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

Author Names Omitted for Anonymous Review

***Abstract*—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 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.**

I. Introduction

Diffusion models [17, 46] have demonstrated remarkable capabilities in image and video generation tasks [3, 12, 34, 43] and are becoming increasingly popular in robotics control [4, 10, 21, 33]. More recently, diffusion models have also shown promise to solve decision-making tasks in autonomous driving (AD) [55, 32, 48, 26]. However, most existing research on the application of diffusion models to AD tasks remains restricted to performance validation in open-loop [7] and simulation-based settings [8, 14], leaving their effectiveness in complex, closed-loop, real-world deployments largely unverified.

Among all task settings, End-to-End Autonomous Driving (E2E AD) [6, 9, 20] represents an important and practically viable direction [49], which leverages powerful deep neural networks and large amounts of real-world data to directly learn multimodal human driving behaviors in complex traffic scenarios. Unfortunately, applying diffusion models to E2E AD and successfully deploying them on real vehicles is far more challenging than conducting experiments in simulation. In this context, the model must be high-capacity to handle diverse scenarios, while remaining compact and efficient to meet the latency requirements of in-vehicle hardware. Additionally, closed-loop real-vehicle testing amplifies issues such as error accumulation and requires a higher level of safety, which is notoriously difficult for learning-based methods to guarantee [8, 13, 54]. Consequently, existing approaches often rely on rule-based post-processing [15] or incorporate strong assistive designs, such as pre-defined anchor trajectories [29] or explicit goal conditions [2, 16], to reduce the learning burden. However, the inherent power of diffusion models is obscured by excessive additional engineering designs, and their potential capability and scalability remain unproven. This raises a critical question: *Are we fully exploiting the potential of diffusion models as AD planners?*

To answer this question, we conducted a systematic and large-scale investigation in diffusion-based E2E AD using a huge amount of real-vehicle data and rigorous road testing. In this journey, we begin by using an industrial-grade perception backbone as the encoder to process image and LiDAR inputs for the end-to-end model, and a vanilla diffusion-based planning head, inspired by Zheng et al. [55], as the decoder to generate the planning trajectory. Building on this base model, we conduct comprehensive ablation studies and summarize our key findings as follows:

- **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 [27] 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**. We observe that directly generating waypoints yields superior spatial awareness, whereas velocity prediction results in smoother trajectories. Therefore, our model outputs velocity but is supervised on both velocity and waypoints. Crucially, we mathematically prove that this hybrid loss formulation does not alter the optimal solution of diffusion training, while allowing us to leverage advantages from both sides.
- **Emergence of data scaling**. Keeping a minimalist and clean design allows our diffusion framework to effectively benefit from data scaling in real-world testing. We find that our model captures richer multimodal driving behaviors and better closed-loop performance when scaling up driving data. Such scaling properties are not observed when training diffusion models on commonly used AD benchmarks [8, 14], due to overly small training datasets.

While the imitation learning pretraining establishes a strong diffusion planner prior, it lacks explicit optimization for safety-critical scenarios. To bridge this gap, we further fine-tune our model using Reinforcement Learning (RL). By re-weighting

the diffusion learning process with safety-aware advantage estimates [40, 54], we effectively align the generated trajectories with safety constraints while preserving the training stability of the diffusion model. Moreover, we prove that this reweighting method is naturally compatible with our previously introduced hybrid loss, yielding a simple implementation.

Finally, we incorporate all the aforementioned innovations into a complete framework, ***Hyper Diffusion Planner*** (HDP), and successfully deploy it on a real-vehicle platform with only simple smoothness post-refinement. We systematically evaluate *HDP* across 8 urban driving scenarios, covering 200 km of road testing with comprehensive evaluation metrics. Empirically, *HDP* demonstrates a significant performance boost, showing a 10x improvement compared to the baseline model. We also provide a detailed analysis to examine the characteristics of the proposed framework and the impacts of key design elements. The results demonstrate that diffusion models, when properly designed and trained, can serve as effective and scalable planners for complex, real-world autonomous driving tasks.

II. Preliminaries

Our work primarily focuses on the planning module of E2E AD systems, where the planner receives the latent representation C from the perception backbone and generates a trajectory τ_0 for downstream control systems. Diffusion models [46] define a forward process that transforms the conditioned trajectory data distribution $q_0(\tau_0|C)$ into a noised distribution $q_{t0}(\tau_t|\tau_0)$. This process is described by the following equation:

$$q_{t0}(\tau_t|\tau_0) = \mathcal{N}(\tau_t \mid \alpha_t\tau_0, \sigma_t^2\mathbf{I}), t \in [0,1], \tag{1}$$

where α_t, σ_t define a pre-defined noise schedule. As $t \to 1$, this schedule ensures that the marginal distribution $q(\tau_1)$ approaches a normal distribution $\mathcal{N}(\tau_1 \mid 0, \mathbf{I})$. The reversed denoising process of Eq. (1) can be equivalently expressed as a diffusion ODE [47]:

$$\mathrm{d}\tau_t = \left[f(t)\tau_t - \frac{1}{2}g^2(t)\nabla_{\tau_t}\log q_t(\tau_t)\right]\mathrm{d}t, \tag{2}$$

where $f(t) = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}, g^2(t) = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t^2$. A commonly used approach for learning diffusion models [17, 43] is to train a neural network $\epsilon_\theta(\tau_t, t, C)$to fit the Gaussian noise ϵ:

$$\mathcal{L} = \mathbb{E}_{t,\tau_0,\tau_t,\epsilon}||\epsilon_\theta(\tau_t, t, C) - \epsilon||_2^2, \tag{3}$$

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 $\epsilon \sim \mathcal{N}(\tau_1 \mid 0, \mathbf{I})$. Then, we can estimate the score function $\nabla_{\tau_t}\log q_t(\tau_t)$ in Eq. (2) using $s_\theta(\tau_t, t, C) = -\epsilon_\theta(\tau_t, t, C)/\sigma_t$, and use an ODE solver to generate the clean data.

III. 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.

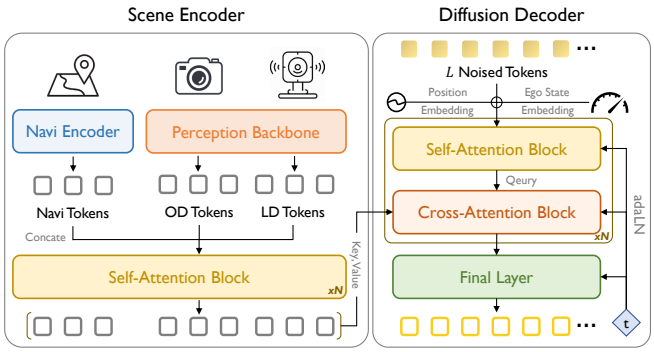


Fig. 1: Model architecture.

A. Base Model

Scene Encoder. We consider an E2E AD system that can directly process multi-modal inputs, including camera images and LiDAR point clouds, to generate planning trajectories [9, 20]. Given our focus on the capabilities of diffusion-based planning, we leverage an in-house validated perception backbone as the encoder for the planner. Briefly, we first compress the heterogeneous sensory data into a unified Bird's Eye View (BEV) feature representation [30]. We then employ two distinct sets of transformer-based queries for different perception tasks: (1) Object Detection (OD) tokens for vehicle and pedestrian localization, and (2) Lane Detection (LD) tokens for road structure understanding. The encoder is pretrained to provide solid representation initialization for the planning task. Subsequently, we concatenate the OD tokens, LD tokens, and Navi tokens (which encode navigation information) and process them using several self-attention blocks for further fusion in the downstream planning module.

Diffusion Decoder. We return to the vanilla version of the transformer-based diffusion model [39] and use it as our diffusion decoder. The planner receives the latent representation C, which includes both OD, LD, and Navi tokens, along with the current velocity, to generate the trajectory $\tau_0 \in \mathbb{R}^{L\times 4}$. This trajectory τ_0 consists of L timesteps, where each timestep contains the ego-centric waypoint coordinates and the cosine and sine values of the heading. The overview of the model architecture, as shown in Fig. 1, begins with splitting the noised trajectory τ_t and projecting it into L tokens, with position embedding and velocity embedding added. The self-attention block is used to fuse information across all noised tokens. Subsequently, a cross-attention block is employed to integrate the trajectory tokens with the concatenated OD, LD, and Navi tokens. Meanwhile, the diffusion timestep t is incorporated into the model through an adaptive layer normalization block [39]. After several blocks, an MLP-based final layer [33, 55] generates the predicted noise, and Eq. (3) is used for model training.

B. Evaluation Metrics

To better examine the effectiveness of our designs, we need reasonable metrics for model evaluation. We consider two types of metrics: open-loop metrics and closed-loop

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

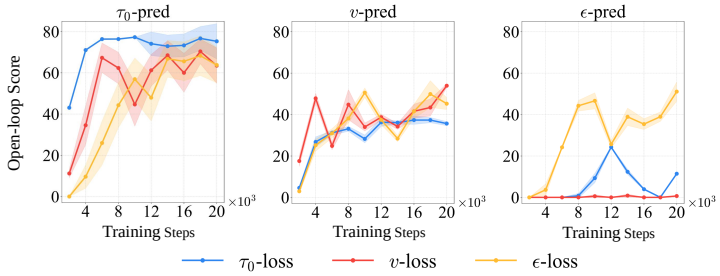


Fig. 2: The learning curve of models trained with different loss designs. The results are averaged over three evaluations.

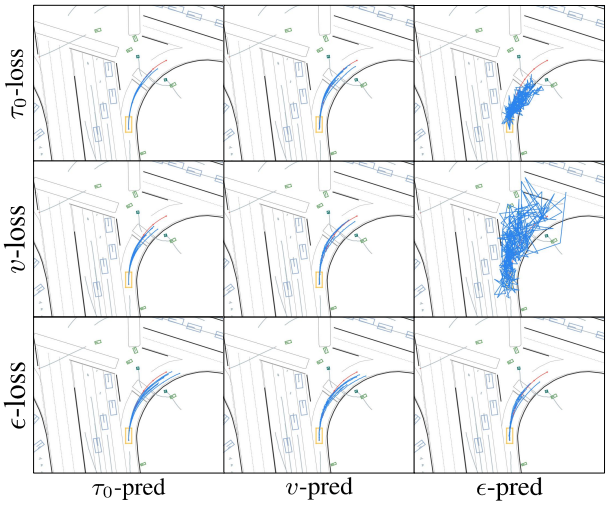


Fig. 3: The open-loop visualization of planning trajectories. 6 generations are plotted for each scene. Ego vehicle in yellow, model predictions in blue and ground truth trajectory in red.

metrics. The former are used to evaluate the quality of the trajectory and multi-modality through data replay, while the latter assess performance in closed-loop real vehicle-testing (see Appendix C for more details).

For open-loop evaluation, following the design used in nuPlan [8], we mainly consider the following metrics: *Average Displacement Error* (ADE), *Final Displacement Error* (FDE), *Comfort*, and *Collision Rate* (CR). To aggregate the metrics for a more comprehensive assessment, we obtain a score S_m for each metric m and compute the final open-loop score as their weighted sum: $(1 - CR) \times \sum_{m\in\mathcal{M}} \omega_m S_m$, where $\mathcal{M} = \{ADE, FDE, Comfort\}$ and ω_m are the corresponding weights. To measure the divergence of the generated trajectories and facilitate the multimodality analysis, we further introduce a *Trajectory Divergence* metric, which is computed as the average pair-wise Euclidean distance of all model generations.

For closed-loop real vehicle testing, we use a fixed route to conduct controlled experiments. We log the *success rate* of six commonly occurring scenarios, including: starting maneuvers, car-following with stopping, navigational lane changes, yielding to VRUs, yielding to cross traffic at intersections, and left and right turns. The success rate is calculated as a weighted mean of all six scenarios, with higher weights assigned to more frequent scenarios for a more accurate evaluation. Additionally, we also compute a *stability score* based on the average of centering performance and speed compliance. The overall *closed-loop* score is then determined as the average of the *success rate* and the *stability score*.

C. 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. Hence, very different design considerations could apply. To address these challenges, we structure our exploration into two separate phases: 1) **Imitation Learning Pre-Training**, where we study how diffusion loss and trajectory representation influence planning trajectory quality, and validate data scaling in a closed-loop setting; and 2) **Reinforcement Learning Post-Training**, where we use RL to further enhance the safety of the pre-trained model by developing a compatible RL algorithm for stable and efficient post-training.

IV. Imitation Learning Pre-Training

A. Diffusion Loss Space

As the score function in the denoising process (Eq. (2)) is generally intractable, in practice, the diffusion model is typically trained to predict one of the three conditioned quantities: the noise ϵ [17], the flow velocity v_t [18], or the clean data τ_0 [41]. These quantities are mutually convertible, allowing for various loss space designs (see Appendix B 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 [27] 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 I and Fig. 2, 3.

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.

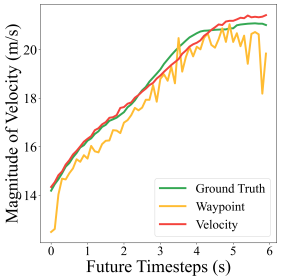


Fig. 4: The v-t curve of generated trajectories using different representations. Waypoint representation suffers severe jitter.

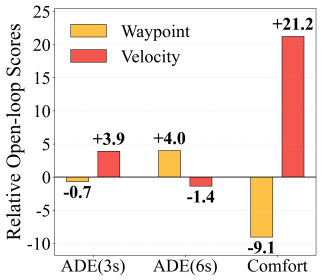


Fig. 5: The relative open-loop score of waypoint and velocity representation. The scores are computed in the same way as stated in Section III-B

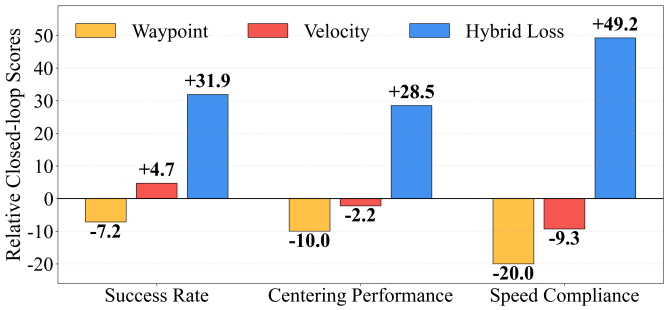


Fig. 6: The relative real-vehicle closed-loop performance of different trajectory representations. The scores are computed in the same way as stated in Section III-B.

To study the reasons for this advantage, we provide further investigation from the following perspectives.

Fast convergence: Fig. 2 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 [27]. 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 Fig. 3. Although most models generate trajectories that resemble the ground truth, the quality 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 [38]. 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.

B. 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 velocity[1] curve, as shown in Fig. 4. 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.

To fix this issue, a possible solution is to use a delta representation of the trajectory to achieve higher-order supervision, i.e., enforce the model to predict the velocity $\tau_0^{\mathbf{v}} = \{(v_x^l, v_y^l)\}_{l=1}^L$ instead of absolute waypoints $\tau_0^{\mathbf{x}} = \{(x^l, y^l)\}_{l=1}^L$ of a trajectory. In the velocity representation, the final trajectory is obtained via integration during inference. Interestingly, we find empirically that these two trajectory representations can have a great impact on the trajectories generated by diffusion planners. As shown in the v-t curves in Fig. 4 and the analysis on the decomposed metrics in Fig. 5, trajectories obtained via velocity representation demonstrate smoothness and stability similar to that of the human driving trajectory, enjoying a much higher comfort score. By contrast, waypoints-represented trajectories suffer from severe jerky movement, but at the same time have a superior ADE score, due to better modeling of global geometric structure. This calls for a balanced consideration to leverage the strengths of both representations while mitigating their limitations.

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 L2 loss on both the directly output velocity and

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

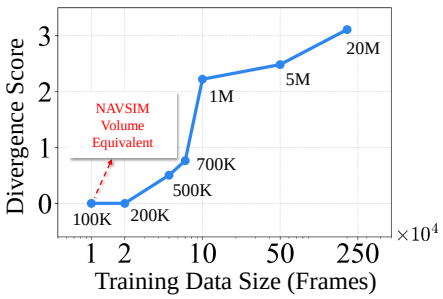


Fig. 7: The divergence score of HDP trained on different sizes of data.

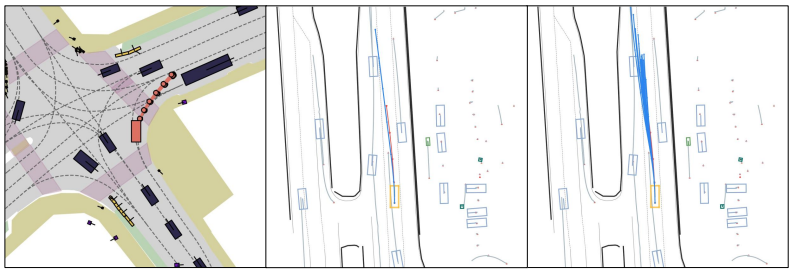


Fig. 8: Planning trajectories generated using HDP trained with different data origins and sizes.

the integrated waypoints:

$$\begin{aligned}\mathcal{L}_{velocity} &= \mathbb{E}_{\tau_0^{\mathbf{v}},\epsilon,t}\|\tau_\theta^{\mathbf{v}} - \tau_0^{\mathbf{v}}\|_2^2 \\ \mathcal{L}_{waypoints} &= \mathbb{E}_{\tau_0^{\mathbf{x}},\epsilon,t}\|M\tau_\theta^{\mathbf{v}}\cdot\Delta t - \tau_0^{\mathbf{x}}\|_2^2 \\ &= \mathbb{E}_{\tau_0^{\mathbf{v}},\epsilon,t}\|M\tau_\theta^{\mathbf{v}}\cdot\Delta t - M\tau_0^{\mathbf{v}}\cdot\Delta t\|_2^2,\end{aligned} \tag{4}$$

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:

$$\mathcal{L}_{hybrid} = \mathcal{L}_{velocity} + \omega\cdot\mathcal{L}_{waypoints}, \tag{5}$$

where ω is a balancing weight. Moreover, we can theoretically show that this hybrid loss is also a valid diffusion loss to obtain the correct marginal score function of the data distribution:

Theorem IV.1. *The hybrid loss in Eq. (5) is equivalent to a diffusion score matching loss under P-norm:*

$$\mathcal{L}_{hybrid} = \mathbb{E}_{\tau_0^{\mathbf{v}},\epsilon,t}[||\tau_\theta^{\mathbf{v}} - \tau_0^{\mathbf{v}}||_P^2], \tag{6}$$

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

It is worth noting that many existing studies adopt L1-norm loss [32] or auxiliary planning loss [22] (e.g., collision loss) in diffusion-based AD tasks, which will lead to a biased score function that does not faithfully reflect the data distribution. Please see Appendix B for proof of Theorem IV.1.

In practice, the integration in $\mathcal{L}_{waypoint}$ could result in gradient accumulation with future timesteps. To avoid the imbalanced gradient distribution over future predictions, we limit the gradient backpropagation to a temporal window of size W by detaching the trajectory history beyond this horizon (see Algorithm 1 in Appendix C for detailed implementation).

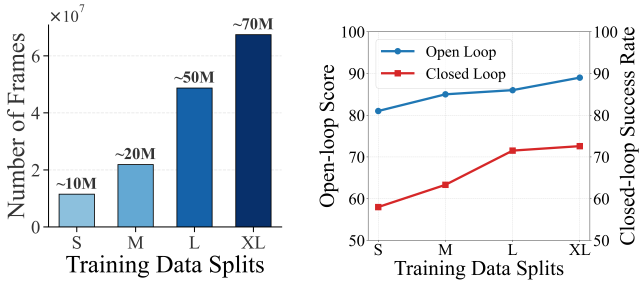
Our proposed hybrid loss enables substantial performance improvement in closed-loop real-vehicle testing. As shown in Fig. 6, training the diffusion model with the hybrid loss improves all closed-loop metrics, outperforming solely using waypoint and velocity representation by a large margin. This shows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

C. Multimodal Capability and Data Scaling

Diffusion models are renowned for their multimodal generation capabilities. However, existing diffusion-based planning models [55, 48] often suffer from severe mode collapse on AD benchmarks [8, 14]. To investigate the reasons for this discrepancy, as well as examine the multimodal capability and scalability of our proposed diffusion-based framework HDP, 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 [14] only contain 100K training data. We evaluate our model's multimodal generation capability in Fig. 7, 8, as well as its open- and closed-loop scaling performance in Fig. 9.

Multimodal generation capability. We train our model with data from 100K (NavSim equivalent) to 20M frames, and use the *trajectory divergence* metric introduced in Section III-B to measure multimodal generation. The results are shown in Fig. 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 Fig. 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. [53], 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 [32] or goal conditioning [51].

Performance scaling. We also observe a continuous improvement in both open-loop and closed-loop performance of our model as the training data increases, as shown in Fig. 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 clear data scaling property on real vehicles. This demonstrates the huge potential of our proposed HDP for



(a) Number of frames of training data splits.

(b) Performance improvement as training data scaling up.

Fig. 9: Data scaling experiments. Both open- and closed-loop performance gain great improvement as training data scale up.

large-scale industrial-level applications.

V. Reinforcement Learning Post-training

Safety remains a critical challenge for imitation learning models. Since the training phase generally does not enforce explicit safety constraints, the resulting models could face serious safety risks during deployment, especially in closed-loop environments. In this section, we introduce an RL method to further enhance the model's safety performance.

A. Diffusion-Based Reinforcement Learning

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 τ_0, and the state s denotes the latent representation C. We consider an RL 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}:

$$\max_{\pi^k} \ \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{a\sim\pi^k}\left[r(s,a)\right] - \frac{1}{\beta} D_{\text{KL}}\left(\pi^k \| \pi^{k-1}\right)\right], \quad (7)$$

where $\mathcal{D}$ is the replay buffer, $\beta > 0$ is the temperature parameter, and $D_{\text{KL}}\left(p\|q\right) = \mathbb{E}_{x\sim p}\left[\log\left(p(x)/q(x)\right)\right]$. The KL-regularized objective in Eq. (7) provides a closed-form solution for π^k as follows [37]:

$$\pi^{k^\star}(a \mid s) \propto \pi^{k-1}(a \mid s) \cdot \exp(\beta r(s,a)). \quad (8)$$

To extract the optimal policy in Eq. (8), one approach is to use classifier guidance to steer the diffusion process toward generating high-reward actions during inference [36, 55]. 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 [5, 31, 54]:

$$\mathcal{L}_{RL} = \mathbb{E}_{t,\epsilon,(s,a)\sim\mathcal{D}}\left[\exp\left(\beta r(s,a)\right) ||\epsilon_\theta^k(a_t,t,s) - \epsilon||_2^2\right] \quad (9)$$

where $a_t = \alpha_t a + \sigma_t \epsilon$, and ϵ_θ^k denotes the parameterized diffusion model corresponding to the policy π^k, as introduced in Section II. The weighted regression loss in Eq. (9) 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 [5, 42], and use RL algorithms like PPO [44] 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.

To maintain consistency with the hybrid loss defined in Eq. (5) used during imitation pretraining, we introduce the RL-hybrid loss for the post-training phase.

$$\mathcal{L}_{RL-hybrid} = \mathbb{E}_{\mathbf{v},\epsilon,t}[\exp(\beta r)\, ||\mathbf{v}_\theta^k - \mathbf{v}||_P^2]. \quad (10)$$

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 V.1. Proof see Appendix B.

Theorem V.1. *Optimal action $a \sim \pi^{k^\star}(a|s)$ in Eq. (8) can be generated by optimizing the weighted diffusion loss in Eq. (10) and solving the diffusion reverse process with the learned $\mathbf{v}^{k^\star}$.*

B. Practical Implementation

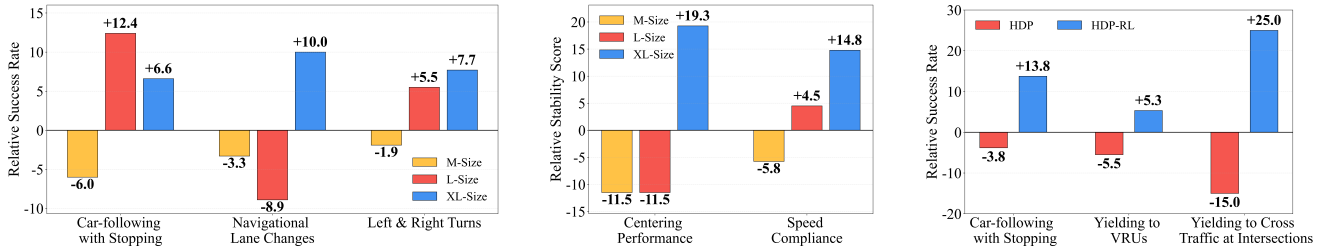
In practice, we initialize the policy π^0 for RL post-training using an imitation model pretrained with the hybrid loss in Eq. (5). Given the safety risks of conducting online RL on real vehicles [24] and the high computational cost of world models [1, 19], we adopt a non-reactive pseudo-closed-loop simulation [14] based on real-world datasets. In this setup, neighboring vehicles replay logged behaviors, while our model generates planning trajectories. We employ the Separating Axis Theorem (SAT) to detect overlaps between the oriented bounding boxes of the ego vehicle and neighboring vehicles. Accordingly, the safety reward is defined as: $r_{\text{safety}} = 1 - \max_{l=1,\cdots,L} c_l$, where c_l imposes a full penalty (1.0) for active collisions, while employing an attenuated penalty (0.3) for rear-end accidents to mitigate the artifacts arising from the non-reactive nature of the simulator. To achieve stable training using Eq. (10), we apply reward group normalization [45] 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. See Appendix C for more details.

VI. Real-Vehicle Testing Results

Model Training. 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 III-A, which uses ϵ-loss and ϵ-pred (Base Model). In Section IV-A, 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 IV-B, we discover that using velocity as a supervision signal performs better than using waypoints (+ Velocity Supervision). Combining both improvements, we introduce a

TABLE II: Main results. represents the highest score in various metrics. 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 (HDP-RL)	-	-	72.89	79.53	76.20



(a) Success rate of frequently occurring scenarios. (b) Details of the stability score. (c) Success rate in safety-related scenarios.

Fig. 10: (a–b) Details the performance of success rate and stability score under different volumes. (c) Compares the success rate before and after RL post-training.

hybrid loss function (+ Hybrid Loss). Furthermore, we scale up the dataset in Section IV-C from the original 20M samples to 50M (+ Data Scaling / L) and 70M (+ Data Scaling / XL), resulting in the final version of HDP. Finally, we apply RL methods in Section V to further enhance safety performance, leading to the model HDP-RL.

Model Inference. 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. [55], which employs the DPM-Solver [35] 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.

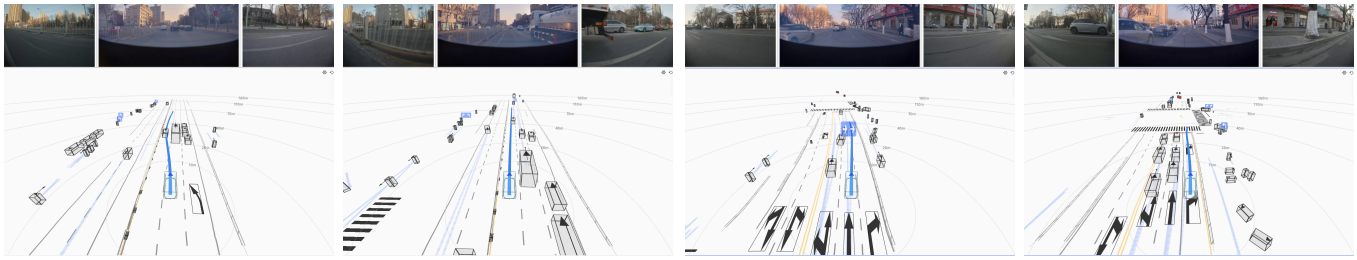
A. Main Results

We present the main results in TABLE II. 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 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 Fig. 10a. There is a noticeable performance drop of 6.2 on the XL-sized datasets compared to the L-sized datasets. This may indicate a trade-off: as the model focuses more on learning the complex "Navigational lane change" behavior (which improves significantly by +18.9), its performance on the simpler "Car-following with stopping" task degrades. As shown in Fig. 10b, 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.

Furthermore, as shown in Fig. 10c, RL significantly improves safety-related performance. However, the overall performance does not improve substantially, likely because our current reward function only considers safety. This may lead the policy to behave conservatively, resulting in lower scores in scenarios that require driving efficiency. We leave the integration of additional reward components for future work.

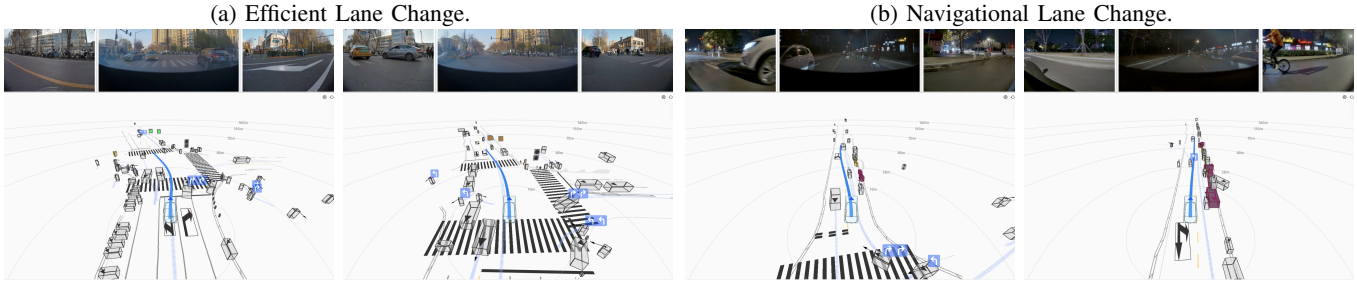
B. Case Study

As shown in Fig. 13, HDP demonstrates a strong capability to handle complex urban driving scenarios. For example, in Fig. 11a, the vehicle changes lanes to avoid a slow-moving truck ahead, showcasing its flexibility. In addition, the vehicle can perform lane changes based on navigation instructions while complying with traffic rules, as illustrated in Fig. 11b. Moreover, it can safely avoid both vehicles and VRUs during driving. We also compare the behavioral differences between



(a) Efficient Lane Change.

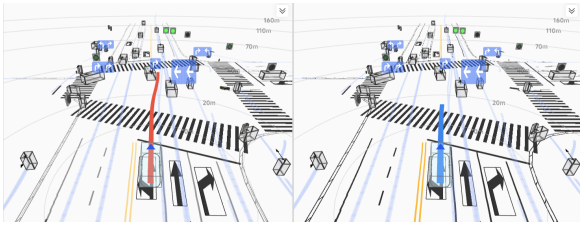
(b) Navigational Lane Change.



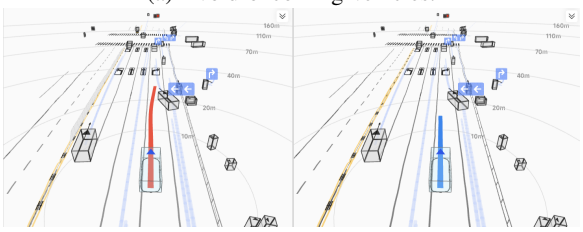
(c) Vehicle avoidance at intersection.

(d) VRU avoidance.

Fig. 11: Closed-loop real-vehicle testing results. Two representative frames from the scenario are captured for illustration.



(a) Avoid oncoming vehicles.



(b) Avoid cutting in vehicles.

Fig. 12: Visualization of open-loop data replay for bad cases in real-vehicle testing before and after RL post-training. HDP-RL in blue and HDP in red.

the HDP and the HDP-RL models. As shown in Fig. 12, the RL post-trained model effectively avoids surrounding vehicles, demonstrating improved safety and proving the effectiveness of the RL algorithm. More cases are shown in Appendix A.

VII. Related Works

Diffusion models [17, 46] have recently gained widespread popularity in decision-making tasks due to their strong capability to model complex data distributions. In autonomous driving, Zheng et al. [55] make a pioneering attempt by applying diffusion models to planning tasks, although they still rely on vectorized scene representations. Liao et al. [32] introduce a truncated denoising process, but this modification disrupts the original diffusion mechanism, and their model still heavily depends on trajectory anchors for trajectory generation. Wang et al. [50] develop an autonomous driving VLA model using a diffusion model to generate trajectories with strong reasoning capabilities. Building upon imitation learning pretrained models, researchers have applied RL methods to diffusion models in order to further improve performance. One straightforward approach is to optimize a reward or value function [52, 11], but backpropagating gradients through the denoising process is often noisy and unstable. Another approach [42] treats each denoising step as a Gaussian transition and applies PPO [44] to diffusion models, though this leads to high computational costs. Li et al. [28] extend this idea from Black et al. [5] in the context of autonomous driving. Moreover, weighted regression [25, 23, 54] offers a simpler alternative for diffusion-based RL. Liang et al. [31] propose a dichotomous policy optimization method and fine-tune a 1 billion-parameter diffusion-based VLA model for autonomous driving, achieving stable training performance.

VIII. 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.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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, 2024.

[5] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, 2024.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning. In *Conference on Robot Learning (CoRL)*, 2023.

[14] 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.

[15] 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.

[16] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15303–15312, 2021.

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[18] 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.

[19] 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*, 2023.

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

[21] 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.

[22] 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 *Pro-*

ceedings of the IEEE/CVF International Conference on Computer Vision, pages 8340–8350, 2023.

[23] 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.

[24] 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.

[25] 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.

[26] Pengxiang Li, Yinan Zheng, Yue Wang, Huimin Wang, Hang Zhao, Jingjing Liu, Xianyuan Zhan, Kun Zhan, and Xianpeng Lang. Discrete diffusion for reflective vision-language-action models in autonomous driving. *arXiv preprint arXiv:2509.20109*, 2025.

[27] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

[28] 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*, 2025.

[29] 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*, 2024.

[30] 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*, 2024.

[31] Ruiming Liang, Yinan Zheng, Kexin Zheng, Tianyi Tan, Jianxiong Li, Liyuan Mao, Zhihao Wang, Guang Chen, Hangjun Ye, Jingjing Liu, Jinqiao Wang, and Xianyuan Zhan. Dichotomous diffusion policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026.

[32] 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.

[33] 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*, 2025.

[34] 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.

[35] 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.

[36] 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.

[37] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[45] 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.

[46] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[47] 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.

[48] 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.

[49] Tesla. Tesla ai day 2022. https://www.youtube.com/watch?v=ODSJsviD_SU, 2022.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

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

APPENDIX

A. Visualization of Real-Vehicle Testing Results

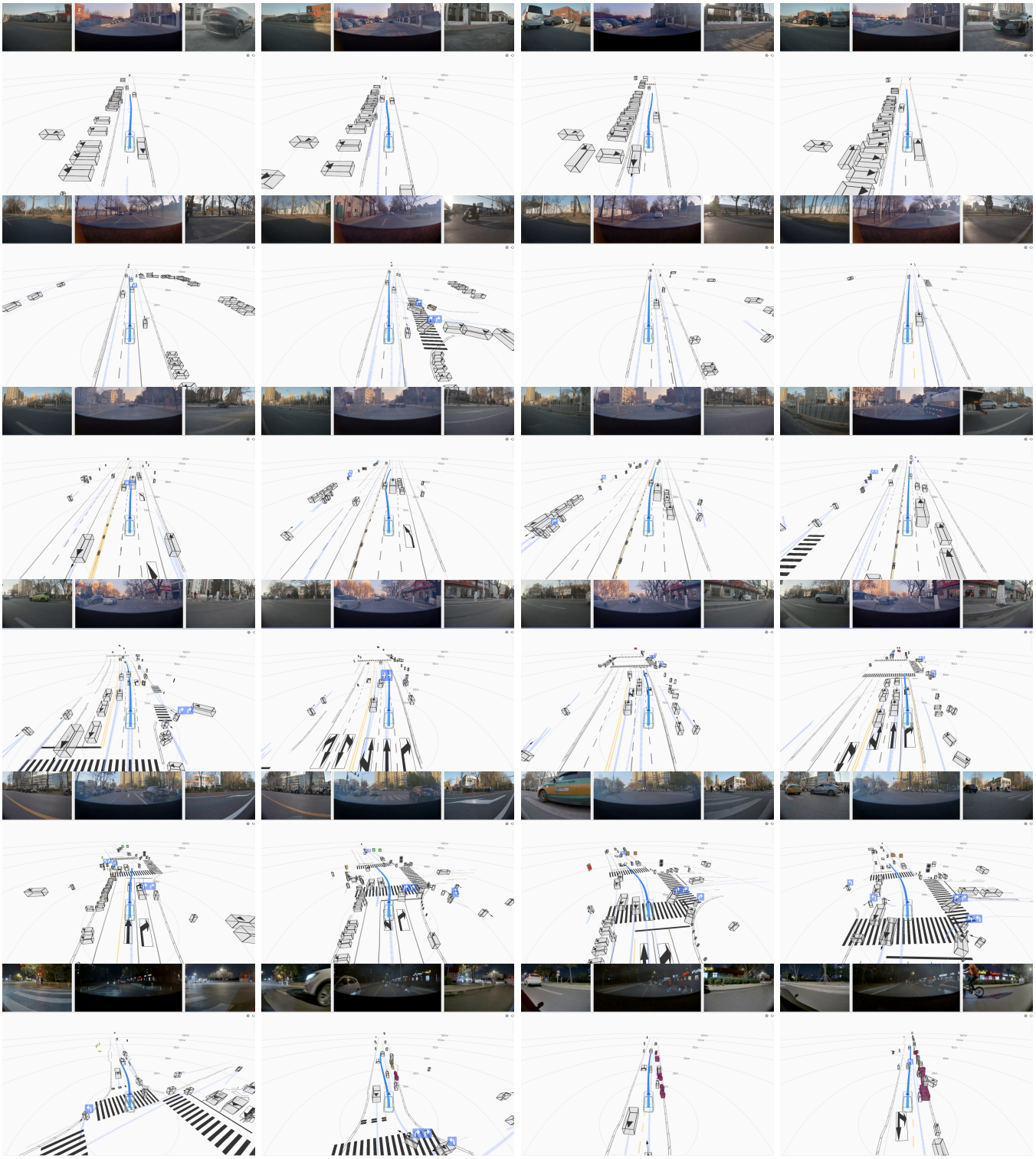


Fig. 13: Closed-loop real-vehicle testing results. Each row contains representative frames from the scenario.

TABLE III: The mutual conversions of diffusion quantities. The predicted quantities are distinguished with $\hat{}$ and the model is parameterized with θ.

	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:** $\mathbb{E}\|\|\hat{\epsilon} - \epsilon\|\|^2$	$\hat{\epsilon} = (\tau_t - \alpha_t\tau_\theta)/\sigma_t$	$\hat{\epsilon} = \sigma_t\tau_t + \alpha_t v_{\theta;t}$	$\hat{\epsilon} = \epsilon_\theta$

B. 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.

1) Diffusion Loss Space: The diffusion models are trained to predict one of the following quantities: τ_0, v_t or ϵ. Given the diffusion timestep t, the predefined noise schedule α_t, σ_t and noised sample τ_t, these quantities are mutually convertible. This provides the freedom of combinations of model predictions (parameterization) and loss functions, as in TABLE III. For instance, we can parameterize the model to output the clear trajectory τ_θ and transform the prediction into noise space to compute the loss:

$$\mathcal{L} = \mathbb{E}_{\tau_0,t,\epsilon}\|\frac{\tau_t - \alpha_t\tau_\theta}{\sigma_t} - \epsilon\|^2 \tag{11}$$

2) Proofs of The Theorems: We provide the proofs of the theorems for the hybrid loss and the RL objectives.

Theorem IV.1. *The hybrid loss in Eq. (5) is equivalent to a diffusion score matching loss under P-norm:*

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

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

Proof: Given Eq. 4 and Eq. 5, the hybrid loss is

$$\begin{aligned}
\mathcal{L}_{hybrid} &= \mathbb{E}_{\tau_0^{\mathbf{v}},\epsilon,t}\left[(\tau_\theta^{\mathbf{v}} - \tau_0^{\mathbf{v}})^T(\tau_\theta^{\mathbf{v}} - \tau_0^{\mathbf{v}})\right] \\
&\quad + \omega\mathbb{E}_{\tau_0^{\mathbf{v}},\epsilon,t}\left[\Delta t^2(\tau_\theta^{\mathbf{v}} - \tau_0^{\mathbf{v}})^T M^T M(\tau_\theta^{\mathbf{v}} - \tau_0^{\mathbf{v}})\right] \\
&= \mathbb{E}_{\tau_0^{\mathbf{v}},\epsilon,t}\left[(\tau_\theta^{\mathbf{v}} - \tau_0^{\mathbf{v}})^T(I + \omega\Delta t^2 M^T M)(\tau_\theta^{\mathbf{v}} - \tau_0^{\mathbf{v}})\right] \\
&= \mathbb{E}_{\tau_0^{\mathbf{v}},\epsilon,t}\left[(\tau_\theta^{\mathbf{v}} - \tau_0^{\mathbf{v}})^T P\tau_\theta^{\mathbf{v}} - \tau_0^{\mathbf{v}})\right] \\
&= \mathbb{E}_{\tau_0^{\mathbf{v}},\epsilon,t}\|\tau_\theta^{\mathbf{v}} - \tau_0^{\mathbf{v}}\|_P^2] \\
&= \mathbb{E}_{\tau_0^{\mathbf{v}},\epsilon,t} D_P(\tau_\theta^{\mathbf{v}}, \tau_0^{\mathbf{v}})
\end{aligned} \tag{12}$$

where $P = I + \omega\Delta t^2 M^T M$ is strictly positive definite and $D_P(u,v) = \|u - v\|_P^2$. This indicates that the hybrid loss is a divergence under metric P. To show that the hybrid loss adheres to the original score matching objective, we only need to prove that the divergence D_P is a Bregman Divergence, which provides unbiased gradients to learn the marginal score function from the conditioned score:

$$\begin{aligned}
D_P(u,v) &= (u-v)^T P(u-v) \\
&= u^T Pu - v^T Pv - < u - v, 2Pv > \\
&= \Phi_P(u) - \Phi_P(v) - < u - v, \nabla\Phi_P(v) >
\end{aligned} \tag{13}$$

where $\Phi_P(u) = u^T Pu$ is strictly convex and $< \cdot\ ,\ \cdot >$ is the inner product. Therefore, we can train a diffusion model using the hybrid loss to obtain the marginal score function. ■

In fact, the choice of the matrix M is mathematically arbitrary since $I + \omega M^T M$ is always positive definite, making it a valid divergence. In our method, we choose the M to be a lower triangular matrix of ones, which is equivalent to element-wise integration of velocity:

$$M\tau_0^{\mathbf{v}} \cdot \Delta t = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_T \end{pmatrix} \cdot \Delta t = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_T \end{pmatrix} \tag{14}$$

Next, we show that the hybrid loss can also be expanded into a reward-weighted diffusion loss for reinforcement learning.

Theorem V.1. *Optimal action $a \sim \pi^{k^\star}(a|s)$ in Eq. (8) can be generated by optimizing the weighted diffusion loss in Eq. (10) and solving the diffusion reverse process with the learned $\mathbf{v}^{k^\star}$.*

Proof: To prove that we can sample from the optimal policy in Eq. 8, we only need to show that the reward-weighted objective in Eq. 10 is equivalent to the corresponding score matching objective of the optimal policy distribution. For simplicity, we omit the state condition in the derivation.

$$\begin{aligned}
&\mathbb{E}_{\mathbf{v}\sim\pi^{k-1},\epsilon\sim p_\epsilon(\epsilon),t\sim p_t(t)}\left[\exp(\beta r)||\mathbf{v}_\theta - \mathbf{v}||_P^2\right] \\
&= \int_{\mathbf{v}}\int_{\epsilon,t} \exp(\beta r)||\mathbf{v}_\theta - \mathbf{v}||_P^2 \cdot \pi^{k-1}(\mathbf{v}) p_\epsilon(\epsilon) p_t(t) \mathrm{d}\epsilon\mathrm{d}t\mathrm{d}\mathbf{v} \\
&= Z\int_{\mathbf{v}}\int_{\epsilon,t} ||\mathbf{v}_\theta - \mathbf{v}||_P^2 \cdot \frac{\exp(\beta r)\pi^{k-1}(\mathbf{v})}{Z} \cdot p_\epsilon(\epsilon) p_t(t) \mathrm{d}\epsilon\mathrm{d}t\mathrm{d}\mathbf{v} \\
&= Z\int_{\mathbf{v}}\int_{\epsilon,t} ||\mathbf{v}_\theta - \mathbf{v}||_P^2 \cdot \pi^{k^\star}(\mathbf{v}) \cdot p_\epsilon(\epsilon) p_t(t) \mathrm{d}\epsilon\mathrm{d}t\mathrm{d}\mathbf{v} \\
&= Z\mathbb{E}_{\mathbf{v}\sim\pi^{k^\star},\epsilon\sim p_\epsilon(\epsilon),t\sim p_t(t)}\left[||\mathbf{v}_\theta - \mathbf{v}||_P^2\right]
\end{aligned} \tag{15}$$

where $Z = \int_{\mathbf{v}} \exp(\beta r)\pi^{k-1}(\mathbf{v})\mathrm{d}\mathbf{v}$ 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. Namely, we can draw samples from $\pi^{k^\star}$ by sampling via the $\mathbf{v}_\theta$ trained with the objective. ■

C. 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.

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 N_1 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:

$$\begin{aligned} S_{ADE} &= 100 \times Clip(1 - \frac{minADE}{Thresh_{ADE}}, 0, 1) \\ S_{FDE} &= 100 \times Clip(1 - \frac{minFDE}{Thresh_{FDE}}, 0, 1) \end{aligned} \tag{16}$$

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 N_1 trajectories:

$$\begin{aligned} Cost &= \frac{1}{N_1} \sum_{i=1}^{N_1} (Cost_{Acc} \times Acc + Cost_{Jerk} \times Jerk) \\ S_{Comfort} &= 100 \times Clip(1 - \frac{Cost}{Thresh_{Comfort}}, 0, 1) \end{aligned} \tag{17}$$

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

$$\begin{aligned} S_{Open-Loop} &= (1 - CR) \times (\sum_{m \in \mathcal{M}} \omega_m S_m) \\ \mathcal{M} &= \{ADE,\ FDE,\ Comfort\} \end{aligned} \tag{18}$$

In addition, to evaluate the multi-modal generation ability of the model, we consider the divergence of each rollout with N_2 generations, measured by the average distance of trajectory endpoints to their geometrical center:

$$Divergence\ Score = \frac{1}{N_2} \sum_{i=1}^{N_2} \| P_i^L - \frac{1}{N_2} \sum_{i=1}^{N_2} P_i^L \|_2 \tag{19}$$

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

- **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 Fig. (14), 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.

$$\begin{aligned} Success\ Rate =& w_1 * s_1 + w_2 * s_2 + w_3 * s_3 \\ &+ w_4 * s_4 + w_5 * s_5 + w_6 * s_6 \end{aligned} \tag{20}$$

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:

$$\begin{aligned} S_{center} &= 100 \times Clip(1 - \frac{k_{center}}{Thresh_{center}}, 0, 1) \\ S_{speed} &= 100 \times Clip(1 - \frac{k_{speed}}{Thresh_{speed}}, 0, 1) \end{aligned} \tag{21}$$

TABLE IV: 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 V: 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

Fig. 14: Real vehicle test route.

TABLE VI: Hyperparameters of HDP

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

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 IV.

2) Implementation Details: We provide the pseudocode for the hybrid loss, as well as the experimental setup details for imitation learning and reinforcement learning training.

- **Hybrid Loss Implementations**. The pseudocode for hybrid loss with detach is shown in Algorithm 1, implemented in torch.

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
```

- **Experimental Setup**. Based on the content in Section VI, we provide the following details of the experimental setup. We adopt the variance-preserving(VP) noise schedule following [55] 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. For RL training, 32 NVIDIA H20 GPUs were used for 8k steps. We report the other detailed setup in TABLE VI.