda_{\text{map}}\mathcal{L}_{\text{map}}. \tag{13}$$

We use the same loss-weight coefficients as VAD [23] for the auxiliary-task objectives, i.e., λ_{det}, λ_{mot}, and λ_{map} follow the weighting scheme in [23]. We set λ_{tra}, and λ_{state} as 1.0.

G.2 Planning Loss (SFT / Action Planner)

The action planner in MAPLE is trained with a generative objective and trajectory-level supervision. In particular, the planner uses a VAE-style latent alignment loss and trajectory regression, together with safety/feasibility regularizers.

VAE Latent Regularization. We use a Kullback–Leibler divergence term to align the latent "reasoning" space with the action/trajectory distribution:

$$\mathcal{L}_{\text{vae}} = D_{\text{KL}}(q(\mathbf{z} \mid \cdot) \,\|\, p(\mathbf{z}))\,. \tag{14}$$

Trajectory Regression. We supervise the predicted waypoints with an MSE (or ℓ_1) regression loss:

$$\mathcal{L}_{\text{mse}} = \frac{1}{T}\sum_{\Delta=1}^{T} \|\hat{\tau}_{t+\Delta} - \tau_{t+\Delta}\|_2^2\,. \tag{15}$$

Safety/Feasibility Regularizers. Following VAD [23], vectorized constraint-based training commonly used in end-to-end driving, we include collision and boundary regularizers to discourage unsafe trajectories:

$$\mathcal{L}_{\text{col}} \;\text{(collision avoidance)}, \qquad \mathcal{L}_{\text{bd}} \;\text{(boundary / drivable-area constraint)}. \tag{16}$$

These terms penalize trajectories that come too close to other agents or violate road boundaries, consistent with constraint-based planning losses in prior work.

Overall Planning Loss. The final planning objective is:

$$\mathcal{L}_{\text{plan}} = \lambda_{\text{vae}}\mathcal{L}_{\text{vae}} + \lambda_{\text{mse}}\mathcal{L}_{\text{mse}} + \lambda_{\text{col}}\mathcal{L}_{\text{col}} + \lambda_{\text{bd}}\mathcal{L}_{\text{bd}}. \tag{17}$$

We use the same loss-weight coefficients as VAD [23] and ORION [10] for the weights λ_{vae}, λ_{mse}, and λ_{col}, and λ_{bd}.

H Training Details

H.1 Bench2Drive Training

The standard Bench2Drive curriculum follows Section 4: 18 epochs of auxiliary pretraining, 24 epochs of rollout SFT, and 6 epochs of RL on 8 NVIDIA H100 GPUs. AdamW uses a learning rate of 2×10^{-4} and weight decay 0.01. We use four input frames, eight reactive-agent planners, and $T = 8$ rollout steps with the temporal rates listed in the main paper.

H.2 Additional nuScenes Fine-Tuning

The optimizer configuration below applies to an additional nuScenes fine-tuning stage after the standard Bench2Drive curriculum; it is not the Bench2Drive training recipe. We use AdamW [32] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay 10^{-5}. The base learning rate is 10^{-5}, with layer-wise decay 0.9, 500 linear warm-up iterations, and a cosine schedule with minimum learning-rate ratio 10^{-3}. Fine-tuning runs for three epochs on 8 NVIDIA H100 GPUs, with one sample per GPU (effective batch size 8). The reported rollout configuration is eight steps at 2 Hz.

H.3 Reactive Agent Selection.

Reactive agents are the traffic participants whose future trajectories are jointly predicted with the ego during self-play rollout. At each rollout step, agents are ranked by their distance to the ego vehicle and the N_R nearest neighbors are designated as reactive; the remaining agents are treated as background agents motion trajectories predicted using motion prediction head.

H.4 Rollout Configuration.

Rollout is performed over N_{step}=8 steps at 2 Hz (a 4-second horizon). At each step, the ego trajectory and reactive-agent trajectories are predicted by the model and used to advance the scene state; the resulting state is fed back as input for the next step. During inference, rollout is disabled and a single forward pass is executed per frame.

H.5 Multi-Planner Setup.

We instantiate N_R behavior-specialized planners drawn from a shared pool $\mathcal{P} = \{P^e\} \cup \{P^{a_j}\}_{j=1}^{N_R}$. In all reported experiments we use N_R=8 reactive-agent planners (see Appendix A for ablations). During RL fine-tuning, the best-performing planner for the ego is selected based on the reward likelihood learned in the GRPO stage.

I Open-loop evaluation on nuScenes

Table 12 reports open-loop planning under the PARA-Drive evaluation protocol [48]. Zero-shot MAPLE uses the Bench2Drive-trained checkpoint; the fine-tuned model includes the additional nuScenes stage. Fine-tuning reduces L2 trajectory error from 0.30 to 0.22 and collision rate from 0.14 to 0.12.

Table 12: nuScenes open-loop planning under the PARA-Drive[48] evaluation protocol. Lower is better. Zero-shot MAPLE is trained on Bench2Drive; MAPLE additionally fine-tunes on nuScenes.

Method	L2 trajectory error↓	Collision rate↓
UniAD [15]	0.95	0.40
VAD [23]	0.91	0.30
PARA-Drive [48]	0.66	0.17
SparseDrive [42]	0.65	0.14
ORION [10]	0.69	0.29
MAPLE (zero-shot)	0.30	0.14
MAPLE	0.22	0.12

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's scope and contributions, clearly describing the proposed MAPLE framework, its multi-agent latent rollout mechanism, diversity-aware training strategy, and the resulting closed-loop performance gains demonstrated in the experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present formal theoretical results (e.g., theorems or propositions with proofs); it focuses on algorithmic design, training methodology, and empirical evaluation of the MAPLE framework.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides sufficient details on datasets, evaluation protocols, model architecture, multi-stage training procedure, and hyperparameters (Sections 3 & 4 and appendix) to enable reproducing the main results of the paper.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: While the paper uses publicly available datasets, the code and trained models are not currently released, which is subject to company legal and compliance review; therefore, full open access is not provided at submission time. We will aim to release the code.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies training and evaluation details, including datasets and splits, multi-stage training procedures, hyperparameters, and optimization choices in Sections 3 and 4, which are sufficient to understand and interpret the reported results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The experimental results are reported as single-run benchmark metrics without error bars, confidence intervals, or statistical significance analysis, which is common practice for closed-loop autonomous driving benchmarks but does not quantify variability across runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper specifies the compute setup (GPU type), training configuration, and execution details in the training section, and additionally includes inference-time analysis, providing sufficient information to understand and reproduce the computational requirements of the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics by using publicly available datasets, avoiding human-subject experimentation, preserving anonymity, and presenting results transparently, with no identified ethical, legal, or compliance concerns.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses broader impacts in Section 6; there are no negative societal impacts associated with this work.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [Yes]

Justification: The paper does not release trained models or new datasets and relies solely on publicly available data; any future release of trained models is subject to institution legal and compliance review, which serves as a safeguard to ensure responsible and controlled dissemination and to mitigate potential misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All external assets used in the paper—such as datasets (e.g., Bench2Drive) and pretrained models (e.g., Qwen2.5)—are properly cited and credited, and their use complies with the licenses and terms of use made publicly available by the original authors; no assets are redistributed in violation of these terms.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not release new datasets, models, or code as standalone assets; therefore, there are no new assets provided alongside the paper that would require accompanying documentation.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects; all experiments are conducted using simulated environments and existing driving datasets without direct human participation.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects; all experiments are conducted in simulated environments and using existing datasets, so there are no study participants or IRB requirements.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Large language models are a core component of MAPLE 's methodology, serving as the backbone of the vision-language-action architecture for latent-space multi-agent rollouts, planning, and behavior modeling; their role and usage are explained in detail in Section 3 of the paper.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.