

Unleashing the Potential of Diffusion Models for End-to-End Autonomous Driving

Yinan Zheng^{1,*}, Tianyi Tan^{1,*}, Bin Huang^{2,*}, Enguang Liu², Ruiming Liang¹, Jianlin Zhang², Jianwei Cui², Guang Chen², Kun Ma², Hangjun Ye², Long Chen², Ya-Qin Zhang¹, Xianyuan Zhan^{1,†}, Jingjing Liu^{1,†}

¹Institute for AI Industry Research (AIR), Tsinghua University, ²Xiaomi EV

*Equal contribution, †Corresponding authors

Diffusion models have become a popular choice for decision-making tasks in robotics, and more recently, are also being considered for solving autonomous driving tasks. However, their applications and evaluations in autonomous driving remain limited to simulation-based or laboratory settings. The full strength of diffusion models for large-scale, complex real-world settings, such as End-to-End Autonomous Driving (E2E AD), remains underexplored. In this study, we conducted a systematic and large-scale investigation to unleash the potential of the diffusion models as planners for E2E AD, based on a tremendous amount of real-vehicle data and road testing. Through comprehensive and carefully controlled studies, we identify key insights into the diffusion loss space, trajectory representation, and data scaling that significantly impact E2E planning performance. Moreover, we also provide an effective reinforcement learning post-training strategy to further enhance the safety and robustness of the learned planner. The resulting diffusion-based learning framework, *Hyper Diffusion Planner* (HDP), is deployed on a real-vehicle platform and evaluated across 6 urban driving scenarios and 200 km of real-world testing, achieving a notable 10x performance improvement over the base model. Our work demonstrates that diffusion models, when properly designed and trained, can serve as effective and scalable E2E AD planners for complex, real-world autonomous driving tasks. Project website: <https://zhengyinan-air.github.io/Hyper-Diffusion-Planner/>.

Emails: zhengyn23@mails.tsinghua.edu.cn, zhanxianyuan@air.tsinghua.edu.cn

1 Introduction

Diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015) have demonstrated remarkable capabilities in image and video generation (Betker et al., 2023; Croitoru et al., 2023; Liu et al., 2024; Rombach et al., 2022) and are becoming increasingly popular in robotics (Black et al., 2024a; Chi et al., 2023; Intelligence et al., 2025; Liu et al., 2025a). Their generative, multimodal nature also makes them a natural fit for autonomous driving (AD), where human driving behavior is inherently diverse and stochastic (Zheng et al., 2025; Tan et al., 2025). Among AD paradigms, End-to-End Autonomous Driving (E2E AD) (Bojarski et al., 2016; Chen et al., 2023; Hu et al., 2023b) has emerged as one of the most practically viable directions toward production-grade autonomy (Tesla, 2022), and a growing body of work has accordingly explored diffusion-based planners in this setting (Liao et al., 2025; Li et al., 2025a; Wang et al., 2026), with encouraging benchmark results.

However, what these benchmarks actually reveal about diffusion-based planning is less clear than it appears, for two reasons. First, they offer only a narrow view of real-world driving. As illustrated in Figure 1, mainstream open-source datasets (Caesar et al., 2021) and simulation environments (Dauner et al., 2024) differ substantially from real-world deployment in data scale, trajectory diversity, and evaluation protocols. In particular, their open-loop and pseudo-closed-loop metrics are widely acknowledged to be poor proxies for true closed-loop performance on the road (Dauner et al., 2023a; Li et al., 2024c; Zheng et al., 2025), so planners that excel in simulation may degrade substantially in real-world deployment, as our experiments confirm. Second, even on these benchmarks, top-performing planners often rely on rule-based post-processing (Fan et al., 2018) or incorporate strong assistive designs—such as pre-defined anchor trajectories (Li et al., 2024a)

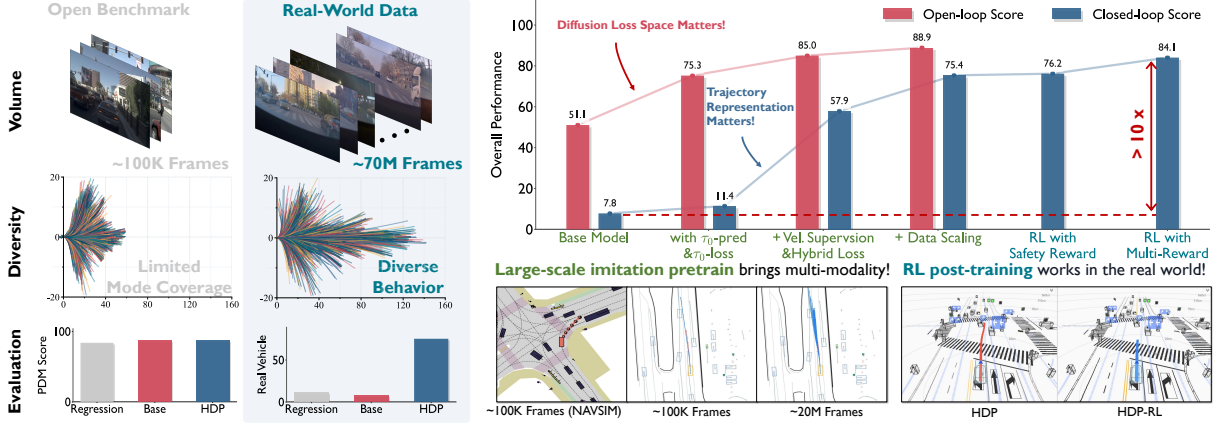


Figure 1 Simulation-based benchmarks differ substantially from real-world deployment, leaving the true capability of diffusion-based planners largely unverified. Through a detailed investigation of the IL/RL training recipe, we unleash the potential of diffusion-based planning, achieving a 10x improvement in real world.

or explicit goal conditions (Albrecht et al., 2021; Gu et al., 2021)— to reduce the learning burden, leaving it unclear how much of the gain comes from the model itself rather than from these hand-crafted priors. Taken together, the true ceiling of diffusion models as AD planners in real world remains unknown, raising a critical question: *Are we fully exploiting the potential of diffusion models as AD planners?*

To answer this question, we conduct a systematic investigation of diffusion-based E2E AD, leveraging massive real-vehicle data and rigorous on-road testing. We adopt a vanilla diffusion-based planning head (Zheng et al., 2025) for trajectory generation. We deliberately keep the diffusion head free of hand-crafted priors, so that any performance gain we observe can be attributed to diffusion modeling itself. Building on this base model, we identify a set of key design principles, each supported by real-vehicle evidence:

- **Diffusion loss space matters.** Our key insight stems from the observation that planning trajectory lives in a low-dimensional manifold, distinct from image generation. Consequently, we re-examine the diffusion loss space design (Li and He, 2025) and find that data (τ_0)-prediction combined with diffusion loss directly supervised on data (τ_0 -loss) best captures the trajectory manifold, enabling better learning performance and high-quality trajectory generation.
- **Trajectory representation matters.** In our τ_0 -prediction setup, predicting waypoints yields better spatial awareness, while predicting velocity gives smoother trajectories. Our model therefore outputs velocity but is supervised on both, achieving substantial closed-loop gains. Although this hybrid loss resembles auxiliary losses (Bansal et al., 2018; Hu et al., 2023b; Jiang et al., 2023) routinely used in AD, we *theoretically prove* that it leaves the diffusion training optimum unchanged, thereby exposing an often-overlooked pitfall: widely used auxiliary losses such as collision penalties or L_1 -norm loss would distort it.
- **Emergence of data scaling.** We further show that, without extra auxiliary losses or hand-crafted priors, our diffusion framework effectively benefits from data scaling in real-world testing. Our model captures richer multimodal driving behaviors, an effect not observed when training on commonly used benchmarks (Caesar et al., 2021; Dauner et al., 2024), whose data are too small to expose it. The same scaling also yields stronger open-loop and closed-loop performance, making our framework a viable planner.

While the imitation pretraining establishes a strong diffusion planner prior, it lacks explicit optimization for safety-critical scenarios. To close this gap, we further introduce Reinforcement Learning (RL) post-training that re-weights the diffusion objective with safety-aware advantage estimates (Peng et al., 2019; Zheng et al., 2024), aligning generated trajectories with safety constraints while preserving training stability. We *theoretically prove* that this reweighting is naturally compatible with our hybrid loss, enabling a clean implementation. The same formulation extends to multi-reward settings and further advances the Pareto frontier, showing that **RL post-training is effective in real-world closed-loop deployment.**

Finally, we integrate the above innovations into a complete framework, *Hyper Diffusion Planner* (HDP), and its RL post-trained variant *HDP-RL*. With only minimal smoothness post-processing, both are deployed

on a real vehicle. We evaluate them across 6 urban driving scenarios over 200 km of road testing under comprehensive metrics: *HDP* / *HDP-RL* achieve a 10x improvement over the base model. We further analyze the characteristics of the framework and the impact of key components, showing that diffusion models, when properly designed and trained, can serve as effective and scalable planners for real-world autonomous driving.

2 Preliminaries

Our work focuses on the planning module of E2E AD systems, where the planner receives the latent representation C from the scene encoder and generates a trajectory τ_0 for downstream control systems. Diffusion models (Sohl-Dickstein et al., 2015) define a forward process that transforms the conditional trajectory distribution $q_0(\tau_0 | C)$ into a noised distribution $q_{t0}(\tau_t | \tau_0) = \mathcal{N}(\tau_t | \alpha_t \tau_0, \sigma_t^2 \mathbf{I})$ for $t \in [0, 1]$, where α_t, σ_t define a pre-defined noise schedule. As $t \rightarrow 1$, this schedule ensures that the marginal distribution $q(\tau_1)$ approaches $\mathcal{N}(\tau_1 | 0, \mathbf{I})$. The reverse denoising process can be expressed as a diffusion ODE (Song et al., 2021):

$$d\tau_t = [f(t)\tau_t - \frac{1}{2}g^2(t)\nabla_{\tau_t} \log q_t(\tau_t)] dt, \quad (1)$$

where $f(t) = \frac{d \log \alpha_t}{dt}$ and $g^2(t) = \frac{d\sigma_t^2}{dt} - 2\frac{d \log \alpha_t}{dt} \sigma_t^2$. A commonly used approach for learning diffusion models (Ho et al., 2020; Rombach et al., 2022) is to train a neural network $\epsilon_\theta(\tau_t, t, C)$ to fit the Gaussian noise ϵ :

$$\mathcal{L} = \mathbb{E}_{t, \tau_0, \tau_t, \epsilon} \|\epsilon_\theta(\tau_t, t, C) - \epsilon\|_2^2, \quad (2)$$

where $t \sim \mathbb{U}(0, 1)$, $\tau_0 \sim q_0(\tau_0 | C)$, $\tau_t \sim q_{t0}(\tau_t | \tau_0)$ and $\epsilon \sim \mathcal{N}(\tau_1 | 0, \mathbf{I})$. Then, we can estimate the score function $\nabla_{\tau_t} \log q_t(\tau_t)$ in Eq. (1) using $s_\theta(\tau_t, t, C) = -\epsilon_\theta(\tau_t, t, C)/\sigma_t$, and use an ODE solver to generate the data.

3 Investigation Roadmap

In this section, we first introduce the base model and evaluation metrics for assessing model performance. Subsequently, we will briefly outline our investigation roadmap aimed at fully unleashing the potential of diffusion models for E2E AD.

3.1 Base Model

Scene Encoder. Following standard E2E AD practice (Chen et al., 2023; Hu et al., 2023b), we adopt a perception backbone that ingests multi-modal inputs (camera and LiDAR) and produces two sets of perception tokens—Object Detection (OD) and Lane Detection (LD). Together with Navi tokens encoding high-level navigation commands, they are concatenated and fused via self-attention into the latent condition $C := \text{Enc}(s)$ consumed by the planner. More architecture and training details are deferred to Appendix D.2.

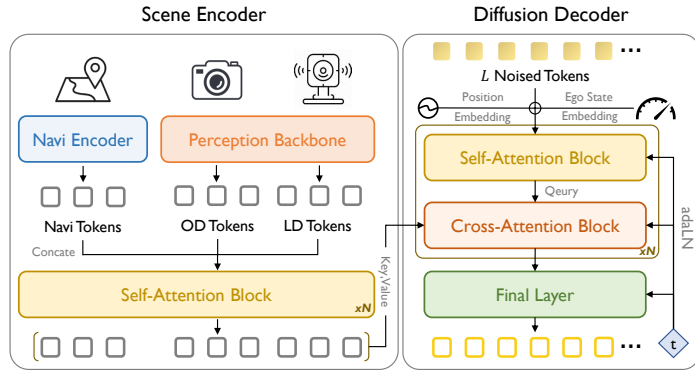


Figure 2 Model architecture of the base model.

Diffusion Decoder. We use a vanilla transformer-based diffusion model (Peebles and Xie, 2023) as the decoder. Conditioned on C and the current ego state, it generates the trajectory $\tau_0 \in \mathbb{R}^{L \times 4}$, where each of the L timesteps contains the ego-centric waypoint coordinates and the cosine/sine of the heading. As illustrated in Figure 2, the noised trajectory τ_t is first split and projected into L tokens, with position and velocity embeddings added, fused via self-attention, and then attended to C through cross-attention. The diffusion timestep t is injected via adaptive layer normalization (Peebles and Xie, 2023). After several such blocks, an MLP head (Liu et al., 2025a; Zheng et al., 2025) produces the prediction, and the model is trained using Eq. (2).

3.2 Evaluation Metrics

We adopt two complementary evaluation protocols. *Open-Loop* metrics evaluate trajectory quality via data replay, while *Closed-Loop* metrics measure performance in real-vehicle testing on a fixed route; full definitions and hyperparameters are in Appendix D.1.

Open-Loop. Following nuPlan (Caesar et al., 2021), we report *Average Displacement Error* (ADE), *Final Displacement Error* (FDE), *Comfort*, and *Collision Rate* (CR), and aggregate them into the *Open-Loop Score* $(1 - CR) \sum_{m \in \{ADE, FDE, Comfort\}} \omega_m S_m$. To probe multimodality, we additionally report *Trajectory Divergence*, the average pairwise Euclidean distance among samples drawn for the same scene.

Closed-Loop. For real vehicle testing, we use a fixed route to conduct controlled experiments. We log the *Success Rate* on six common driving scenarios, with any takeover counted as a failure: starting maneuvers, car-following with stopping, navigational lane changes, yielding to VRUs, yielding to cross traffic at intersections, and left and right turns. Per-scenario rates are aggregated by frequency-weighted averaging for a more representative evaluation. We further compute a *Stability Score* as the mean of *Centering Performance* and *Speed Compliance*, capturing off-center driving or improper speed even without takeover. The overall *Closed-Loop Score* is the mean of the two.

3.3 Roadmap Overview

With the base model and evaluation metrics ready, we start our journey to unleash the potential of diffusion models for E2E AD. As the AD planning task is obviously different from image generation tasks, with its output trajectories residing on a relatively low-dimensional manifold, need to satisfy hard constraints like collision avoidance, and is evaluated in a closed-loop setting that easily suffers from error accumulation. To address these challenges, we structure our exploration into two phases: 1) **Imitation Learning Pre-Training**, examining how diffusion loss and trajectory representation shape planning quality and validating closed-loop data scaling; and 2) **Reinforcement Learning Post-Training**, where a compatible RL algorithm enables stable, efficient enhancement of the pretrained model.

4 Imitation Learning Pre-training

4.1 Diffusion Loss Space

As the score function in the denoising process (Eq. (1)) is generally intractable, in practice, the diffusion model is typically trained to predict one of the three conditioned quantities: the noise ϵ (Ho et al., 2020), the flow velocity v_t (Ho et al., 2022), or the clean data τ_0 (Ramesh et al., 2022). These quantities are mutually convertible, allowing for various loss space designs (see Appendix C for more details). For instance, the diffusion model can be parameterized to output τ_0 , while being supervised with ϵ -loss. However, models trained in different loss spaces can exhibit distinct learning dynamics (Li and He, 2025) and planning behaviors. To investigate the impact of loss space design on the planning task, we trained our model with all 9 prediction-loss combinations and conducted open-loop evaluations. The results are shown in Table 1 and Figure 3, 4. Most models achieved competitive performance (except ϵ -pred with τ_0 - and v -loss), successfully capturing the expert policy in the training data while demonstrating multimodal generation capability. In addition, among these models, the τ_0 -prediction model trained with τ_0 -loss stands out prominently. To study the reasons for this advantage, we provide further investigation from the following perspectives.

Fast Convergence. Figure 3 displays the aggregated scores of models at various training stages. While the model utilizing τ_0 -prediction converged rapidly with increased training steps, the other two approaches experienced notable instability. This disparity stems from differences in the inherent dimensionality of the target manifold (Li and He, 2025). Because the trajectory τ_0 resides in a low-dimensional manifold, the neural network can capture it more easily. Conversely, the ϵ and v targets are supported on much higher-dimensional spaces and therefore require greater model capacity. Furthermore, τ_0 -loss works best for the τ_0 -prediction model compared with the other two choices.

High Generation Quality. Furthermore, we visualized the generated trajectories of different models, as shown in Figure 4. Although most models generate trajectories that resemble the ground truth, the quality

Table 1 : Aggregated open-loop score. The models are trained for 2×10^4 steps in total, and the results are averaged over 3 evaluations. Averaged open-loop score in black and standard variance in gray.

	τ_0 -pred	v -pred	ϵ -pred
τ_0 -loss: $\mathbb{E}[\ \tau_\theta - \tau_0\ _2^2]$	75.27 $\pm$ 8.53	35.64 $\pm$ 0.97	11.43 $\pm$ 0.45
v -loss: $\mathbb{E}[\ v_{\theta;t} - v_t\ _2^2]$	63.47 $\pm$ 8.58	53.91 $\pm$ 0.87	0.66 $\pm$ 0.40
ϵ -loss: $\mathbb{E}[\ \epsilon_\theta - \epsilon\ _2^2]$	63.78 $\pm$ 8.59	45.24 $\pm$ 2.84	51.07 $\pm$ 4.67

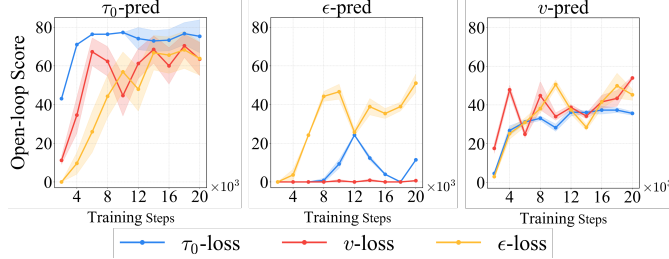


Figure 3 : The learning curve of models with different loss designs. Results are averaged over three evaluations.

varies across different loss space designs. Some models (especially ϵ -pred) generate trajectories with noticeable non-smoothness and irregular jitters, leading to abrupt changes in heading direction and velocity, while the τ_0 -prediction models generate trajectories with better kinematic coherence. This disparity likely stems from denoising dynamics during the final, low-noise steps (Ning et al., 2025). Unlike ϵ - and v -prediction models, which struggle to estimate faint noise signals and consequently generate high-frequency artifacts, data prediction demonstrates superior stability. By directly predicting the trajectory, it effectively suppresses noise to yield smoother, kinematically consistent trajectories. Another thing worth noting is that the ϵ -prediction models trained with τ_0 - and v -loss suffered a complete breakdown. The failure of these two modes can be attributed to the extremely high variance of training objectives in which the noise target is scaled by $1/\alpha_t$. In conclusion, the τ_0 -prediction model with τ_0 -loss yields both fast convergence and high-quality generation, making it a suitable choice for further investigation. Therefore, we choose this design as the default for the following investigative experiments and discussions.

4.2 Trajectory Representation

In the previous section, we identified τ_0 -prediction with τ_0 -loss as a suitable diffusion loss space for planning tasks, which achieves much better learning and open-loop performance. However, when taking a finer-grained inspection on higher-order statistics of generated trajectories, we find that directly using trajectory waypoints as τ_0 could easily result in noticeable jerky movements on the velocity¹ curve, as shown in Figure 5. This indicates that while the model captures the global geometric structure of the trajectory, it fails to enforce local temporal coherence, which could be highly detrimental to closed-loop real-vehicle performance.

A natural fix is to use a delta representation of the trajectory for higher-order supervision: enforcing the model to predict the velocity $\tau_0^v = \{(v_x^l, v_y^l)\}_{l=1}^L$ instead of absolute waypoints $\tau_0^x = \{(x^l, y^l)\}_{l=1}^L$, with the final trajectory recovered via integration at inference. Interestingly, we find empirically that these two trajectory representations have a striking impact on the trajectories generated by diffusion planners. As

¹Note that the term "velocity" in this section refers to physical kinematic velocity rather than the diffusion velocity.

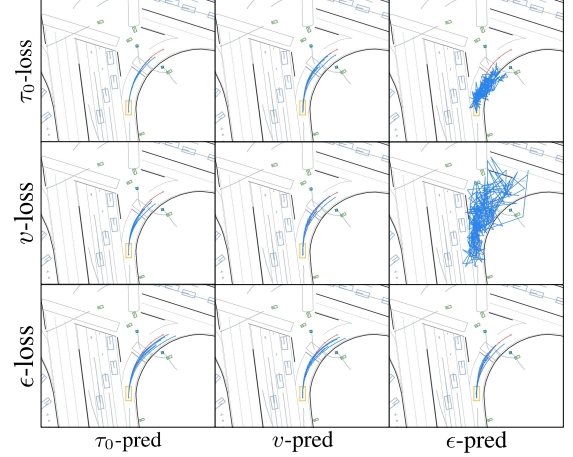


Figure 4 The open-loop visualization of planning trajectories. 6 generations are plotted for each scene. Ego vehicle in yellow, model predictions in blue and ground truth trajectory in red.

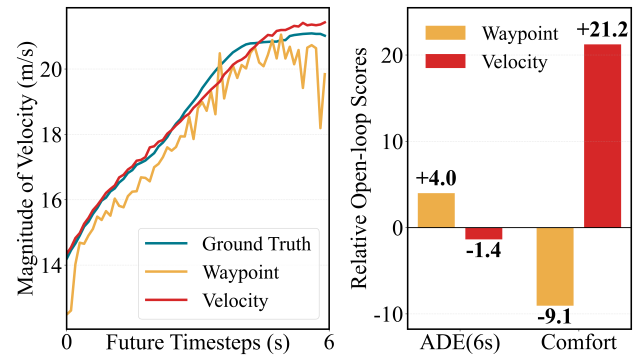


Figure 5 Comparison of prediction quality with waypoint and velocity representations.

shown by the v - t curves and decomposed metrics in Figure 5, velocity-represented trajectories demonstrate smoothness and stability similar to human driving, enjoying a much higher comfort score. By contrast, waypoint-represented trajectories suffer from severe jerky motion, but achieve a superior ADE thanks to better modeling of global geometric structure.

Hybrid Loss. An intuitive idea is to supervise the model with both waypoints and velocity representation simultaneously. However, we find that the magnitude of waypoint coordinates in a trajectories increase greatly along the time-axis, while the velocity representations are more concentratedly distributed, resulting in better numerical stability when learning with a diffusion model.

Therefore, we retain the skeleton of velocity representation, but also incorporate the waypoint supervision through a carefully designed hybrid loss. Specifically, the model outputs the velocity of the planned trajectory, and we compute the L_2 loss on both the directly output velocity and the integrated waypoints:

$$\begin{aligned}\mathcal{L}_{velocity} &= \mathbb{E}_{\tau_0^v, \epsilon, t} \|\tau_\theta^v - \tau_0^v\|_2^2 \\ \mathcal{L}_{waypoints} &= \mathbb{E}_{\tau_0^x, \epsilon, t} \|M\tau_\theta^v \cdot \Delta t - \tau_0^x\|_2^2 \\ &= \mathbb{E}_{\tau_0^v, \epsilon, t} \|M\tau_\theta^v \cdot \Delta t - M\tau_0^v \cdot \Delta t\|_2^2,\end{aligned}\tag{3}$$

where Δt is the time interval of neighboring frames, and M is a lower triangular matrix of ones that integrates the velocity into waypoints. The final hybrid loss is a weighted sum of these two losses:

$$\mathcal{L}_{hybrid} = \mathcal{L}_{velocity} + \omega \cdot \mathcal{L}_{waypoints},\tag{4}$$

where ω is a balancing weight. Moreover, we can theoretically show that this Eq. (4) is also a valid diffusion loss to obtain the correct marginal score function of the data distribution. We first recall the following property of Bregman divergences (Banerjee et al., 2005). Please see Appendix C.2 for proof of Lemma 4.1.

Lemma 4.1 (Lipman et al. (2024)). *For any positive-definite $P \succ 0$, the quadratic form $D_P(u, v) = (u - v)^\top P (u - v)$ is a Bregman divergence. The unique minimizer of the conditional regression objective $\mathbb{E}_{\tau_0^v, \epsilon, t} [D_P(\tau_\theta^v, \tau_0^v)]$ is the conditional expectation $\tau_\theta^{v,*}(\tau_t^v, t) = \mathbb{E}[\tau_0^v | \tau_t^v]$, which by Tweedie’s formula recovers the marginal score function in Eq. (1).*

With Lemma 4.1 in hand, it suffices to verify that the hybrid loss in Eq. (4) takes the form of a positive-definite quadratic Bregman divergence. Please see Appendix C.2 for proof of Theorem 4.2.

Theorem 4.2. *The hybrid loss in Eq. (4) is equivalent to a score matching loss under P -norm:*

$$\mathcal{L}_{hybrid} = \mathbb{E}_{\tau_0^v, \epsilon, t} [\|\tau_\theta^v - \tau_0^v\|_P^2],\tag{5}$$

where $P = I + \Delta t^2 \cdot \omega M^\top M$ is positive-definite. The minimizer of the loss recovers the marginal score function in Eq. (1).

Remark 4.3 (Generality of M). *Theorem 4.2 only requires $M^\top M \succeq 0$, which holds for any matrix M , so $\mathcal{L}_{hybrid}$ remains a valid score matching objective under any choice of kinematic-coupling matrix. The lower-triangular integration matrix in Eq. (3) is the specific instantiation we adopt for its direct physical interpretation as velocity-to-waypoint integration.*

Remark 4.4 (Bias of non-Bregman losses). *The L_1 -norm losses (Liao et al., 2025) (not strictly convex) and auxiliary planning losses (Jiang et al., 2023) (depend on the prediction) used in AD are not Bregman divergences. By Lemma 4.1, their minimizers do not coincide with $\mathbb{E}[\tau_0^v | \tau_t^v]$, yielding biased score estimators.*

Detached Integral. The integration in $\mathcal{L}_{waypoint}$ accumulates gradients along the temporal axis, causing imbalanced supervision across future timesteps. We mitigate this by restricting gradient backpropagation

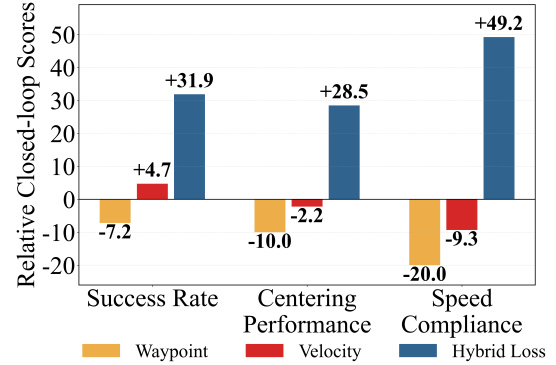


Figure 6 Comparison of waypoint, velocity representations and velocity with hybrid loss, which yields both steadiness and smoothness.

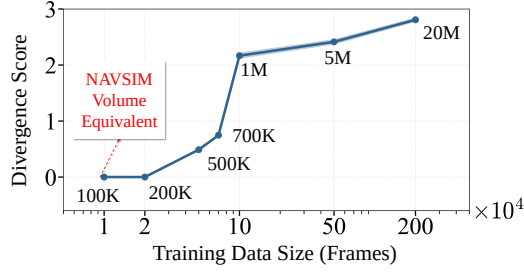


Figure 7 Divergence score evaluated under different training data sizes.

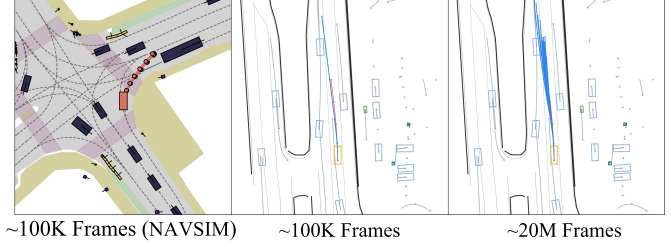


Figure 8 Visualization of planning trajectories obtained under different data sources and dataset sizes.

through the integration to a temporal window of W steps, achieved via stop-gradient on the trajectory history beyond this horizon. This preserves the forward value of $\mathcal{L}_{hybrid}$ and does not alter the minimizer characterized by Theorem 4.2; it serves purely as training stabilization. The matrix-form expression $\hat{\tau}_\theta^x = M_W \tau_\theta^y \Delta t + \text{sg}((M - M_W) \tau_\theta^y \Delta t)$ and full implementation are deferred to Appendix D.3.

Our proposed hybrid loss enables substantial performance improvement in closed-loop real-vehicle testing. As shown in Figure 6, training the diffusion model with the hybrid loss improves all closed-loop metrics, outperforming solely using waypoint and velocity representation by a large margin. This shows that the hybrid loss indeed effectively combines the merits of both representations, capturing the overall vehicle motion trend while preserving the kinematic coherence.

4.3 Multimodal Capability and Data Scaling

Diffusion models are renowned for their multimodal generation capabilities. However, existing diffusion-based planning models (Zheng et al., 2025; Tan et al., 2025) often suffer from severe mode collapse on AD benchmarks (Caesar et al., 2021; Dauner et al., 2024). To investigate the reasons for this discrepancy, as well as examine the multimodal capability and scalability of our proposed framework, we conduct a series of controlled data scaling experiments, spanning from 100K to over 70M real-vehicle training frames. By comparison, existing mainstream E2E AD benchmarks like NAVSIM (Dauner et al., 2024) only contain 100K training data. We evaluate our model’s multimodal generation capability in Figure 7, 8, as well as its open- and closed-loop scaling performance in Figure 9.

Multimodal Generation Capability. We train our model with data from 100K (NAVSIM equivalent) to 20M frames, and use the *Trajectory Divergence* introduced in Section 3.2 to measure multimodal generation. The results are shown in Figure 7. It is observed that the model exhibits negligible multimodal capability when trained on 100K frames of data, consistent with the mode collapse observation in existing AD benchmarks. However, as the training frames increase, the divergence score grows rapidly, suggesting the emergence of multimodal behavior and enhanced generalization performance. This can also be verified by inspecting the generated planning trajectories in Figure 8, that the generated planning trajectories exhibit clear multimodal behavior when trained with 20M frames of data, whereas all trajectories collapse to a single mode when trained on only 100K frames. Our finding is consistent with the theoretical results in Zhang et al. (2023), that diffusion models need sufficient training data for generalization. It also demonstrates that diffusion models can capture multimodal behavior in diverse driving scenarios with proper scaling of training data, even without prior knowledge or bias, such as anchor (Liao et al., 2025) or goal conditioning (Xing et al., 2025).

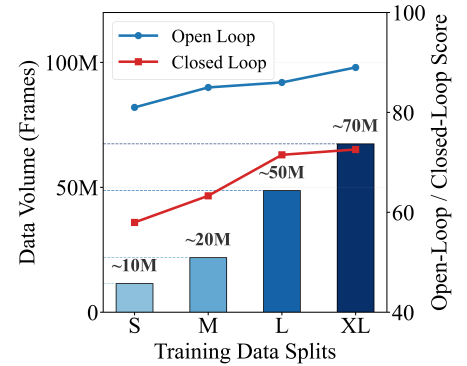


Figure 9 Data scaling experiments. Both open- and closed-loop performance improve as data scales up.

Performance Scaling. We also observe a continuous improvement in both open- and closed-loop performance of our model as the data increases, as shown in Figure 9. By simply increasing the number of training data

from 10M to 70M frames, the model’s closed-loop performance increased by more than 20%, and open-loop increased by 10%, indicating a data scaling property on real vehicles. This demonstrates the huge potential of our methods for large-scale industrial-level applications. However we do not find prominent closed-loop improvement for regression model with data scaling, as shown in Appendix E.1.

5 Reinforcement Learning Post-training

KL-Regularized RL. We adopt standard RL notation conventions, formulating the diffusion-based planner as the policy $\pi(a|s)$. Specifically, the action a corresponds to the generated trajectory $\tau_0^{\mathbf{v}}$, and the state s is the driving scene. The diffusion model takes as input the encoded scene $C := \text{Enc}(s)$ from the scene encoder of Section 3.1. We consider an fine-tuning setting where, at iteration k , we aim to optimize the policy π^k to maximize the expected reward $r(s, a)$, starting from the previous policy π^{k-1} :

$$\max_{\pi^k} \mathbb{E}_{s \sim \mathcal{D}} \left[\mathbb{E}_{a \sim \pi^k(\cdot|s)} [r(s, a)] - \frac{1}{\beta} D_{\text{KL}}(\pi^k(\cdot|s) \parallel \pi^{k-1}(\cdot|s)) \right], \quad (6)$$

where $\mathcal{D}$ is the replay buffer, $\beta > 0$ is the temperature, and $D_{\text{KL}}(p||q) = \mathbb{E}_{x \sim p} [\log(p(x)/q(x))]$. The KL-regularized objective in Eq. (6) provides a closed-form solution for π^k as follows (Nair et al., 2020):

$$\pi^{k*}(a|s) \propto \pi^{k-1}(a|s) \cdot \exp(\beta r(s, a)). \quad (7)$$

To extract the optimal policy in Eq. (7), one approach is to use classifier guidance to steer the diffusion process toward generating high-reward actions during inference (Lu et al., 2023; Zheng et al., 2025). However, this method requires additional inference-time gradient computation, which is very costly and difficult to implement on real vehicles. An alternative approach is to employ a weighted regression loss based on the diffusion imitation loss (Black et al., 2024b; Liang et al., 2026; Zheng et al., 2024). Specialized to our setting— τ_0 -prediction loss on the velocity-based trajectory representation (so that $a \equiv \tau_0^{\mathbf{v}}$ and $\tau_t^{\mathbf{v}} = \alpha_t \tau_0^{\mathbf{v}} + \sigma_t \epsilon$)—it reads

$$\mathcal{L}_{RL} = \mathbb{E}_{t, \epsilon, (s, \tau_0^{\mathbf{v}}) \sim \mathcal{D}} \left[\exp(\beta r(s, \tau_0^{\mathbf{v}})) \parallel \tau_{\theta}^{\mathbf{v};k}(\tau_t^{\mathbf{v}}, t, C) - \tau_0^{\mathbf{v}} \parallel_P^2 \right], \quad (8)$$

where $\tau_{\theta}^{\mathbf{v};k}$ denotes the parameterized diffusion model corresponding to the policy π^k , and the actions $\tau_0^{\mathbf{v}}$ in $\mathcal{D}$ are drawn from the previous-iteration policy π^{k-1} . The weighted regression loss in Eq. (8) only modifies the imitation loss with a weight term, maintaining almost the same computational cost as IL. In contrast, other methods model the denoising process as a multi-step MDP with Gaussian transitions to estimate intermediate log-likelihoods (Black et al., 2024b; Ren et al., 2025), and use RL algorithms like PPO (Schulman et al., 2017) for policy optimization. However, these approaches require storing gradients for all denoising steps during inference and assume a large number of steps to ensure Gaussian transition validity, leading to significantly increased computational cost. We further provide a detailed discussion in Appendix E.2.

RL-Hybrid Loss. To maintain consistency with the hybrid loss defined in Eq. (4) used during imitation pretraining, we introduce the RL-hybrid loss for the post-training phase. This consistency loss design helps mitigate distribution shift, which is a common problem in the offline-to-online fine-tuning setting (Levine et al., 2020). We provide further results in Section 6.2.

$$\mathcal{L}_{RL\text{-}hybrid} = \mathbb{E}_{t, \epsilon, (s, \tau_0^{\mathbf{v}}) \sim \mathcal{D}} \left[\exp(\beta r(s, \tau_0^{\mathbf{v}})) \parallel \tau_{\theta}^{\mathbf{v};k}(\tau_t^{\mathbf{v}}, t, C) - \tau_0^{\mathbf{v}} \parallel_P^2 \right]. \quad (9)$$

Besides, we prove that the hybrid loss can be naturally combined during the RL post-training procedure to optimize the policy, due to its simple formulation as a weighted regression, as shown in Theorem 5.1. Proof see Appendix C.3.

Theorem 5.1. *Optimal action $a \sim \pi^{k*}(a|s)$ in Eq. (7) can be generated by optimizing the weighted diffusion loss in Eq. (9) and solving the diffusion reverse process with the learned $\tau_{\theta}^{\mathbf{v};k*}$.*

In practice, we initialize the policy π^0 for RL post-training using an imitation model pretrained with the hybrid loss in Eq. (4). We first consider the safety reward r_{safety} , which captures collision risk and yields a

Table 2 Main results. represents the highest score. The open-loop score is evaluated through data replay on test datasets, while the closed-loop score is obtained from real-world road testing on a real-vehicle platform.

Model Name	Data Size	Open-Loop Score	Closed-Loop Score		
			Success Rate	Stability Score	Overall Score
Base Model	M	51.07	15.67	0.00	7.83
with τ_0 -loss & τ_0 -pred	M	75.27	22.84	0.00	11.42
+ Velocity Supervision	M	84.38	34.72	9.24	21.98
+ Hybrid Loss	M	85.05	61.88	53.88	57.88
+ Data Scaling	L	86.07	70.59	59.00	64.79
+ Data Scaling (HDP)	XL	88.94	71.24	79.53	75.38
+ RL with safety reward (HDP-RL [†])	-	-	72.89	79.53	76.20
+ RL with multi-rewards (HDP-RL)	-	-	83.49	84.65	84.07

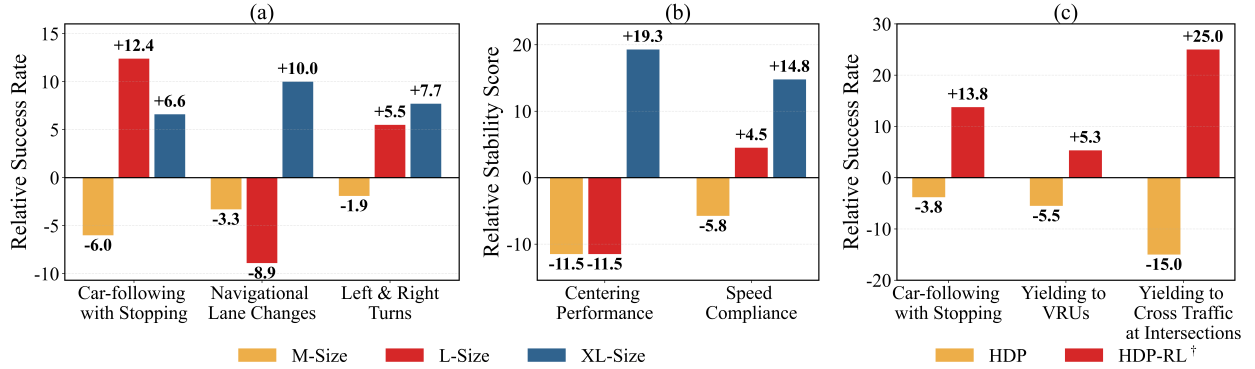


Figure 10 (a) Relative success rate vs. dataset volume in frequently-occurring scenarios; (b) stability score under the same scaling; (c) relative success rate before and after RL fine-tuning in safety-critical scenarios.

safety-aware policy. To further enhance performance, our framework naturally extends to a multi-reward setting: we refine the safety signal into a continuous risk score r_{risk} , and additionally include r_{follow} for car-following comfort and r_{lane} for lane-keeping robustness. The training reward r used in Eq. (9) is then a weighted sum of these components. See Appendix D.4, D.5 for more implementation details.

6 Real-Vehicle Testing Results

Given the above findings and designs for diffusion-based planning methods for E2E AD, we incorporate all the aforementioned innovations into a complete framework, *Hyper Diffusion Planner* (HDP). We begin with the base model introduced in Section 3.1, which uses ϵ -loss and ϵ -pred (Base Model). In Section 4.1, we find that using τ_0 -pred and τ_0 -loss achieves the best trajectory quality among other diffusion loss variants (with τ_0 -loss & τ_0 -pred). Afterwards, in Section 4.2, we discover that using velocity as a supervision signal performs better than using waypoints (+ Velocity Supervision). Combining both improvements, we introduce a hybrid loss function (+ Hybrid Loss). Furthermore, we scale up the dataset in Section 4.3 from the original 20M samples to 50M (+ Data Scaling / L) and 70M (+ Data Scaling / XL), resulting in the final version of HDP. Applying the RL method in Section 5 with only r_{safety} gives HDP-RL[†], while extending to the multi-reward setting gives HDP-RL. Implementation details are deferred to Appendix D.4. As shown in Figure 11, our model handles complex real-world urban driving scenarios well; more cases are provided in Appendix A.

6.1 Imitation Learning Pretraining Lays a Strong Foundation

As shown in Table 2. HDP achieves nearly a 10x improvement in closed-loop performance compared to the base model. For the open-loop setting, during imitation pretraining, it shows that a well-designed loss function and data scaling can steadily improve performance. However, a significant improvement in the closed-loop score is observed only after applying the hybrid loss, highlighting the difference between open-loop

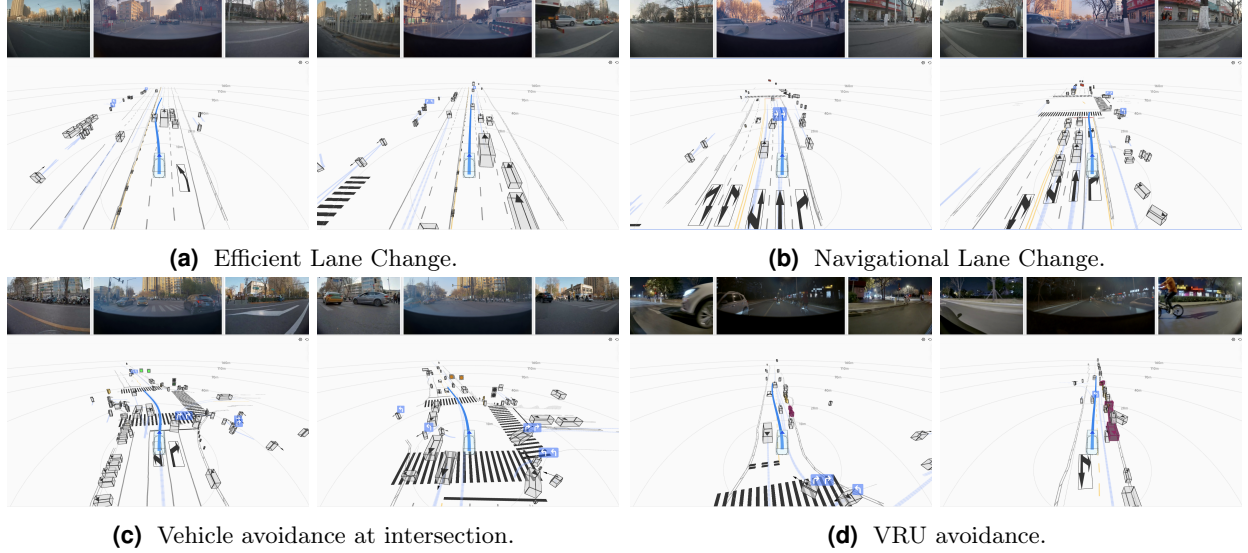


Figure 11 Closed-loop real-vehicle testing, illustrated with two representative frames.

and closed-loop metrics. The key insight is that the hybrid loss greatly enhances stability, allowing the model to have a higher probability of completing each task, thereby achieving an overall noticeable improvement. In addition, when scaling up the data, we show the relative success rate on frequent scenarios, as illustrated in Figure 10. On XL, we observe a trade-off: “Navigational lane change” improves by +18.9 while “Car-following with stopping” drops by 6.2, suggesting that the model reallocates capacity toward more complex behaviors as data scales up. As shown in Figure 10, we also observe a significant gain in the stability score, including both centering performance and speed compliance, indicating that the model better captures the underlying data distribution when trained on larger datasets. To further demonstrate the effectiveness of our method, we provide more comparison against baseline methods under real-vehicle setting in Appendix E.1.

6.2 Reinforcement Learning Post-Training Yields a Reliable Planner

RL with Safety Reward. As shown in Figure 10 and Table 2, $HDP\text{-}RL^\dagger$ substantially improves safety-related metrics. Figure 12 further illustrates the effect of RL post-training through two representative cases. In Figure 12(a), when an oncoming vehicle slightly intrudes into the ego lane, $HDP\text{-}RL^\dagger$ proactively steers laterally away to enlarge the safety margin, whereas HDP maintains a centered trajectory and leaves much narrower clearance. In Figure 12(b), when a vehicle abruptly cuts in from the adjacent lane, $HDP\text{-}RL^\dagger$ smoothly yields space to the cut-in vehicle, while HDP stays closer to the original path and passes the intruder at a notably smaller distance. These two cases consistently show that $HDP\text{-}RL^\dagger$ produces safer and more defensive maneuvers around surrounding traffic participants, confirming the effectiveness of our RL algorithm.

RL with Multi-Rewards. However, with only r_{safety} , the overall performance does not improve substantially. Since real-world AD is inherently a multi-objective problem, we extend the RL stage to the multi-reward setting: our best model $HDP\text{-}RL$ achieves a 10-point improvement (Table 2) over HDP , showing that RL post-training truly works for real-world AD. Moreover, under the same challenging setting, our RL fine-tuning approach

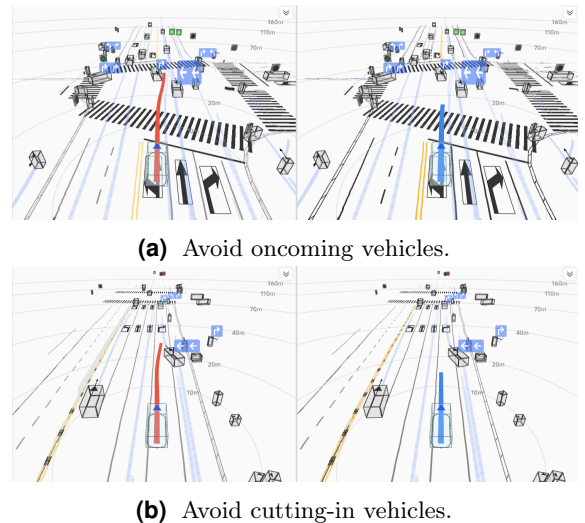


Figure 12 Replay of bad cases before and after post-training. $HDP\text{-}RL^\dagger$ in blue, HDP in red.

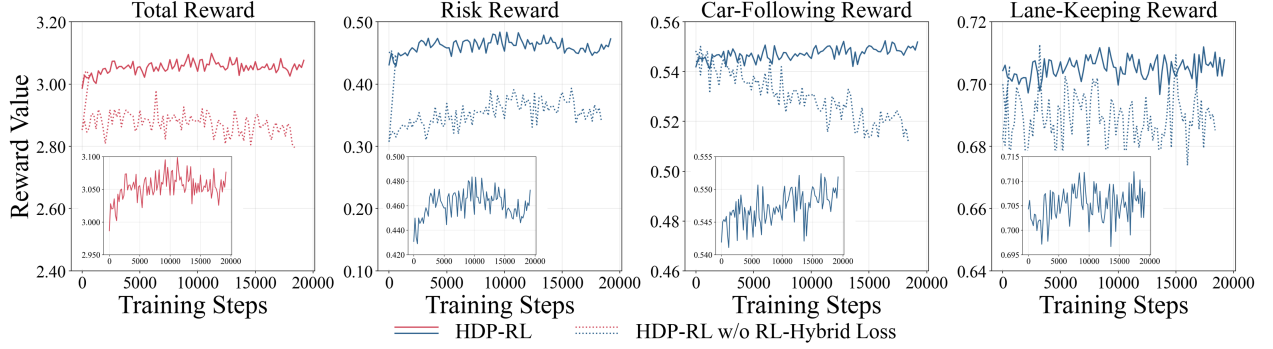


Figure 13 Reward versus training steps: HDP-RL with vs. without the RL-hybrid loss in Eq. (9). The subgraph is the image after rescaling the coordinates.

serves as an effective alternative to baseline RL methods, as shown in the Appendix E.2.

RL-Hybrid Loss Matters. We further show that maintaining alignment with imitation pretraining is necessary when using the hybrid loss, as illustrated in Figure 13. In this setup, our *HDP-RL* employs the RL-hybrid loss, whereas the baseline without RL-hybrid loss uses vanilla weighted regression, as in Eq. (8). Because the policy is imitation-pretrained with the hybrid loss, preserving it during RL is critical: without it, we observe a substantial performance drop at the beginning of RL, which we attribute to distribution shift. By contrast, *HDP-RL* exhibits steadily increasing reward, with multiple reward components improving and advancing the Pareto frontier, while closed-loop performance improves as well. We emphasize that forgoing the hybrid loss may be acceptable when imitation pretraining uses the standard diffusion loss; however, as discussed in Section 4.2, the hybrid loss is important for further improving closed-loop performance.

7 Conclusion

In this paper, we introduce the *Hyper Diffusion Planner (HDP)*, a novel framework that effectively harnesses the generative capabilities of diffusion models for E2E AD. Through comprehensive and controlled studies, we identify key insights into the diffusion loss space, trajectory representation, and data scaling, revealing their critical impact on E2E planning performance. Furthermore, we integrate an effective RL post-training strategy to enhance the safety and robustness of the learned planner. *HDP* is deployed on a real-vehicle platform and validated across 6 urban driving scenarios and 200 km of real-world testing, achieving a notable 10x performance improvement over the base diffusion planner. These results demonstrate that diffusion models, when properly designed and trained, serve as effective and scalable solutions for complex, real-world autonomous driving tasks. Due to space limit, more discussion on limitations and future direction can be found in Appendix F.

Acknowledgments

This work is supported by Xiaomi EV and funding from Wuxi Research Institute of Applied Technologies, Tsinghua University under Grant 20242001120 and the Xiongan AI Institute. Furthermore, we would like to thank Zhiming Li, Huahang Liu, Yan Wang, Xuhui Lu, Xiaojun Ni and Guang Li from Xiaomi EV for their resource support and real vehicle deployment support. We would like to express our gratitude to Quanyun Zhou, Qi Tang, Cheng Chen, Xibin Yue, Qing Li from Xiaomi EV for their valuable discussion. In addition, we thank Jianxiong Li and Zhihao Wang from AIR, Tsinghua University, and Kexin Zheng from The University of Hong Kong for their helpful discussions.

References

- Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.
- Stefano V Albrecht, Cillian Brewitt, John Wilhelm, Balint Gyevnar, Francisco Eiras, Mihai Dobre, and Subramanian Ramamoorthy. Interpretable goal-based prediction and planning for autonomous driving. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1043–1049. IEEE, 2021.
- Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.
- Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.
- James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science*. <https://cdn.openai.com/papers/dall-e-3.pdf>, 2(3):8, 2023.
- Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0 : A vision-language-action flow model for general robot control, 2024a.
- Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, 2024b.
- Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.
- Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.
- Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.
- Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *arXiv preprint arXiv:2306.16927*, 2023.
- Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.
- Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *Pattern Analysis and Machine Intelligence (PAMI)*, 2023.
- Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *The Twelfth International Conference on Learning Representations*, 2023.
- Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(9):10850–10869, 2023.
- Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning. In *Conference on Robot Learning (CoRL)*, 2023a.
- 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, 2023b.
- 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

- Haoyang Fan, Fan Zhu, Changchun Liu, Liangliang Zhang, Li Zhuang, Dong Li, Weicheng Zhu, Jiangtao Hu, Hongye Li, and Qi Kong. Baidu apollo em motion planner, 2018.
- Junru Gu, Chen Sun, and Hang Zhao. Densentnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15303–15312, 2021.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.
- Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023a.
- 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, 2023b.
- Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.
- 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.
- Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023.
- Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 international conference on robotics and automation (ICRA)*, pages 8248–8254. IEEE, 2019.
- Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.
- Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.
- Pengxiang Li, Yinan Zheng, Yue Wang, Huimin Wang, Hang Zhao, Jingjing Liu, Xianyu Zhan, Kun Zhan, and Xianpeng Lang. Discrete diffusion for reflective vision-language-action models in autonomous driving. *arXiv preprint arXiv:2509.20109*, 2025a.
- Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.
- 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. *arXiv preprint arXiv:2506.08052*, 2025b.
- Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, et al. Hydra-mdp: End-to-end multimodal planning with multi-target hydra-distillation. *arXiv preprint arXiv:2406.06978*, 2024a.
- Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: learning bird’s-eye-view representation from lidar-camera via spatiotemporal transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024b.
- Zhiqi Li, Zhiding Yu, Shiyi Lan, Jiahao Li, Jan Kautz, Tong Lu, and Jose M Alvarez. Is ego status all you need for open-loop end-to-end autonomous driving? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14864–14873, 2024c.
- Ruiming Liang, Yinan Zheng, Kexin Zheng, Tianyi Tan, Jianxiong Li, Liyuan Mao, Zhihao Wang, Guang Chen, Hangjun Ye, Jingjing Liu, Jinqiao Wang, and Xianyu Zhan. Dichotomous diffusion policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026.

- 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.
- Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code, 2024. <https://arxiv.org/abs/2412.06264>.
- Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025a.
- Tenglong Liu, Jianxiong Li, Yinan Zheng, Haoyi Niu, Yixing Lan, Xin Xu, and Xianyuan Zhan. Skill expansion and composition in parameter space. In *The Thirteenth International Conference on Learning Representations*, 2025b.
- Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024.
- Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35: 5775–5787, 2022.
- Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. *arXiv preprint arXiv:2304.12824*, 2023.
- Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.
- Mang Ning, Mingxiao Li, Jianlin Su, Haozhe Jia, Lanmiao Liu, Martin Beneš, Albert Ali Salah, and Itir Onal Ertugrul. Dctdiff: Intriguing properties of image generative modeling in the dct space. In *The Forty-Second International Conference on Machine Learning (ICML 2025)*, 2025.
- William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.
- Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.
- Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.
- Allen Z Ren, Justin Lidard, Lars Lien Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, 2025.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- 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.
- Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.
- Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2020.
- Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.
- Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

- Tianyi Tan, Yinan Zheng, Ruiming Liang, Zexu Wang, Kexin Zheng, Jinliang Zheng, Jianxiong Li, Xianyuan Zhan, and Jingjing Liu. Flow matching-based autonomous driving planning with advanced interactive behavior modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.
- Tesla. Tesla ai day 2022. https://www.youtube.com/watch?v=ODSJsviD_SU, 2022.
- Junli Wang, Yinan Zheng, Xueyi Liu, Zebin Xing, Pengfei Li, Guang Li, Kun Ma, Guang Chen, Hangjun Ye, Zhongpu Xia, et al. Meanfuser: Fast one-step multi-modal trajectory generation and adaptive reconstruction via meanflow for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.
- Yan Wang, Wenjie Luo, Junjie Bai, Yulong Cao, Tong Che, Ke Chen, Yuxiao Chen, Jenna Diamond, Yifan Ding, Wenhao Ding, et al. Alpamayo-r1: Bridging reasoning and action prediction for generalizable autonomous driving in the long tail. *arXiv preprint arXiv:2511.00088*, 2025.
- 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. *arXiv preprint arXiv:2503.05689*, 2025.
- Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.
- Huijie Zhang, Jinfan Zhou, Yifu Lu, Minzhe Guo, Peng Wang, Liyue Shen, and Qing Qu. The emergence of reproducibility and generalizability in diffusion models. *arXiv preprint arXiv:2310.05264*, 2023.
- Kexin ZHENG, Lauriane Teyssier, Yinan Zheng, Yu Luo, and Xianyuan Zhan. Towards robust zero-shot reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.
- Yinan Zheng, Jianxiong Li, Dongjie Yu, Yujie Yang, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Safe offline reinforcement learning with feasibility-guided diffusion model. In *The Twelfth International Conference on Learning Representations*, 2024.
- 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.

A Visualization of Real-Vehicle Testing Results

Below we provide more visualizations of real-vehicle testing besides Section 6.



Figure 14 Closed-loop real-vehicle testing results. Each row contains representative frames from the scenario.

B Related Works

Diffusion Model. Diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015) learn complex data distributions through a forward noising process and a learned reverse denoising process. They have achieved remarkable performance in image and video generation (Betker et al., 2023; Croitoru et al., 2023; Liu et al., 2024; Rombach et al., 2022), and have more recently been adopted as expressive policy classes in robotics control (Chi et al., 2023; Black et al., 2024a; Intelligence et al., 2025; Liu et al., 2025a). Their ability to capture multimodal action distributions makes them a natural fit for autonomous driving (AD), where human driving behavior is similarly diverse and stochastic, and has motivated a wave of recent attempts to apply diffusion to AD planning.

End-to-End Autonomous Driving. End-to-end autonomous driving (E2E AD) directly maps raw sensor inputs to trajectory or control outputs through deep neural networks (Bojarski et al., 2016; Chen et al., 2023). Modern E2E systems integrate perception, prediction, and planning into a single differentiable model (Hu et al., 2023b; Jiang et al., 2023), and frequently rely on auxiliary supervision on intermediate representations or trajectories to stabilize training (Bansal et al., 2018). Most existing approaches are evaluated on open-loop datasets (Caesar et al., 2019) or simulation-based benchmarks (Caesar et al., 2021; Dauner et al., 2024; Liu et al., 2025b). However, open-loop metrics are widely acknowledged to be poor proxies for true closed-loop performance on the road (Dauner et al., 2023a; Li et al., 2024c; Zheng et al., 2025), while simulation-based settings exhibit a non-trivial sim-to-real gap. Conclusions drawn from these benchmarks therefore offer limited insight into real-world deployment, which motivates the on-road evaluation pursued in this work.

Diffusion-Based Planning for Autonomous Driving. Zheng et al. (2025) make a pioneering attempt by applying diffusion models to AD planning, although their formulation still relies on vectorized scene representations. Liao et al. (2025) introduce a truncated denoising process; however, this modification disrupts the original diffusion mechanism, and their model continues to depend heavily on pre-defined trajectory anchors for trajectory generation. Wang et al. (2025) develop a vision-language-action (VLA) model with a diffusion-based decoder, leveraging large language priors to generate trajectories with strong reasoning capability. Tan et al. (2025) and Li et al. (2025a) explore alternative parameterizations such as flow matching and discrete diffusion in the AD context. Despite strong benchmark numbers, many of these methods still rely on hand-crafted priors—rule-based post-processing (Fan et al., 2018), pre-defined anchor trajectories (Li et al., 2024a), or explicit goal conditioning (Albrecht et al., 2021; Gu et al., 2021)—to remain competitive, which makes it difficult to attribute the reported gains to diffusion modeling itself and raises concerns about transfer to real-world deployment.

Diffusion-Based Reinforcement Learning. Building upon imitation-pretrained diffusion models, several lines of work apply RL to further align generated outputs with task-specific objectives. One direction optimizes a differentiable reward or value function by back-propagating gradients through the denoising process (Xu et al., 2023; Clark et al., 2023); however, these gradients are typically noisy and lead to unstable training. A second direction (Ren et al., 2025) treats each denoising step as a Gaussian transition and applies policy-gradient methods such as PPO (Schulman et al., 2017) to the diffusion process, but this incurs substantial computational overhead. Li et al. (2025b) extend this idea from Black et al. (2024b) to AD planning, while weighted-regression-based approaches (Lee et al., 2023; Kang et al., 2023; Zheng et al., 2024; ZHENG et al., 2025) offer simpler alternatives that retain the supervised structure of diffusion training. More recently, Liang et al. (2026) propose a dichotomous policy optimization method and fine-tune a one-billion-parameter diffusion-based VLA model for AD with stable training. Despite this rapid progress, most existing methods are validated only in simulation, and their real-vehicle effectiveness under safety constraints and out-of-distribution scenarios remains largely unexplored. Our work fills this gap, showing that an RL post-trained diffusion planner delivers measurable closed-loop gains in real-vehicle deployment.

C Theoretical Analysis

In this section, we provide details on the conversion between different types of diffusion losses and predictions, as well as the proofs of the theorems.

Table 3 Mutual conversions among the three diffusion quantities. Columns index the model parameterization and rows index the loss space. Predicted quantities are denoted with $\hat{\cdot}$ and the network is parameterized by θ . Diagonal entries are the canonical objectives; off-diagonal entries first transform the network output into the target space and then compute the squared error.

	τ_0 -pred.	v_t -pred.	ϵ -pred.
τ_0 -loss: $\mathbb{E}\ \hat{\tau}_0 - \tau_0\ ^2$	$\hat{\tau}_0 = \tau_\theta$	$\hat{\tau}_0 = \alpha_t \tau_t - \sigma_t v_{\theta;t}$	$\hat{\tau}_0 = (\tau_t - \sigma_t \epsilon_\theta)/\alpha_t$
v_t -loss: $\mathbb{E}\ \hat{v}_t - v_t\ ^2$	$\hat{v}_t = (\alpha_t \tau_t - \tau_\theta)/\sigma_t$	$\hat{v}_t = v_{\theta;t}$	$\hat{v}_t = (\epsilon_\theta - \sigma_t \tau_t)/\alpha_t$
ϵ -loss: $\mathbb{E}\ \hat{\epsilon} - \epsilon\ ^2$	$\hat{\epsilon} = (\tau_t - \alpha_t \tau_\theta)/\sigma_t$	$\hat{\epsilon} = \sigma_t \tau_t + \alpha_t v_{\theta;t}$	$\hat{\epsilon} = \epsilon_\theta$

C.1 Diffusion Loss Space

Following the standard diffusion formulation (Ho et al., 2020; Song et al., 2021), the forward process diffuses a clean trajectory τ_0 into a noised sample

$$\tau_t = \alpha_t \tau_0 + \sigma_t \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \quad (10)$$

where $\{\alpha_t, \sigma_t\}$ is a predefined noise schedule and $t \in [0, T]$ indexes the diffusion timestep. A diffusion model is then trained to predict one of three equivalent quantities: the clean trajectory τ_0 , the injected noise ϵ , or the velocity $v_t = \alpha_t \epsilon - \sigma_t \tau_0$. Given $(t, \alpha_t, \sigma_t, \tau_t)$, these three quantities are linearly inter-convertible, so any one of them is sufficient to recover the other two.

This convertibility decouples what the network predicts (the **parameterization**) from the space in which the loss is computed (the **loss space**), yielding a 3×3 design grid. The diagonal entries correspond to the canonical τ_0 -, v_t -, and ϵ -objectives, where the parameterization and the loss space coincide, while the off-diagonal entries first map the network output to the target space via the closed-form transforms in Table 3 and then compute the squared error. As an example, parameterizing the network as a clean-trajectory predictor τ_θ but computing the loss in noise space gives

$$\mathcal{L} = \mathbb{E}_{\tau_0, t, \epsilon} \left\| \frac{\tau_t - \alpha_t \tau_\theta}{\sigma_t} - \epsilon \right\|^2. \quad (11)$$

Although the nine combinations share the same underlying score-matching objective, they are not numerically equivalent during training: each combination induces a different implicit weighting over noise levels. As a result, the choice of (parameterization, loss space) materially affects optimization stability, sample quality, and the relative emphasis the model places on coarse vs. fine-grained trajectory details. We therefore treat parameterization and loss space as two orthogonal design axes and study them jointly in our experiments.

C.2 Proofs of Lemma 4.1 and Theorem 4.2

In this subsection, we prove that the hybrid loss in Eq. (4) is a valid diffusion score-matching objective whose unique minimizer recovers the marginal score. We first establish a general Bregman-divergence lemma (Lemma 4.1), then apply it to identify the P -norm structure of the hybrid loss and its score-matching minimizer (Theorem 4.2).

Lemma 4.1 (Lipman et al. (2024)). *For any positive-definite $P \succ 0$, the quadratic form $D_P(u, v) = (u - v)^\top P(u - v)$ is a Bregman divergence. The unique minimizer of the conditional regression objective $\mathbb{E}_{\tau_0^\mathbf{y}, \epsilon, t} [D_P(\tau_\theta^\mathbf{y}, \tau_0^\mathbf{y})]$ is the conditional expectation $\tau_\theta^{\mathbf{y},*}(\tau_t^\mathbf{y}, t) = \mathbb{E}[\tau_0^\mathbf{y} | \tau_t^\mathbf{y}]$, which by Tweedie’s formula recovers the marginal score function in Eq. (1).*

Proof. (i) **D_P is a Bregman divergence.** The potential $\Phi_P(u) = u^\top P u$ is strictly convex (since $P \succ 0$) with $\nabla \Phi_P(u) = 2Pu$, and direct expansion gives

$$\Phi_P(u) - \Phi_P(v) - \langle \nabla \Phi_P(v), u - v \rangle = (u - v)^\top P(u - v) = D_P(u, v).$$

(ii) **Minimizer.** Conditioning on $\tau_t^\mathbf{v}$ and expanding the quadratic form in $f := \tau_\theta^\mathbf{v}(\tau_t^\mathbf{v}, t)$,

$$\mathbb{E}[D_P(f, \tau_0^\mathbf{v}) \mid \tau_t^\mathbf{v}] = f^\top P f - 2f^\top P \mathbb{E}[\tau_0^\mathbf{v} \mid \tau_t^\mathbf{v}] + C(\tau_t^\mathbf{v}),$$

with C independent of f . The first-order condition yields $\tau_\theta^{\mathbf{v},*}(\tau_t^\mathbf{v}, t) = \mathbb{E}[\tau_0^\mathbf{v} \mid \tau_t^\mathbf{v}]$, and the Hessian $2P \succ 0$ makes this the unique minimizer.

(iii) **Connection to the score.** By Tweedie's formula applied to $\tau_t^\mathbf{v} = \alpha_t \tau_0^\mathbf{v} + \sigma_t \epsilon$,

$$\mathbb{E}[\tau_0^\mathbf{v} \mid \tau_t^\mathbf{v}] = \frac{1}{\alpha_t} (\tau_t^\mathbf{v} + \sigma_t^2 \nabla_{\tau_t^\mathbf{v}} \log q_t(\tau_t^\mathbf{v})),$$

which is in one-to-one correspondence with the marginal score in Eq. (1). $\square$

Theorem 4.2. *The hybrid loss in Eq. (4) is equivalent to a score matching loss under P -norm:*

$$\mathcal{L}_{\text{hybrid}} = \mathbb{E}_{\tau_0^\mathbf{v}, \epsilon, t} [\|\tau_\theta^\mathbf{v} - \tau_0^\mathbf{v}\|_P^2],$$

where $P = I + \Delta t^2 \cdot \omega M^\top M$ is positive-definite. The minimizer of the loss recovers the marginal score function in Eq. (1).

Proof. (i) $\mathcal{L}_{\text{hybrid}}$ as a P -norm. The two terms of the hybrid loss admit matrix forms (using linearity of M for the second):

$$\mathcal{L}_{\text{velocity}} = \mathbb{E}[(\tau_\theta^\mathbf{v} - \tau_0^\mathbf{v})^\top I (\tau_\theta^\mathbf{v} - \tau_0^\mathbf{v})], \quad \mathcal{L}_{\text{waypoints}} = \mathbb{E}[\Delta t^2 (\tau_\theta^\mathbf{v} - \tau_0^\mathbf{v})^\top M^\top M (\tau_\theta^\mathbf{v} - \tau_0^\mathbf{v})].$$

Plugging into Eq. (4),

$$\mathcal{L}_{\text{hybrid}} = \mathbb{E}[(\tau_\theta^\mathbf{v} - \tau_0^\mathbf{v})^\top (I + \omega \Delta t^2 M^\top M) (\tau_\theta^\mathbf{v} - \tau_0^\mathbf{v})] = \mathbb{E}[\|\tau_\theta^\mathbf{v} - \tau_0^\mathbf{v}\|_P^2],$$

with $P := I + \omega \Delta t^2 M^\top M$.

(ii) $P \succ 0$. P is symmetric, and for any $z \neq 0$,

$$z^\top P z = \|z\|_2^2 + \omega \Delta t^2 \|M z\|_2^2 > 0.$$

(iii) **Minimizer.** With $P \succ 0$, $\mathcal{L}_{\text{hybrid}}$ matches the conditional regression objective in Lemma 4.1, so its minimizer is $\tau_\theta^{\mathbf{v},*}(\tau_t^\mathbf{v}, t) = \mathbb{E}[\tau_0^\mathbf{v} \mid \tau_t^\mathbf{v}]$, which by Tweedie equals the marginal score in Eq. (1). $\square$

C.3 Proof of Theorem 5.1

We formalize the RL post-training as a Markov Decision Process (MDP) (Sutton et al., 1998) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where the state $s \in \mathcal{S}$ is the driving scene, the action $a \equiv \tau_0^\mathbf{v} \in \mathcal{A}$ is the velocity-parameterized trajectory, and the policy $\pi^k(a \mid s)$ is parameterized by the diffusion model $\tau_\theta^{\mathbf{v};k}$ at iteration k . We update π^k from the previous policy π^{k-1} by solving the KL-regularized objective in Eq. (6):

$$\max_{\pi^k} \mathbb{E}_{s \sim \mathcal{D}} \left[\mathbb{E}_{a \sim \pi^k(\cdot \mid s)} [r(s, a)] - \frac{1}{\beta} D_{\text{KL}}(\pi^k(\cdot \mid s) \parallel \pi^{k-1}(\cdot \mid s)) \right],$$

subject to the simplex constraint $\int_a \pi^k(a \mid s) da = 1$ for all s . With state density $d(s)$ from $\mathcal{D}$ and multiplier α_s for the simplex constraint, the Lagrangian is

$$\begin{aligned} \mathcal{L}(\pi^k, \alpha) &= \int_s d(s) \left[\int_a \pi^k(a \mid s) r(s, a) da - \frac{1}{\beta} \int_a \pi^k(a \mid s) \log \frac{\pi^k(a \mid s)}{\pi^{k-1}(a \mid s)} da \right] ds \\ &\quad + \int_s \alpha_s \left(\int_a \pi^k(a \mid s) da - 1 \right) ds. \end{aligned}$$

Taking the derivative with respect to $\pi^k(a \mid s)$ and setting it to zero yields

$$r(s, a) - \frac{1}{\beta} \left(\log \frac{\pi^k(a \mid s)}{\pi^{k-1}(a \mid s)} + 1 \right) + \frac{\alpha_s}{d(s)} = 0.$$

Solving for π^k and absorbing the a -independent term $\exp(\beta\alpha_s/d(s) - 1)$ into the normalization gives the closed-form optimum (Nair et al., 2020)

$$\pi^{k^*}(a | s) \propto \pi^{k-1}(a | s) \cdot \exp(\beta r(s, a)),$$

matching Eq. (7). Next, we show that the hybrid loss can be expanded into a reward-weighted diffusion loss whose minimizer recovers this π^{k^*} .

Theorem 5.1. *Optimal action $a \sim \pi^{k^*}(a|s)$ in Eq. (7) can be generated by optimizing the weighted diffusion loss in Eq. (9) and solving the diffusion reverse process with the learned $\tau_\theta^{\mathbf{v};k^*}$.*

Proof. Here we consider the velocity of future trajectory as action $a = \tau_0^{\mathbf{v}}$. To prove that we can sample from the optimal policy in Eq. 7, we only need to show that the reward-weighted objective in Eq. 9 is equivalent to the corresponding score matching objective of the optimal policy distribution. For simplicity, we omit the state condition in the derivation.

$$\begin{aligned} & \mathbb{E}_{\tau_0^{\mathbf{v}} \sim \pi^{k-1}, \epsilon \sim p_\epsilon(\epsilon), t \sim p_t(t)} \left[\exp(\beta r) \|\tau_\theta^{\mathbf{v};k} - \tau_0^{\mathbf{v}}\|_P^2 \right] \\ &= \int_{\tau_0^{\mathbf{v}}, \epsilon, t} \exp(\beta r) \|\tau_\theta^{\mathbf{v};k} - \tau_0^{\mathbf{v}}\|_P^2 \cdot \pi^{k-1}(\tau_0^{\mathbf{v}}) p_\epsilon(\epsilon) p_t(t) d\epsilon dt d\tau_0^{\mathbf{v}} \\ &= Z \int_{\tau_0^{\mathbf{v}}, \epsilon, t} \|\tau_\theta^{\mathbf{v};k} - \tau_0^{\mathbf{v}}\|_P^2 \cdot \frac{\exp(\beta r) \pi^{k-1}(\tau_0^{\mathbf{v}})}{Z} \\ & \quad \cdot p_\epsilon(\epsilon) p_t(t) d\epsilon dt d\tau_0^{\mathbf{v}} \\ &= Z \int_{\tau_0^{\mathbf{v}}, \epsilon, t} \|\tau_\theta^{\mathbf{v};k} - \tau_0^{\mathbf{v}}\|_P^2 \cdot \pi^{k^*}(\tau_0^{\mathbf{v}}) \cdot p_\epsilon(\epsilon) p_t(t) d\epsilon dt d\tau_0^{\mathbf{v}} \\ &= Z \mathbb{E}_{\tau_0^{\mathbf{v}} \sim \pi^{k^*}, \epsilon \sim p_\epsilon(\epsilon), t \sim p_t(t)} \left[\|\tau_\theta^{\mathbf{v};k} - \tau_0^{\mathbf{v}}\|_P^2 \right] \end{aligned} \tag{12}$$

where $Z = \int_{\tau_0^{\mathbf{v}}} \exp(\beta r) \pi^{k-1}(\tau_0^{\mathbf{v}}) d\tau_0^{\mathbf{v}}$ is the normalizing constant. This indicates that the reward-weighted objective is equivalent to the standard score matching objective over the optimal policy, scaled by a constant that does not change the minimizer. By Theorem 4.2, this minimizer recovers the marginal score of π^{k^*} . Consequently, the diffusion reverse process driven by the learned $\tau_\theta^{\mathbf{v};k^*}$ samples from $\pi^{k^*}(a | s)$, completing the proof. $\square$

D Experimental Details

In this section, we provide the experimental details, including the metrics used for open-loop and closed-loop evaluation, as well as the implementation details of imitation learning pre-training and reinforcement learning post-training.

D.1 Evaluation Metric Design

We consider two types of evaluation metrics: open-loop metrics for assessing trajectory quality, and closed-loop metrics for evaluating performance during real-vehicle testing.

Open-Loop Metrics. To perform a comparable open-loop evaluation, we consider widely adopted open-loop measures and compute a final score as the aggregated open-loop metric. The diffusion model exhibits multi-modal behavior during trajectory generation. We generate 6 trajectories for evaluation. To reduce randomness, we compute the minADE (minimum average Euclidean distance between each predicted trajectory and the ground truth across all waypoints) and minFDE (minimum Euclidean distance between the final waypoint of each predicted trajectory and the ground truth) and calculate the corresponding scores as follows:

$$\begin{aligned} S_{ADE} &= 100 \times \text{Clip}\left(1 - \frac{\min ADE}{\text{Thresh}_{ADE}}, 0, 1\right) \\ S_{FDE} &= 100 \times \text{Clip}\left(1 - \frac{\min FDE}{\text{Thresh}_{FDE}}, 0, 1\right) \end{aligned} \tag{13}$$

in which we clip and scale the corresponding scores into $[0, 100]$. To evaluate the comfort and smoothness of the model’s generated trajectories, we computed a comfort score as a combination of average acceleration (Acc) and jerk ($Jerk$), and calculated the average score over the 6 trajectories:

$$\begin{aligned} Cost &= \frac{1}{N_1} \sum_{i=1}^{N_1} (Cost_{Acc} \times Acc + Cost_{Jerk} \times Jerk) \\ S_{Comfort} &= 100 \times Clip(1 - \frac{Cost}{Thresh_{Comfort}}, 0, 1) \end{aligned} \quad (14)$$

The final aggregated open-loop score is computed as a weighted sum of the previous metrics scaled by the average collision rate (CR):

$$\begin{aligned} S_{Open-Loop} &= (1 - CR) \times (\sum_{m \in \mathcal{M}} \omega_m S_m) \\ \mathcal{M} &= \{ADE, FDE, Comfort\} \end{aligned} \quad (15)$$

In addition, to evaluate the multi-modal generation ability of the model, we consider the divergence of each rollout with 64 generations, measured by the average distance of trajectory endpoints to their geometrical center:

$$Divergence\ Score = \frac{1}{N_2} \sum_{i=1}^{N_2} \|P_i^L\| - \frac{1}{N_2} \sum_{i=1}^{N_2} \|P_i^L\|_2 \quad (16)$$

where P_i^L is the endpoint of the i -th trajectory. The choice of hyperparameters can be found in Table 4.

Table 4 Hyperparameters for the open-loop metrics.

Hyperparameter	Value	Hyperparameter	Value
$Thresh_{ADE}$	4	$Cost_{jerk}$	0.5
$Thresh_{FDE}$	8	ω_{ADE}	0.35
$Thresh_{Comfort}$	200	ω_{FDE}	0.25
$Cost_{Acc}$	1.0	$\omega_{Comfort}$	0.40

Table 5 Hyperparameters for the closed-loop metrics.

Hyperparameter	Value	Hyperparameter	Value
w_1	0.1	w_5	0.1
w_2	0.25	w_6	0.2
w_3	0.25	$Thresh_{center}$	40
w_4	0.1	$Thresh_{speed}$	40

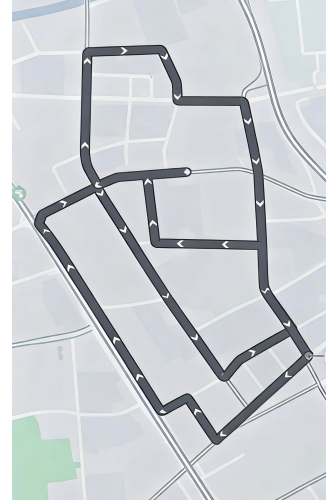


Figure 15 Real vehicle test route.

Closed-Loop Metrics. We provide two types of closed-loop metrics: the success rate and the stability score. To ensure a fair comparison, we use a fixed route, as shown in Figure (15), for each model, and each model performs two loops. During the test, we mark specific scenarios, including starting maneuvers (s_1), car-following with stopping (s_2), navigational lane changes (s_3), yielding to VRUs (s_4), yielding to cross traffic at intersections (s_5), and left and right turns (s_6). For each task, a trial is considered a failure if a human takeover occurs; otherwise, it is considered a success. The success rate for each scenario is then calculated. To obtain an overall success rate, we compute a weighted mean across all six scenarios, assigning higher weights to more frequent scenarios for a more accurate evaluation.

$$\begin{aligned} Success\ Rate &= w_1 s_1 + w_2 s_2 + w_3 s_3 \\ &\quad + w_4 s_4 + w_5 s_5 + w_6 s_6 \end{aligned} \quad (17)$$

Moreover, we also consider the stability score. Unlike the success rate, this metric is not constrained to specific scenarios. Instead, we evaluate abnormal centering behavior and abnormal speeds, such as driving too slowly

or too fast. We record the occurrences of these abnormal behaviors and normalize the counts per 100 km (k_{center}, k_{speed}). Afterward, we calculate the scores for centering performance and speed compliance:

$$\begin{aligned} S_{center} &= 100 \times \text{Clip}\left(1 - \frac{k_{center}}{\text{Thresh}_{center}}, 0, 1\right) \\ S_{speed} &= 100 \times \text{Clip}\left(1 - \frac{k_{speed}}{\text{Thresh}_{speed}}, 0, 1\right) \end{aligned} \quad (18)$$

Finally, we calculate the overall score using the average of the centering performance and speed compliance scores. The choice of hyperparameters can be found in Table 4.

D.2 Scene Encoder Details

We consider an E2E AD system that directly processes multi-modal sensory inputs (camera images and LiDAR point clouds) to generate planning trajectories (Chen et al., 2023; Hu et al., 2023b). Since our contribution centers on diffusion-based planning, we adopt an in-house validated perception backbone as the scene encoder Enc rather than re-engineering perception from scratch. Heterogeneous sensory streams are first compressed into a unified Bird’s Eye View (BEV) feature map (Li et al., 2024b), exposing a common spatial frame for downstream perception and planning. On top of the BEV features, we employ two distinct sets of transformer-based queries: Object Detection (OD) tokens for vehicle and pedestrian localization, and Lane Detection (LD) tokens for road structure understanding; the encoder is pretrained on these perception tasks to provide a solid initialization before being plugged into the planner. At planning time, the OD and LD tokens are concatenated with Navi tokens (encoding high-level navigation commands) and further fused via self-attention blocks, yielding the latent condition $C := \text{Enc}(s)$ consumed by the diffusion decoder.

D.3 Hybrid Loss Implementations

The pseudocode for hybrid loss with detach is shown in Algorithm 1, implemented in torch. In practice, we set $W = L - 1$. We further illustrate the stop-gradient operation in hybrid loss with a small example: $L = 6$ time steps and detach window $W = 3$. The matrix-form expression

$$\hat{\tau}_{\theta}^x = M_W \tau_{\theta}^v \Delta t + \text{sg}((M - M_W) \tau_{\theta}^v \Delta t) \quad (19)$$

Algorithm 1 Hybrid Loss with Detach

```
def detached_integral(v, W, dt):
    # v: velocity of future trajectory
    # W: gradient detach window size
    # dt: time interval
    wpt_sg = torch.cumsum(v.detach()) * dt
    shift_sg = torch.roll(wpt_sg, shifts=W)
    shift_sg[:W] = 0

    wpt = torch.cumsum(v) * dt
    shift = torch.roll(wpt, shifts=W)
    shift[:W] = 0

    return wpt + shift_sg - shift

def hybrid_loss(pred_v, gt_v, W, omega):
    # omega: loss balancing weight
    # pred_v: predicted future velocity
    # gt_v: ground truth future velocity
    l_v = (pred_v - gt_v) ** 2
    l_wpt = (detached_integral(pred_v, W) - torch.cumsum(gt_v)) ** 2
    return l_v + omega * l_wpt
```

Algorithm 2 RL-Hybrid Loss

```

def rl_hybrid_loss(r, beta, pred_v, gt_v, W, omega):
    # r: per- candidate reward
    # beta: temperature
    # pred_v, gt_v, W, omega: same as Algorithm 1

    r_n = (r - r.mean()) / (r.std() + 1e-6)
    weight = torch.exp(beta * r_n).detach()

    return weight * hybrid_loss(pred_v, gt_v, W, omega)
  
```

The integration matrix M and its W -banded version M_W are

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad M_W = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

The first $W = 3$ rows of M_W coincide with M (early waypoints fall entirely within the W -step horizon), whereas later rows retain only the most recent W entries.

Forward pass. Write the predicted velocities as $\tau_\theta^\mathbf{v} = (v^1, v^2, \dots, v^6)^\top$. The two branches in Eq. (19) decompose as

$$M_W \tau_\theta^\mathbf{v} \Delta t = \begin{pmatrix} v^1 \\ v^1 + v^2 \\ v^1 + v^2 + v^3 \\ v^2 + v^3 + v^4 \\ v^3 + v^4 + v^5 \\ v^4 + v^5 + v^6 \end{pmatrix} \Delta t, \quad (M - M_W) \tau_\theta^\mathbf{v} \Delta t = \begin{pmatrix} 0 \\ 0 \\ 0 \\ v^1 \\ v^1 + v^2 \\ v^1 + v^2 + v^3 \end{pmatrix} \Delta t.$$

Their sum is the standard waypoint integration $M \tau_\theta^\mathbf{v} \Delta t = (v^1, v^1 + v^2, \dots, \sum_{j=1}^6 v^j)^\top \Delta t$. Since $\text{sg}(\cdot)$ is the identity in forward, $\hat{\tau}_\theta^\mathbf{x} = M \tau_\theta^\mathbf{v} \Delta t = \tau_\theta^\mathbf{x}$ numerically: *the loss value is the same as without detach.*

Backward pass. Only the $M_W \tau_\theta^\mathbf{v} \Delta t$ branch contributes to the gradient (the second branch is wrapped in sg). The waypoint loss $\mathcal{L}_{\text{waypoints}} = \sum_{i=1}^L (\hat{x}_\theta^i - x_0^i)^2$ therefore yields

$$\frac{\partial \mathcal{L}_{\text{waypoints}}}{\partial v^k} = 2\Delta t \sum_{i: (M_W)_{ik}=1} (\hat{x}_\theta^i - x_0^i) = 2\Delta t \sum_{i=k}^{\min(k+W-1, L)} (\hat{x}_\theta^i - x_0^i). \quad (20)$$

Concretely, for $L = 6, W = 3$ the gradient of $\mathcal{L}_{\text{waypoints}}$ w.r.t. each velocity component picks up residuals only from the listed waypoints:

component	full integration (M)	detached ($M_W, W = 3$)
v^1	$\{1, 2, 3, 4, 5, 6\}$	$\{1, 2, 3\}$
v^2	$\{2, 3, 4, 5, 6\}$	$\{2, 3, 4\}$
v^3	$\{3, 4, 5, 6\}$	$\{3, 4, 5\}$
v^4	$\{4, 5, 6\}$	$\{4, 5, 6\}$
v^5	$\{5, 6\}$	$\{5, 6\}$
v^6	$\{6\}$	$\{6\}$

Without detach, v^1 accumulates residuals from *all six* future waypoints, whereas v^6 only sees one—the imbalance grows with L . With detach, each v^k accumulates at most W residuals, evening out the gradient magnitudes across time steps.

Theoretical compatibility. The decomposition in Eq. (19) only modifies the backward pass; the forward value of $\hat{\tau}_\theta^x$, and hence of $\mathcal{L}_{\text{hybrid}}$, is identical to the un-detached case. Consequently, the minimizer characterized by Theorem 4.2 is unchanged: the global minimizer remains the marginal score function. The detach merely reshapes the gradient field for training stabilization.

D.4 Implementation Details

Imitation Learning. Based on the content in Section 6, we provide the following details of the experimental setup. We adopt the variance-preserving (VP) noise schedule following Zheng et al. (2025) and use 6 sampling steps for efficient generation. Training was conducted using 64 NVIDIA H20 GPUs, with a batch size of 160 per GPU over 10 epochs, with a warmup phase. We use AdamW optimizer with a learning rate of 5×10^{-4} , weight decay of 0.01. We report the other detailed setup in Table 6. After being well trained, our models are deployed on a real vehicle platform for real-world closed-loop testing. Specifically, the model is first converted to the ONNX format and then optimized using TensorRT’s inference compiler to enable hardware-accelerated execution. Furthermore, for multi-step inference, we follow the approach used by Zheng et al. (2025), which employs the DPM-Solver (Lu et al., 2022) to accelerate the sampling process, achieving a final inference speed that easily meets the 10Hz requirement. It is worth noting that we apply only a light post-processing smoothing step after the model output, ensuring that the evaluation accurately reflects the model’s inherent performance.

Reinforcement Learning. Given the safety risks of online RL on real vehicles (Kendall et al., 2019) and the high computational cost of world-model-based simulators (Agarwal et al., 2025; Hu et al., 2023a), we train with a non-reactive pseudo-closed-loop simulator (Dauner et al., 2024) built on real-world datasets, while still evaluating the trained model on real vehicles. In this setup, neighboring vehicles replay logged behaviors, while our model generates planning trajectories. To achieve stable training using Eq. (9), we apply reward group normalization (Shao et al., 2024) to obtain an appropriate numerical range for weighting. Additionally, we discard samples in which all actions receive identical rewards to improve learning effectiveness. Finally, we employ Exponential Moving Average (EMA) for policy updates to further enhance stability. More details about rewards design are shown in Appendix D.5.

D.5 Reward Function Details

We provide concrete forms of the four rewards used in Section 5. All rewards return a scalar in $[0, 1]$, evaluated on the candidate trajectory over the planning horizon of L steps. The single-reward baseline uses r_{safety} alone; the multi-reward setting replaces it with r_{risk} and adds r_{follow} and r_{lane} .

Safety Reward r_{safety} . Collisions between the ego and each neighboring vehicle are detected via the Separating Axis Theorem (SAT) applied to their oriented bounding boxes at every future timestep l . The per-step collision indicator is $c_l = 1.0$ for active (head-on or lateral) collisions, $c_l = 0.3$ for rear-end collisions (attenuated to mitigate non-reactive simulator artifacts), and $c_l = 0$ otherwise. The trajectory-level reward is

$$r_{\text{safety}} = 1 - \max_{l=1, \dots, L} c_l. \quad (21)$$

Risk Reward r_{risk} . A continuous, near-miss-aware refinement of r_{safety} . At each timestep l we compute three sub-scores in $[0, 1]$ from time-to-collision (TTC), time-headway (THW), and occupancy distance to static/uncertain regions. Each sub-score is obtained by passing the raw quantity through a speed-adaptive shaping function: 1 when the geometry is comfortably safe, decaying to 0 as the configuration becomes critical. We aggregate conservatively—taking the most pessimistic value across both time and signals—yielding

$$r_{\text{risk}} = \min_{l=1, \dots, L, m \in \{\text{TTC}, \text{THW}, \text{OCC}\}} s_l^m. \quad (22)$$

Table 6 Hyperparameters of *HDP* / *HDP-RL*

Type	Parameter	Symbol	Value
IL	Num. block	-	6
	Dim. hidden layer	-	256
	Num. multi-head	-	8
	Hybrid loss weight	ω	0.1
RL	Group size	-	32
	Temperature	β	1.0
	EMA	-	0.05
	Risk reward weight	λ_{risk}	1.0
	Car-following reward weight	λ_{follow}	3.0
	Lane-keeping reward weight	λ_{lane}	2.5

The same rear-end attenuation as in r_{safety} is applied so that artifacts of the non-reactive simulator are not over-penalized.

Car-Following Reward r_{follow} . When a leader vehicle is present, this reward encourages human-like car-following, following the spirit of classical ACC heuristics. At each timestep l we score the ego on four intuitive aspects, each normalized to $[0, 1]$:

- *time gap* s_l^{gap} : how close the current headway (in seconds) is to a speed-dependent ideal – shorter at low speed, longer at high speed;
- *spacing* s_l^{dist} : how close to the target vehicle, clipped to a safe min/max range;
- *speed match* s_l^{spd} : how well the ego speed tracks the leader’s speed;
- *comfort* s_l^{cmf} : a penalty on longitudinal acceleration or braking that exceeds comfortable bounds.

The four scores are averaged over time and across aspects:

$$r_{\text{follow}} = \frac{1}{4L} \sum_{l=1}^L \left(s_l^{\text{gap}} + s_l^{\text{dist}} + s_l^{\text{spd}} + s_l^{\text{cmf}} \right). \quad (23)$$

Lane-Keeping Reward r_{lane} . This reward encourages trajectories to stay close to the lane centerline during lane-keeping, improving lateral robustness. For each predicted waypoint we measure its perpendicular distance to the nearest lane centerline and convert it into a score $s_l^{\text{ctr}} \in [0, 1]$ that is 1 on the centerline and decays linearly to 0 once the offset exceeds roughly half a lane width. The reward is the temporal average:

$$r_{\text{center}} = \frac{1}{L} \sum_{l=1}^L s_l^{\text{ctr}}. \quad (24)$$

To avoid penalizing intentional lateral maneuvers, we mask out samples whose expert reference is off-lane or exhibits lane-change behavior.

Total Training Reward. The reward r used in Eq. (9) is

$$r = \begin{cases} r_{\text{safety}}, & \text{single-reward baseline,} \\ \lambda_{\text{risk}} r_{\text{risk}} + \lambda_{\text{follow}} r_{\text{follow}} + \lambda_{\text{lane}} r_{\text{lane}}, & \text{multi-reward setting.} \end{cases} \quad (25)$$

Table 7 Compare the open-loop and closed-loop scores with baselines.

Model Name	Data Size	Open-Loop Score	Closed-Loop Score		
			Success Rate	Stability Score	Overall Score
Regression Model	XL	73.98	21.81	0.00	10.91
Base Model	M	51.07	15.67	0.00	7.83
with τ_0 -loss & τ_0 -pred	M	75.27	22.84	0.00	11.42
HDP	XL	88.94	71.24	79.53	75.38
HDP-RL	-	-	83.49	84.65	84.07

E Baseline Comparison Results

E.1 IL Pre-Training Results

On top of the results in Section 6, we select two baselines for comparison: (1) a regression-adaptation of *HDP* that shares the same architecture but is trained with regression loss; (2) an anchor-based planning algorithm with truncated diffusion (Liao et al., 2025). To ensure the maximum capability gain from training, we use the largest dataset (70M frames) for training the two baselines.

Regression Adaptation. The regression model has been used as a baseline method for autonomous driving with compatible benchmark performance (Chitta et al., 2023; Liao et al., 2025). In our settings, the regression adaptation shares the same scene encoder with *HDP*. For the decoder, we replace the embeddings of noised trajectory with a group of learnable queries to extract information from the scene features and to predict the final trajectory. For the model architecture, we retain the DiT-like architecture that uses stacked layers with attention blocks. The model is supervised with L_2 loss to predict the future trajectory in waypoint representation. For inference, the model directly outputs one trajectory for each rollout.

We evaluate the regression adaptation model in both open-loop and closed-loop settings, as shown in Table 7. The model demonstrates compatible performance in terms of open-loop score, benefiting from large-scale training with sufficient data coverage. However, the model does not perform well in real-vehicle closed-loop evaluation, achieving an overall score (10.91) much lower than our *HDP* (75.38). This indicates the inability of simple regression to scale up in real-world closed-loop scenarios, calling for more power generation capability of diffusion models with our proposed improvements.

Anchor-Based Truncated Diffusion. The model shares the scene encoder with *HDP* as well. For the trajectory decoder, we followed DiffusionDrive (Liao et al., 2025) for implementation. Specifically, we first generate N_{anchor} anchors using the training data with k-means algorithm. We select $N_{anchor} = 64$ as a balance between mode coverage and compute efficiency compared with $N_{anchor} = 20$ used in (Liao et al., 2025) for NavSim benchmark, given the larger data volume of ours. The decoder is composed of $N_{layer} = 6$ repeated layers, each of which comprises separate attention blocks for scene features and ego-vehicle features, and two MLP-based heads for trajectory prediction and scoring respectively. For the diffusion part, we use DDIM (Song et al., 2020) with $T = 1000$. The model takes in slightly-noised anchors ($t \leq 50$) as well as scene embeddings as input, and predicts a group of trajectory candidates and classification scores in each layer. The output candidates of preceding layers are used as input for subsequent layers to enable multiple forward passes for refinement. During training, we select one positive sample from each of the N_{layer} candidate groups. This sample is chosen based on the anchor closest to the ground truth. For supervision, L_1 loss is applied to

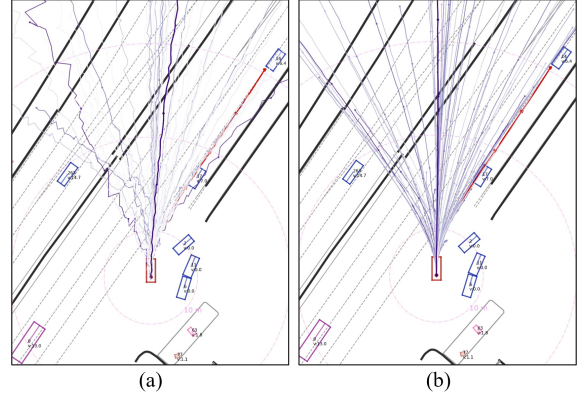


Figure 16 Visualization of anchor-based truncated diffusion using (a) offset prediction and (b) direct prediction. The ground truth is in red and model predictions in purple. The deeper the trajectory is, the higher the predicted score.

the N_{layer} trajectories. Additionally, focal loss is applied to the N_{layer} classification scores, using the one-hot encoded index of the closest anchor as the target label. For inference, the model starts from noisy anchors with $t = 8$, and the final prediction is selected from the last layer output as the trajectory with the highest score.

In practice, we find a severe performance mismatch compared with benchmark results from DiffusionDrive (Liao et al., 2025) as shown in Figure 16. The model’s generations exhibit obvious irregular jitter, leading to low quality trajectories. We credit this to the offset prediction in the original implementation, which predicts the offset of noisy anchors from the ground truth: $\text{offset} = \tau_0 - (\alpha_t \tau_0 + \sigma_t \epsilon) = (1 - \alpha_t) \tau_0 - \sigma_t \epsilon \approx -\sigma_t \epsilon$, $t \ll T$. This resembles the ϵ -prediction as discussed in Section 4.1. Instead, we supervise the direct output of the model with the ground truth. This results in better smoothness in the generated trajectories, as shown in Figure 16. However, we still find that the model’s predictions are strongly influenced by the anchor and fail to pick the correct candidate, and are unfortunately infeasible for real-vehicle deployment.

E.2 RL Post-Training Results

To benchmark our RL-hybrid loss in Eq. (9), we additionally implement a baseline following (Black et al., 2024b; Ren et al., 2025) (DPPO-style): the denoising chain $\tau_T \rightarrow \dots \rightarrow \tau_0$ is treated as a T -step MDP with per-step Gaussian transitions $\pi_\theta(\tau_{t-1} | \tau_t, s) = \mathcal{N}(\mu_\theta(\tau_t, t, s), \Sigma_t)$, and a clipped PPO surrogate is applied at each transition:

$$\mathcal{L} = -\mathbb{E}_{s, k, t} [\min(\rho_t^k A_t^k, \text{clip}(\rho_t^k, 1 \pm \epsilon) A_t^k)], \quad (26)$$

where ρ_t^k is the per-step current/old log-prob ratio and A_t^k is a within-state advantage normalized over K trajectory candidates per state, with rollouts collected under an EMA-updated old policy. In practice this baseline was fragile to train and required a battery of numerical safeguards. The results are shown in Table 8. DPPO-style methods exhibit signs of reward hacking: although the stability score improves substantially, the low success rate indicates frequent takeovers, resulting in an unreliable planner. HDP-RL, in contrast, improves both metrics, and the multi-reward setting further enhances performance.

Table 8 Compare the closed-loop scores with DPPO-style RL methods (Black et al., 2024b; Ren et al., 2025).

Model Name	Closed-Loop Score		
	Success Rate	Stability Score	Overall Score
HDP-RL [†] (only safety reward)	72.89	79.53	76.20
DPPO-style methods (multi-rewards)	65.50	89.76	77.63
HDP-RL (multi-rewards)	83.49	84.65	84.07

Let T denote the number of denoising steps, K the number of candidates per state. The per-state cost of one RL update is:

- **HDP-RL (ours):** the buffer holds K trajectory tuples (s, τ_0, r) ; each is processed by one forward-backward at a single random t . Rollout cost: $K T$ no-grad forwards. Optimization cost: K forward-backwards.
- **DPPO-style methods:** the buffer holds $K T$ transition tuples (the entire denoising chain for each candidate); each transition contributes one forward-backward to evaluate $\mathcal{L}_{DPPO}$. Rollout cost: $K T$ no-grad forwards. Optimization cost: $K T$ forward-backwards—a $T \times$ overhead over RL-hybrid.

F Limitations & Discussions & Future Work

Here, we discuss the limitations, potential solutions, and interesting directions for future research.

- **Lack of Trajectory Selection Mechanism.** Our framework directly adopts the trajectory generated by the diffusion model as the final planning output, without an additional selection or scoring stage. The strong closed-loop performance on real vehicles already demonstrates that the generated trajectories are of high quality. Nevertheless, from a practical deployment perspective, an explicit trajectory selection mechanism is often beneficial for filtering out occasional low-quality samples and further improving robustness.

Solution and future work: A common solution is to introduce an auxiliary scoring head, e.g., a learned scorer or a rule-based filter that ranks multiple candidate trajectories and selects the best one (Li et al., 2024a). However, the central goal of this work is to thoroughly investigate the inherent capability of diffusion models for E2E AD, and we therefore deliberately avoid introducing additional engineering components that may obscure the contribution of the diffusion model itself. Combining HDP with a learned trajectory scorer is a natural and promising direction for production-oriented systems, and is fully complementary to our findings.

- **Pseudo Closed-Loop Reinforcement Learning.** Our RL post-training is conducted in a non-reactive pseudo closed-loop simulator built upon real-world logged data, where neighboring vehicles replay their recorded behaviors and do not react to the ego vehicle. This inevitably introduces a gap with truly interactive closed-loop training, and may underestimate certain reactive behaviors of the surrounding agents.

Solution and future work: A natural alternative is to perform reactive closed-loop RL within a learned driving world model (Hu et al., 2023a; Agarwal et al., 2025), so that surrounding agents can respond to the ego vehicle. However, current world models are still extremely expensive to train and run at scale, and their generative quality on truly out-of-distribution scenarios remains limited, making them more suitable as offline evaluators rather than as scalable RL environments at this stage. Despite this limitation, we observe that even with a simple non-reactive pseudo closed-loop simulator, our reward-weighted diffusion RL already yields substantial improvements on real-vehicle tests, validating the effectiveness of the proposed RL post-training framework. Combining HDP-RL with reactive world models is an exciting future direction that could further amplify these gains.

- **Compatibility with VLM/VLA Backbones.** In this work, we instantiate the diffusion planner on top of an in-house perception backbone, and have not investigated how vision–language pretraining or large-scale vision–language–action (VLA) models (Wang et al., 2025; Liang et al., 2026) can be combined with our framework. As a result, the world knowledge and reasoning capabilities accumulated by recent VLM/VLA foundations are not yet fully exploited in HDP.

Solution and future work: An important observation is that the contributions of this paper—diffusion loss space, hybrid trajectory representation, data scaling, and the RL-hybrid post-training—are largely orthogonal to the choice of upstream encoder. HDP can therefore be naturally used as a *plug-in diffusion action head* on top of a VLM/VLA backbone, where the planner inherits the rich semantic and commonsense priors from large-scale vision–language pretraining, while still benefiting from our findings on stable, scalable diffusion-based action generation. We view this combination as a particularly promising path towards more generalizable and reasoning-aware E2E driving policies, and leave a thorough investigation as future work.

- **Limited Simulation-Based Evaluation.** Our evaluation primarily relies on offline open-loop replay metrics and large-scale closed-loop real-vehicle road tests. This makes our results harder to compare side-by-side with prior diffusion-based planners that are mainly evaluated in simulation.

Solution and future work: We argue that this is in fact a deliberate choice rather than a pure shortcoming. Existing simulation benchmarks are known to suffer from non-trivial biases, including limited scenario diversity, non-reactive log-replay neighbors, and a notable gap between simulator metrics and on-road performance (Dauner et al., 2023b). Instead, we validate HDP on a real vehicle across 6 urban scenarios and 200 km of road testing, and observe that our key design choices consistently translate into substantial closed-loop improvements, leading to a planner that performs reliably in real-world driving. Nevertheless, we acknowledge that real-vehicle testing is costly and not easily reproducible by the broader community, and we believe that building more realistic, reactive, and diverse closed-loop driving benchmarks is an important and pressing direction for future research.

Overall, although certain limitations exist, our work provides the systematic and large-scale investigation of diffusion-based planning for real-world E2E AD. Through carefully controlled studies, we identify a set of clean and principled design choices—diffusion loss space, hybrid trajectory representation, data scaling, and the RL-hybrid post-training—that together unleash the potential of diffusion models as a scalable and deployable E2E AD planner. The 10x closed-loop improvement achieved on real vehicles, with only a light smoothness post-processing and without heavy rule-based engineering, demonstrates that diffusion models can serve as a strong foundation for the next generation of end-to-end autonomous driving systems.