HAD: Combining Hierarchical Diffusion with Metric-Decoupled RL for End-to-End Planning

Wenhao Yao[1,2], Xinglong Sun[3], Zhenxin Li[1,2], Shiyi Lan[3], Zi Wang[3], Jose M. Alvarez[3], and Zuxuan Wu[1,2⋆]

[1] Institute of Trustworthy Embodied AI, Fudan University
[2] Shanghai Key Laboratory of Multimodal Embodied AI
[3] NVIDIA
william_yao_2000@outlook.com, zxwu@fudan.edu.cn

Abstract. End-to-end planning has emerged as a dominant paradigm for autonomous driving, where recent models often adopt a scoring-selection framework to choose trajectories from a large set of candidates, with diffusion-based decoding showing strong promise. However, directly selecting from the entire candidate space remains difficult to optimize, and Gaussian perturbations used in diffusion often introduce unrealistic trajectories that complicate the denoising process. In addition, for training these models, reinforcement learning (RL) has shown promise, but existing end-to-end RL approaches typically rely on a single coupled reward without structured signals, limiting optimization effectiveness. To address these challenges, we propose HAD, an end-to-end planning framework with a Hierarchical Diffusion Policy that decomposes planning into a coarse-to-fine process. To improve trajectory generation, we introduce Structure-Preserved Trajectory Expansion, which produces realistic candidates while maintaining kinematic structure. For policy learning, we develop Metric-Decoupled Policy Optimization (MDPO) to enable structured RL optimization across multiple driving objectives. Extensive experiments show that HAD achieves new state-of-the-art performance on both NAVSIM and HUGSIM, outperforming prior arts by a huge margin: +2.3 EPDMS on NAVSIM and +4.9 Route Completion on HUGSIM.

1 Introduction

Planning is a pivotal component in the autonomous driving pipeline. Recent advances have led to a paradigm shift toward end-to-end planning [4, 12, 26, 30, 61], where perception, prediction, and planning are integrated into a unified model that directly outputs the driving trajectory from raw sensor inputs. By eliminating intermediate hand-crafted modules, such approaches mitigate error accumulation and enable more holistic reasoning over the driving scene [12].

To enhance end-to-end planner performance, several works [4, 26, 28–30, 53] adopt a scoring-selection paradigm, where a policy ranks candidate trajectories from discretized action space represented by a fixed vocabulary of an-

⋆ Corresponding author.

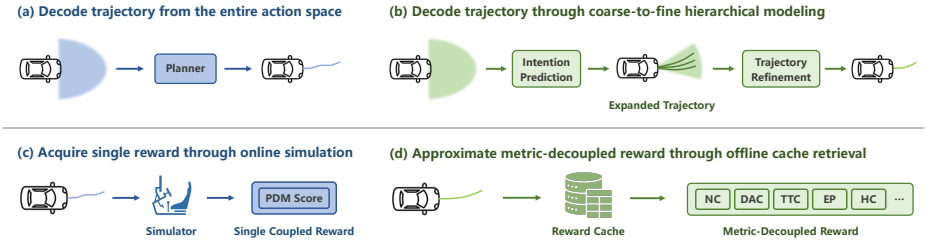

Fig. 1: Comparison with existing end-to-end planning methods. Prior approaches search the entire driving space and use a single coupled reward from online simulation. Our method narrows the search via Hierarchical Diffusion Policy and approximates metric-decoupled rewards through offline retrieval.

chors (e.g. 8192) and selects the best one. However, performance is fundamentally limited by the coverage and quality of the anchor set, causing failures in out-of-vocabulary scenarios [30]. To increase behavioral diversity, recent works [10, 15, 20, 25, 30, 32, 61] explore generation with anchored-diffusion, which samples additional trajectories with Gaussian noise perturbations around the pre-defined trajectory anchors.

Despite these advances, several limitations remain. First, as shown in Fig. 1 (a), the candidate action space is extremely large, making it a difficult optimization problem for a policy model to directly learn to rank and select from the entire set in a single pass [53]. Second, naive Gaussian noise sampling is not well suited for driving trajectories. Even in anchored diffusion setups [30], Gaussian perturbations often generate unrealistic motions, such as oscillatory or zigzag trajectories. These low-quality candidates introduce substantial noise into the optimization process and make the denoising stage more challenging.

Moreover, to train these end-to-end policy models, imitation learning [4, 6, 12, 30] has been the predominant paradigm. However, recent work has shown that imitation learning is fundamentally limited by the quality of human demonstrations [26, 28]. Therefore, some works [20, 28, 61] try to incorporate reinforcement learning (RL), where the policy learns to predict trajectories that maximize a reward signal reflecting driving quality, independent of human demonstrations.

Nevertheless, two challenges remain for RL in end-to-end driving. First, existing methods typically rely on a single coupled reward, such as the aggregated PDM score [7]. However, driving requires balancing multiple criteria—e.g., collision avoidance, lane keeping, etc—while a single scalar reward is often too coarse and can lead to reward hacking. Recent RL advances in the LLM domain, such as GDPO [31], explore multiple rewards but still aggregate them into a single objective. Second, reward computation must be efficient in practice. Existing approaches [20, 61] compute rewards on-the-fly via simulation during training, introducing huge computational overhead and reducing training efficiency.

To address the aforementioned limitations, we propose **HAD**. To reduce the difficulty of directly decoding a trajectory from the entire action space, we introduce a Hierarchical Diffusion Policy that follows the coarse-to-fine reasoning

process of human driving. The policy consists of two diffusion-based denoising stages, as shown in Fig. 1 (b). The first stage identifies a small set of plausible high-level driving intentions represented by trajectory anchors. The second stage then refines these candidates by performing low-level trajectory denoising, effectively "zooming in" on the plausible intentions to produce the final driving maneuver. To generate high-quality trajectory candidates around these intention anchors for low-level trajectory refinement, we further propose Structure-Preserved Trajectory Expansion, which maintains the intrinsic kinematic structure of driving trajectories. Instead of standard Gaussian noise sampling, which often disrupts trajectory structure, we project trajectory anchors from Cartesian space into polar coordinates and apply carefully designed radial scaling and angular offset perturbations. Finally, to better capture the multi-dimensional nature of driving objectives, we introduce Metric-Decoupled Policy Optimization (MDPO), an RL optimization framework that explicitly decouples training into multiple heads, each optimizing a specific driving metric. This design enables more structured and targeted optimization across different aspects of driving behavior. To further improve training efficiency, we also introduce a Trajectory Reward Retrieval mechanism that enables near-instant reward computation via pre-computation and caching of rewards on trajectory anchors combined with training-time nearest-neighbor matching, as shown in Fig. 1 (d).

We comprehensively evaluate HAD across multiple benchmarks. On the widely used NAVSIM [7] benchmark, HAD achieves 90.2 PDMS on NAVSIM v1 and 88.6 EPDMS on NAVSIM v2, outperforming the previous state-of-the-art method [18] by 2.3 points. In addition, HAD demonstrates strong closed-loop planning capability on the HUGSIM [59] benchmark, achieving a 47.5 Route Completion rate and a 30.8 HD-Score, surpassing prior art [28] by 4.9 points in Route Completion and 1.9 points in HD-Score.

Our contribution can be summarized as follows:

- We propose Hierarchical Diffusion Policy that decomposes planning into two stages: high-level intention establishment and low-level trajectory refinement, which reduces the optimization difficulty of trajectory selection from large action space.
- We propose Structure-Preserved Trajectory Expansion to sample high-quality trajectory candidates, maintaining kinematic structure for denoising.
- We develop Metric-Decoupled Policy Optimization (MDPO) to enable fine-grained RL training balancing multiple driving objectives, with an Offline Trajectory Reward Retrieval Scheme to efficiently acquire reward via pre-computed anchor rewards and nearest-neighbor retrieval during training.
- HAD achieves state-of-the-art performance on NAVSIM and HUGSIM.

2 Related Works

2.1 End-to-end Planning

End-to-end planning methods unify the autonomous driving pipeline from perception to planning into a single network, directly processing raw sensor inputs

to output the final driving trajectory. Early approaches [12, 16, 39, 43, 50] apply imitation learning. UniAD [12] was the first work to adopt the planning-oriented pipeline to center all other driving tasks around planning. Some methods [14, 16, 43] discover sparse scene representation to extract more valuable features for planning. However, the optimization objective of imitation learning is singular, making models risky of causal confusion problem. To resolve the limitation, selection-based methods [4, 26, 27, 33, 34, 41, 49, 53] introduce a predefined trajectory vocabulary, and further use multiple metrics to score each trajectory, selecting the most suitable candidate as the model output. Generative approaches [17,30,51,57,58] utilize VAE [19] or Diffusion Policy [5] to model dynamic trajectory distribution to reach multi-modal planning. Recently, to resolve some hard corner cases in driving, some other methods [11,25,38] utilize or distill the general VLM [1,46] knowledge to build robust and instruction-followed planners. The above methods directly predict trajectories from the whole unstructured, large driving space. Different from these one-stage approaches, our hierarchical modeling pipeline follows coarse-to-fine human driving logic and further eases the causal fusion problem.

2.2 Reinforcement Learning for Planning

Several approaches [9, 13, 18, 20, 24, 45, 52, 61] leverage reinforcement learning (RL) in planning to better align trajectory outputs with pre-defined driving rules. MaRLn [45] discovers the potential of RL on planning, but suffers from low training efficiency. RAD [9] establishes a virtual interactive driving environment based on 3DGS to adopt RL training. Some methods like DriveAdapter [13] and Raw2Drive [52] focus on narrowing the gap between raw sensor data and privileged data required for RL training. Other works [18,20,61] try to integrate RL training with the superior diffusion-based method. EvaDrive [18] introduces a multi-round adversarial RL paradigm to train trajectory generation and scoring models for given scenarios. DiffusionDriveV2 [61] proposes Intra-Anchor GRPO and Inter-Anchor GRPO to improve RL for trajectory anchors. These methods require closed-loop driving simulators or reconstructed environments to get a single ensembled trajectory reward, which is computationally expensive and causes policy fusion. Our Trajectory Reward Retrieval Scheme efficiently gets the reward during training, and the Metric-Decoupled Policy Optimization treats the reward as a multi-dimensional vector, improving the policy granularity.

2.3 Hierarchical Modeling

Hierarchical modeling is beneficial to progressively refine predictions from coarse to fine, enabling models to handle complex tasks more effectively while improving robustness and stability. It has been widely adopted in various domains of computer vision, like object detection [2,37,56,60], segmentation [22,40], and image generation [8,35,36,44]. A few methods [48,53,54] employ hierarchical modeling in end-to-end planning. However, they either employ hierarchical modeling in a fixed trajectory set [53] or ignore detailed exploration in the local region [48,54],

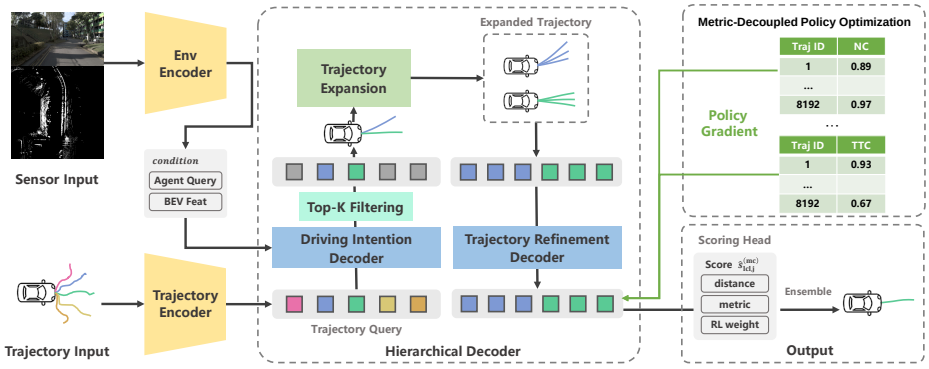

Fig. 2: Overview of HAD. The Hierarchical Diffusion Policy decomposes planning into Driving Intention Establishment and Local Trajectory Refinement. MDPO provides decoupled, structured optimization signals for training.

resulting in low-quality output trajectories. Our proposed Hierarchical Diffusion Policy explores the local region well through the Structure-preserved Trajectory Expansion Algorithm, leading to accurate output planning trajectories.

3 Methods

In this section, we present the proposed method HAD. We first introduce the Hierarchical Diffusion Policy with the Structure-Preserved Trajectory Expansion algorithm. Next, we present Metric-Decoupled Policy Optimization (MDPO), a reinforcement learning algorithm designed for end-to-end driving. Finally, we introduce an efficient reward retrieval scheme to improve training efficiency.

3.1 Hierarchical Diffusion Strategy

HAD introduces a novel hierarchical diffusion strategy, which treats the trajectory denoising process in a coarse-to-fine manner, decoupling trajectory planning into high-level Driving Intention Establishment and low-level Local Trajectory Refinement stages.

As illustrated in Fig. 2, the framework consists of two stages. First, the Driving Intention Decoder establishes high-level driving intentions by selecting a small set of plausible trajectory anchors. These anchors are then expanded using the Structure-Preserved Trajectory Expansion algorithm to generate diverse local maneuvers around each intention while preserving the intrinsic kinematic structure of trajectories. Second, the Trajectory Refinement Decoder performs fine-grained adjustments within this local region to produce the final trajectory.

Driving Intention Establishment This stage pre-defines a sparse set of M_1 trajectory anchors $\{\tau_j\}_{j=1}^{M_1}$ to roughly cover different driving intentions. During

training, a diffusion forward process is applied to these anchors:

$$\tilde{\tau}_j^{(i)} = \sqrt{\bar{\alpha}^{(i)}}\tau_j + \sqrt{1-\bar{\alpha}^{(i)}}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \tag{1}$$

where $\tau_j = \{(x_{j,t}, y_{j,t})\}_{t=1}^{T}$ denotes the j-th trajectory including T coordinates, $\bar{\alpha}^{(i)}$ represents the cumulative noise schedule at diffusion timestep $i \in [1, T_{\text{trunc}}]$, and $\boldsymbol{\epsilon}$ is the gaussian noise. HAD leverages a Transformer-based decoder [3, 47] to implement trajectory denoising. The encoded perturbed trajectories interact with the environment feature to produce refined trajectories and their corresponding confidence scores:

$$cond = \text{Enc}_{\text{env}}(Img, Lidar, Status) \tag{2}$$

$$\{f_j\}_{j=1}^{M_1} = \text{Enc}_{\text{traj}}\left(\{\tilde{\tau}_j\}_{j=1}^{M_1}\right) \tag{3}$$

$$\left\{(\hat{\tau}_{\text{gbl},j}, \hat{s}_{\text{gbl},j}^{(c)})\right\}_{j=1}^{M_1} = \text{Dec}_{\text{gbl}}\left(\{f_j\}_{j=1}^{M_1}, cond\right) \tag{4}$$

where Enc_{env} is a multi-modal environment encoder identical to the Transfuser [6] backbone, fusing features from images Img, LiDAR $Lidar$ and ego-vehicle status $Status$ to extract environmental feature $cond$, f_j is the j-th trajectory query encoded by trajectory MLP encoder Enc_{traj}, Dec_{gbl} is the Driving Intention Decoder. The decoder yields two critical outputs: the denoised trajectory $\hat{\tau}_{\text{gbl},j}$ and the classification score $\hat{s}_{\text{gbl},j}^{(c)}$. The score evaluates whether the denoised trajectory is in the same driving sub-region as the human trajectory. We select candidates with top-K classification score as the filtered trajectory set $\mathcal{T}_{\text{gbl}} = \{\hat{\tau}_{\text{gbl},j} \mid j \in \mathcal{I}_{\text{topK}}\}$, where $\mathcal{I}_{\text{topK}} = \text{argsort}_K(\{\hat{s}_{\text{gbl},j}^{(c)}\}_{j=1}^{M_1})$, representing the driving intention.

Structure-Preserved Trajectory Expansion To ensure a comprehensive exploration of the local driving region around the plausible intentions, HAD incorporates a Structure-Preserved Trajectory Expansion algorithm. This approach enriches trajectory diversity while strictly maintaining the intrinsic kinematic trajectory structure, easing the risk of trajectory structure damage caused by naive Gaussian noise sampling [5,20,30,61]. Given a trajectory $\tau = \{(x_t, y_t)\}_{t=1}^{T}$ and an expansion number n_{exp}, this algorithm outputs an expanded trajectory set $\mathcal{T}_{\text{exp}} = \{\tau_j\}_{j=1}^{n_{\text{exp}} \cdot n_{\text{exp}}}$. Specifically, the algorithm first converts the Cartesian coordinates of τ into the *polar domain*, represented as $\tau = \{(\rho_t, \theta_t)\}_{t=1}^{T}$. Moreover, we define a set of radial scaling coefficients $\{\lambda_u\}_{u=1}^{n_{\text{exp}}}$ and angular offset coefficients $\{\delta_v\}_{v=1}^{n_{\text{exp}}}$. The algorithm traverses these coefficients to apply linear transformations in the polar space, resulting in the expanded trajectories:

$$\rho_t^{(u,v)} = \lambda_u \cdot \rho_t, \quad \theta_t^{(u,v)} = \theta_t + \delta_v \tag{5}$$

$$x_t^{(u,v)} = \rho_t^{(u,v)} \cos\theta_t^{(u,v)}, \quad y_t^{(u,v)} = \rho_t^{(u,v)} \sin\theta_t^{(u,v)} \tag{6}$$

The resulting trajectory is denoted as $\tau^{(u,v)} = \{(x_t^{(u,v)}, y_t^{(u,v)})\}_{t=1}^{T}$.

We apply the trajectory expansion algorithm to the top-K candidates $\mathcal{T}_{\text{gbl}}$, yielding an expanded set $\mathcal{T}_{\text{gbl-exp}} = \{\tau_j\}_{j=1}^{M_2}$, where $M_2 = K \cdot n_{\text{exp}}^2$ denotes the total number of trajectories in the second stage.

Local Trajectory Refinement This stage performs fine-grained trajectory refinement over the expanded candidates to produce the final refined local driving maneuver. Given the expanded trajectory set $\mathcal{T}_{\text{gbl-exp}} = \{\tau_j\}_{j=1}^{M_2}$, the model employs the Trajectory Refinement Decoder Dec_{lcl} to refine these candidates:

$$\{g_j\}_{j=1}^{M_2} = \text{Enc}_{\text{traj}}\left(\{\tau_j\}_{j=1}^{M_2}\right) \tag{7}$$

$$\left\{(\hat{\tau}_{\text{lcl},j}, \hat{s}_{\text{lcl},j}^{(\text{mc})})\right\}_{j=1}^{M_2} = \text{Dec}_{\text{lcl}}\left(\{g_j\}_{j=1}^{M_2}, cond\right) \tag{8}$$

where *cond* represents the environmental features extracted in the first denoising stage, τ_j denotes the expanded trajectory, g_j is the encoded expanded trajectory query, and $\hat{\tau}_{\text{lcl},j}$ is the refined trajectory.

Besides decoding trajectories, HAD introduces multiple MLP heads in Dec_{lcl} to predict different metrics for each decoded trajectory, denoted as $\hat{s}_{\text{lcl},j}^{(\text{mc})}$. Specifically, the metric set mc covers three categories: (i) $\hat{s}_{\text{lcl},j}^{(\text{dist})}$, the distance to the expert trajectory, (ii) $\hat{s}_{\text{lcl},j}^{(m)}$, safety-critical metrics defined in the NAVSIM benchmark [7], and (iii) $\hat{s}_{\text{lcl},j}^{(m\text{-rl})}$, the metric-decoupled reinforcement learning weights (detailed in Sec. 3.2). The final planning output is derived through a weighted average of the refined candidates:

$$\hat{s}_{\text{lcl},j}^{(\text{pdms})} = \sum_{mp} \lambda^{(mp)} \log \hat{s}_{\text{lcl},j}^{(mp)} + \lambda_{\text{avg}} \log\left(\sum_{ma} \lambda^{(ma)} s_{\text{lcl},j}^{(ma)}\right) \tag{9}$$

$$\hat{s}_{\text{lcl},j} = \gamma_{\text{dist}} \hat{s}_{\text{lcl},j}^{(\text{dist})} + \gamma_{\text{pdms}} \hat{s}_{\text{lcl},j}^{(\text{pdms})} + \gamma_{\text{rl}} \hat{s}_{\text{lcl},j}^{(\text{rl})} \tag{10}$$

$$\hat{\tau} = \sum_j w_j \hat{\tau}_{\text{lcl},j}, \quad \text{where } w_j = \texttt{Softmax}(\hat{s}_{\text{lcl},j}) \tag{11}$$

where $\hat{s}_{\text{lcl},j}^{(\text{pdms})}$ is the linear combination of logarithm of different metric scores $\hat{s}_{\text{lcl},j}^{(m)}$, mp and ma are penalty metrics and average metrics. $\hat{s}_{\text{lcl},j}^{(\text{rl})}$ ensembles the metric-decoupled reinforcement learning weights $\hat{s}_{\text{lcl},j}^{(m\text{-rl})}$, following similar score ensemble procedure as $\hat{s}_{\text{lcl},j}^{(\text{pdms})}$. γ_{dist}, γ_{pdms}, and γ_{rl} are coefficients, and $\hat{\tau}_{\text{lcl},j}$ is the trajectory output from the Local Trajectory Refinement stage.

3.2 Metric-Decoupled Policy Optimization

To better align diffusion-decoded trajectories with safety-related driving rules, we employ reinforcement learning for policy optimization. Prior end-to-end RL approaches, such as DiffusionDriveV2 [61], typically rely on a single aggregated reward representing overall trajectory quality. However, such a coarse reward

signal can hinder effective optimization and may lead to reward hacking [42]. To address this issue, we propose Metric-Decoupled Policy Optimization (MDPO), which enables more structured optimization across multiple driving metrics.

Specifically, the first stage of the Hierarchical Diffusion Policy generates K candidate trajectories. Through a trajectory expansion algorithm, these candidates are transformed into K sets of local trajectories $\{\mathcal{T}_{\text{gbl-exp},k}\}_{k=1}^{K}$, where each set contains $M_{\text{sub}} = n_{\text{exp}}^2$ trajectories. For the k-th set $\mathcal{T}_{\text{gbl-exp},k} = \{\hat{\tau}_{\text{lcl},j}\}_{j\in\mathcal{I}_k}$, MDPO first computes the selection probability $p_j^{(m)}$ for each trajectory on each safety metric, then calculates the reward J_k for the k-th trajectory set:

$$p_j^{(m)} = \frac{\exp\left(\hat{s}_{\text{lcl},j}^{(m\text{-rl})}\right)}{\sum_{i\in\mathcal{I}_k}\exp\left(\hat{s}_{\text{lcl},i}^{(m\text{-rl})}\right)} \tag{12}$$

$$\bar{r}_j^{(m)} = \frac{r_j^{(m)} - \text{mean}\left(\{r_i^{(m)} \mid i\in\mathcal{I}_k\}\right)}{\text{std}\left(\{r_i^{(m)} \mid i\in\mathcal{I}_k\}\right) + \epsilon} \tag{13}$$

$$J_k = \sum_m \alpha^{(m)} \sum_{j\in\mathcal{I}_k} p_j^{(m)} \cdot \bar{r}_j^{(m)} \tag{14}$$

where m denotes the safety metrics, and $\hat{s}_{\text{lcl},j}^{(m\text{-rl})}$ is the RL logit predicted by the model for the j-th trajectory regarding the m-th metric. $\bar{r}_j^{(m)}$ represents the normalized reward for metric m, $\mathcal{I}_k$ denotes trajectory indices in the k-th set. Finally, $\alpha^{(m)}$ is the predefined coefficient for each metric, J_k is the reward of the k-th trajectory set. The total reward J is defined as the mean reward across all trajectory sets: $J = \frac{1}{K}\sum_{k=1}^{K} J_k$.

3.3 Efficient Offline Trajectory Reward Retrieval

Driving reward scores typically require running a simulator to compute safety metrics such as collision avoidance [61]. However, such a simulation is a CPU-intensive task and time-consuming, making it inefficient and difficult to scale when executed on-the-fly during training, as illustrated in Fig. 3 (a). Therefore, we propose an efficient Offline Trajectory Reward Retrieval Scheme based on driving area discretization, as shown in Fig. 3 (b). Inspired by recent trajectory-selection frameworks [26, 27], HAD incorporates a extensive trajectory vocabulary $\mathcal{V} = \{\tau_{\text{ref},j}\}_{j=1}^{N}$, providing dense N trajectories covering the whole feasible driving area. For each trajectory $\tau_{\text{ref},j}$ in the vocabulary $\mathcal{V}$, we calculate its safety metric scores $s_{\text{ref},j}^{(m)}$ in advance, and these scores are stored in a trajectory reward caching table. During online training, for an arbitrary output trajectory $\hat{\tau}$ predicted by the model, our scheme performs a nearest-neighbor query to find the closest reference trajectory τ_{ref,j^*} in $\mathcal{V}$:

$$\tau_{\text{ref},j^*} = \underset{\tau_{\text{ref},j}\in\mathcal{V}}{\arg\min}\,\text{Dist}(\hat{\tau}, \tau_{\text{ref},j}) \tag{15}$$

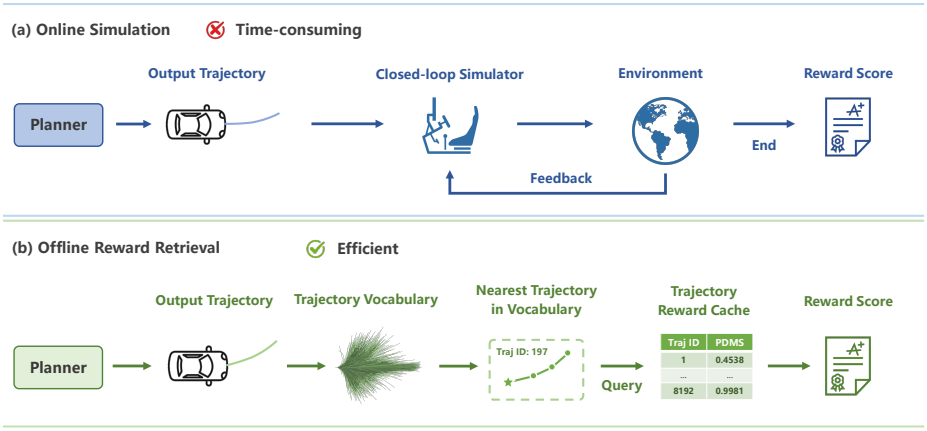

Fig. 3: Comparison between simulation-based trajectory evaluation and our proposed Offline Reward Retrieval scheme.

The corresponding scores $s^{(m)}_{\text{ref},j^*}$ are directly utilized as the approximation of the simulation score of $\hat{\tau}$.

This strategy converts expensive online simulation into an efficient nearest-neighbor trajectory retrieval from the offline cache. As a result, reinforcement learning rewards can be obtained much more efficiently, significantly improving model training efficiency.

3.4 Loss Function

The loss function of HAD consists of three components: the perception loss L_{percept}, the driving intent establishment loss L_{global}, and the local trajectory refinement loss L_{local}:

$$L = \lambda_{\text{percept}} L_{\text{percept}} + \lambda_{\text{global}} L_{\text{global}} + \lambda_{\text{local}} L_{\text{local}} \tag{16}$$

The perception loss L_{percept} follows the design of Transfuser [6], including the detection loss for 3D bounding box regression, the classification loss for object categories, and the semantic segmentation loss on bird's-eye-view (BEV) representation. The driving intent establishment loss L_{global} is employed to supervise the output of the first-stage decoder Dec_{gbl}, comprising the trajectory regression loss and the region classification loss, which determines whether the predicted trajectory and the ground-truth human trajectory belong to the same spatial region. The local trajectory refinement loss L_{local} aims to supervise the refined trajectories and their multi-dimensional quality metrics produced in the second stage, including the distance to the human expert trajectory, supervision for safety metrics, and the reinforcement learning (RL) objective. More details about the loss function are listed in the Appendix.

4 Experiments

4.1 Implementation Details

Datasets The experiments are mainly conducted on two datasets: NAVSIM [7] and HUGSIM [59]. NAVSIM is an open-loop planning benchmark. The output trajectory is judged by a simulator to determine whether this trajectory follows driving rules. The evaluation metric of NAVSIM is PDMS, which is aggregated by several driving sub-metrics:

$$\text{PDMS} = \left(\prod_{m \in S_{\text{pen}}} \text{s}_m \right) \times \left(\frac{\sum_{w \in S_{\text{avg}}} \text{w}_w \times \text{s}_w}{\sum_{w \in S_{\text{avg}}} \text{w}_w} \right), \tag{17}$$

where S_{pen} and S_{avg} denote penalty and weighted average metrics. NAVSIM develops two evaluation schemes: NAVSIM v1 and NAVSIM v2. In NAVSIM v1, the sub-metrics in S_{pen} include No Collisions (NC) and Drivable Area Compliance (DAC), while S_{avg} includes Ego Progress (EP), Time-to-Collision (TTC) and Comfort (C). NAVSIM v2 introduces 4 new sub-metrics: Driving Direction Compliance (DDC) and Traffic Light Compliance (TLC) belong to S_{pen}, while Lane Keeping (LK) and Extended Comfort (EC) belong to S_{avg}.

HUGSIM is a closed-loop planning benchmark covering over 400 simulation scenarios. These scenarios are categorized into four difficulty levels: Easy, Medium, Hard, and Extreme. HUGSIM provides a comprehensive metric, HD-Score, to evaluate the closed-loop performance of planning models across multiple dimensions including No Collision (NC), Driveable Area Compliance (DAC), Time-to-Collision (TTC), Comfort (COM), and Route Completion (R_c). During the simulation, the HD-Score at each timestep t is calculated as follows:

$$\text{HD-Score}_t = \left(\prod_{m \in \{NC, DAC\}} score_m \right) \cdot \left(\frac{\sum_{w \in \{TTC, COM\}} weight_w \cdot score_w}{\sum_{w \in \{TTC, COM\}} weight_w} \right), \tag{18}$$

the final HD-Score is obtained by averaging the scores across all timesteps.

Model Details The backbone of HAD is identical to Transfuser [6]. The environment encoder Enc_{env} adopts a dual-modal pipeline: both the image and LiDAR branches utilize a ResNet34 backbone. The resulting environment feature *cond* comprises encoded query vectors of other agents and an 8×8 BEV feature map. The trajectory encoder Enc_{traj} is a 2-layer MLP, while both the Driving Intention Decoder Dec_{gbl} and Trajectory Refinement Decoder Dec_{lcl} consist of 1 Transformer layer. For the Hierarchical Diffusion Policy, we set the number of predefined candidate trajectories M_1 to 20. After the first-stage denoising, $K = 2$ top-scoring trajectories are selected. In the trajectory expansion algorithm, the expansion factor n_{exp} is set to 5. The corresponding length coefficient set $\{\lambda_u\}$ is $\{0.92, 0.96, 1.0, 1.04, 1.08\}$, and the angular coefficient set $\{\delta_v\}$ is $\{-6°, -3°, 0°, 3°, 6°\}$. This process generates $M_{\text{sub}} = 25$ trajectories within

Table 1: Results on NAVSIM v1. "C+L" denotes that the model requires both RGB image and LiDAR as inputs, while "C" indicates the model only receives image input.

Method	Input	NC ↑	DAC ↑	EP ↑	TTC ↑	C ↑	PDMS ↑
Human	—	100	100	87.5	100	99.9	94.8
Transfuser [6]	C+L	97.7	92.8	79.2	92.8	100	84.0
UniAD [12]	C	97.8	91.9	78.8	92.9	100	83.4
VADv2 [4]	C	97.9	91.7	77.6	92.9	100	83.0
LAW [23]	C	96.4	95.4	81.7	88.7	99.9	84.6
DRAMA [55]	C+L	98.0	93.1	80.1	94.8	100	85.5
GoalFlow [51]	C+L	98.3	93.8	79.8	94.3	100	85.7
Hydra-MDP [26]	C+L	98.3	96.0	78.7	94.6	100	86.5
DiffusionDrive [30]	C+L	98.2	96.2	82.2	94.7	100	88.1
WoTE [24]	C+L	98.5	96.8	81.9	94.9	99.9	88.3
DriveSuprim [53]	C	97.8	97.3	86.7	93.6	100	<u>89.9</u>
HAD	C+L	98.2	97.3	87.4	97.5	100	**90.2**
HAD-L	C	98.1	97.2	87.2	97.3	100	<u>89.9</u>

Table 2: Results on NAVSIM v2. "C+L" denotes that the model requires both RGB image and LiDAR as inputs, while "C" indicates the model only receives image input.

Method	Input	NC ↑	DAC ↑	DDC ↑	TL ↑	EP ↑	TTC ↑	LK ↑	HC ↑	EC ↑	EPDMS ↑
Human	—	100	100	99.8	100	87.4	100	100	98.1	90.1	90.3
Ego Status MLP	—	93.1	77.9	92.7	99.6	86.0	91.5	89.4	98.3	85.4	64.0
Transfuser [6]	C+L	96.9	89.9	97.8	99.7	87.1	95.4	92.7	98.3	87.2	76.7
HydraMDP++ [21]	C	97.2	97.5	99.4	99.6	83.1	96.5	94.4	98.2	70.9	81.4
DriveSuprim [53]	C	97.5	96.5	99.4	99.6	88.4	96.6	95.5	98.3	77.0	83.1
DiffusionDriveV2 [61]	C+L	97.7	96.6	99.2	99.8	88.9	97.2	96.0	97.8	91.0	85.5
DiffRefiner [54]	C	98.5	97.4	99.6	99.8	87.6	97.7	97.7	98.3	86.2	86.2
EvaDrive [18]	C	98.8	98.5	98.9	99.8	96.6	98.4	94.3	97.8	55.9	86.3
HAD	C+L	98.2	97.3	99.2	99.8	87.4	97.5	95.2	98.3	86.2	**88.6**
HAD-L	C	98.1	97.2	99.3	99.8	87.2	97.3	95.3	98.3	86.4	<u>88.5</u>

each local region and $M_2 = 50$ trajectories in total. The trajectory vocabulary $\mathcal{V}$ for the Offline Reward Approximation Scheme consists of $N = 8192$ reference trajectories. The coefficients γ_{dist}, γ_{pdms}, and γ_{rl} in Equ. 10 are 0.6, 0.05, and 0.01. We train two model variants: HAD and HAD-L. HAD requires both image and LiDAR inputs. HAD-L follows Latent Transfuser [6], replacing LiDAR inputs with learnable positional embeddings, thus only requiring image inputs.

4.2 Quantitative Results

Result on NAVSIM Tab. 1 and Tab. 2 present the performance of HAD on the NAVSIM benchmark. On NAVSIM v1, HAD achieves 90.2 PDMS, outperforming DiffusionDrive by 2.1 PDMS. On the more challenging NAVSIM

Table 3: Results on the HUGSIM dataset. "RC" denotes the Route Completion rate, and "HDS" represents the HD-Score metric. The asterisk symbol "*" indicates that the results are evaluated on both public and private datasets; otherwise, the results refer to the performance on the public dataset only.

Method	Easy		Medium		Hard		Extreme		Overall	
	RC	HDS	RC	HDS	RC	HDS	RC	HDS	RC	HDS
UniAD* [12]	58.6	48.7	41.2	29.5	40.4	27.3	26.0	14.3	40.6	28.9
VAD* [16]	38.7	24.3	27.0	9.9	25.5	10.4	23.0	8.2	27.9	12.3
Latent TransFuser* [6]	68.4	52.8	40.7	24.6	36.9	19.8	25.5	8.1	41.4	24.8
Latent TransFuser [6]	60.4	42.5	39.4	17.7	32.7	11.8	27.9	10.6	37.9	18.0
GTRS-Dense [29]	64.2	55.5	50.0	39.0	20.7	11.7	22.3	14.3	38.0	28.6
ZTRS [28]	74.4	60.8	50.9	34.2	32.7	20.5	21.9	11.0	42.6	28.9
HAD-L	76.9	65.9	49.3	31.4	36.4	18.0	39.1	22.5	**47.5**	**30.8**

Table 4: Ablation on different trajectory evaluation metrics in hierarchical denoising stages.

Global Denoising			Local Denoising			EPDMS ↑
Dist.	Safety	RL	Dist.	Safety	RL	
✓			✓			86.7
✓			✓	✓		87.3
✓			✓	✓	✓	**88.6**
✓	✓	✓	✓	✓	✓	87.2

Table 5: Ablation on the number of first-stage selected trajectories and the number of expanded trajectories.

Training		Inference		EPDMS ↑
Top-K	n^2_{exp}	Top-K	n^2_{exp}	
1	5×5	1	5×5	88.1
2	5×5	2	5×5	88.5
2	5×5	2	7×7	**88.6**
4	5×5	4	5×5	86.4
20	5×5	20	5×5	79.8

v2 benchmark, HAD reaches 88.6 EPDMS, surpassing previous state-of-the-art methods such as DiffRefiner by 2.4 and EvaDrive by 2.3 PDMS, respectively. Furthermore, the camera-only variant HAD-L performs within 0.3% and 0.1% of HAD on NAVSIM v1 and v2, demonstrating the model has robustness across different input modalities.

Result on HUGSIM Tab. 3 presents the evaluation results of HAD-L on the high-fidelity closed-loop benchmark HUGSIM. HAD-L achieves 47.5 Route Completion and 30.8 HD-Score. Compared to ZTRS, HAD-L improves the RC and HD-Score by 4.9 and 1.9 points. The performance improvement becomes even more pronounced in challenging scenarios. In extreme scenarios characterized by frequent and aggressive agent attacks, HAD-L maintains an RC of 39.1 and an HD-Score of 22.5, significantly surpassing all baseline methods by a huge margin. These results highlight the model's exceptional driving capabilities in highly interactive, challenging environments.

4.3 Ablation Studies

Trajectory Metric Scores in Hierarchical Modeling In our hierarchical modeling, we use the distance towards the human expert trajectory to select top-

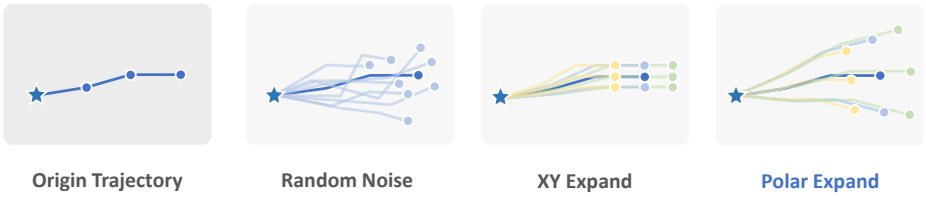

Fig. 4: Illustration of different trajectory expansion algorithms. Directly adding random noise is harmful to trajectory kinematic structure, expanding trajectory in the Cartesian space (XY Expand) suffer from insufficient exploration on local region, and expansion in the polar space (Polar Expand) leads to comprehensive exploration.

Table 6: Comparison of different trajectory expansion algorithms.

Expansion	NC ↑	DAC ↑	DDC ↑	TL ↑	EP ↑	TTC ↑	LK ↑	HC ↑	EC ↑	EPDMS ↑
Random Noise	98.1	93.3	99.3	99.8	87.6	96.9	95.6	98.3	85.4	84.9
XY Expand	98.2	93.8	99.4	99.8	87.6	97.1	95.4	98.3	87.5	85.9
Polar Expand	98.2	97.3	99.2	99.8	87.4	97.5	95.2	98.3	86.2	**88.6**

K candidates in the Driving Intention Establishment stage, while utilizing multiple metrics in the Local Trajectory Refinement in Equ. 8. The result in Tab. 4 shows the influence of different metric scores within the hierarchical framework, including the distance, safety metric scores, and RL weights. When the model only utilizes the distance to the human trajectory as supervision in trajectory refinement, it only achieves an 86.7 EPDMS. Incorporating the safety metric as scoring supervision raises the performance to 87.3 EPDMS. Furthermore, adding reinforcement learning training in the Local Trajectory Refinement stage further improves the EPDMS to 88.6. This indicates that using comprehensive metrics and adopting RL training helps the model learns a more robust driving policy aligning with driving rules. Moreover, introducing safety metric scoring and RL supervision in the Driving Intention Establishment stage leads to a performance drop to 87.2. This indicates that the whole unstructured driving space makes it difficult for complex reward signals to provide valuable information.

Trajectory Expansion Algorithm Our proposed Structure-preserved Trajectory Expansion Algorithm effectively explores the local driving region while preserving the trajectory kinematic structure. Tab. 5 and Tab. 6 further show the superiority. Tab. 5 shows the ablation on the number of first-stage filtered trajectories K and the expansion scale $n_{\text{exp}} \times n_{\text{exp}}$. The results show that constraining K to a small range (e.g., $K = 2$) significantly enhances model performance, achieving the optimal 88.6 EPDMS. When K is increased to include all 20 trajectories, the EPDMS drops to 79.8. This indicates the necessity to adopt hierarchical modeling in end-to-end planning. Furthermore, increasing the sampling density from 5×5 to 7×7 during inference yields a performance gain of 0.1 EPDMS. This suggests that a denser local area search during inference helps the model precisely locate better trajectories.

Table 7: Ablation on the effectiveness of MDPO. Results show that Metric-Decoupled Policy Optimization improves overall performance compared to using a coupled reward and optimization.

Reward	Decoupled	NC ↑	DAC ↑	DDC ↑	TL ↑	EP ↑	TTC ↑	LK ↑	HC ↑	EC ↑	EPDMS ↑
Dist.		92.0	91.2	96.6	99.5	88.7	91.6	85.1	41.3	3.9	62.6
PDMS		97.9	96.7	99.3	99.7	88.1	97.1	94.7	98.2	84.5	87.8
Dist. + PDMS		97.9	96.2	99.1	99.8	88.0	96.9	94.3	98.3	80.8	86.9
Decoupled Metrics	✓	98.2	97.3	99.2	99.8	87.4	97.5	95.2	98.3	86.2	**88.6**

Tab. 6 compares our trajectory expansion algorithm with other approaches illustrated in Fig. 4. Directly adding random noise to trajectories damages their intrinsic structure, only leading to 84.9 EPDMS. Expanding the trajectory in the Cartesian coordinate system [61] preserves the trajectory structure, but fails to provide a spatially balanced exploration. As shown in Fig. 4, when the trajectory y-coordinate is small, the exploration range along the y-axis becomes severely restricted. In comparison, our expansion in the polar coordinate system can make a sufficient coverage of the local area, leading to the best 88.6 EPDMS.

Superiority of MDPO Tab. 7 investigates the RL setting, showing the superiority of MDPO. Firstly, we validate the reward choice when adopting a single RL weight head. The first three rows in Tab. 7 reveal that adopting the comprehensive PDMS score as the reward function can help the model achieve the optimal 87.8 EPDMS. After adopting our proposed MDPO to decouple different metric rewards, the performance is further improved to 88.6 EPDMS, which shows that reward decoupling leads to a better driving policy.

4.4 Training Efficiency Analysis

According to our testing, the training efficiency of the model is significantly enhanced by the proposed Offline RL Reward Retrieval Scheme. By utilizing this retrieval mechanism, the reward acquisition latency for a single trajectory is reduced from 0.2449s to 0.0042s. Consequently, the total training duration is decreased from 64.4 hours to 13.6 hours. This shows that, compared with online simulation-based training such as DiffusionDriveV2 [61], our scheme can achieve a 5× speedup, demonstrating superior efficiency.

5 Conclusion

In this paper, we propose HAD to address the critical limitations of current end-to-end planners on the optimization burden in large action spaces and the challenges of online reinforcement learning. HAD introduces a Hierarchical Diffusion Policy to decompose planning into coarse-to-fine denoising stages, a Structure-preserved Trajectory Expansion algorithm to generate diverse trajectory candidates while keeping their kinematic structure, and a Metric-Decoupled Policy Optimization (MDPO) framework with an Offline Reward Retrieval Scheme

to efficiently learn a fine-grained driving policy. Experimental results on the NAVSIM and HUGSIM benchmarks demonstrate the superior performance of HAD on both open-loop and closed-loop planning.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant No. 62472098) and the Science and Technology Commission of Shanghai Municipality (No. 25511106100).

References

1. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., Ge, W., Guo, Z., Huang, Q., Huang, J., Huang, F., Hui, B., Jiang, S., Li, Z., Li, M., Li, M., Li, K., Lin, Z., Lin, J., Liu, X., Liu, J., Liu, C., Liu, Y., Liu, D., Liu, S., Lu, D., Luo, R., Lv, C., Men, R., Meng, L., Ren, X., Ren, X., Song, S., Sun, Y., Tang, J., Tu, J., Wan, J., Wang, P., Wang, P., Wang, Q., Wang, Y., Xie, T., Xu, Y., Xu, H., Xu, J., Yang, Z., Yang, M., Yang, J., Yang, A., Yu, B., Zhang, F., Zhang, H., Zhang, X., Zheng, B., Zhong, H., Zhou, J., Zhou, F., Zhou, J., Zhu, Y., Zhu, K.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025)
2. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: CVPR (June 2018)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV. p. 213–229 (2020)
4. Chen, S., Jiang, B., Gao, H., Liao, B., Xu, Q., Zhang, Q., Huang, C., Liu, W., Wang, X.: Vadv2: End-to-end vectorized autonomous driving via probabilistic planning. arXiv preprint arXiv:2402.13243 (2024)
5. Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B.C., Song, S.: Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In: Proceedings of Robotics: Science and Systems (July 2023)
6. Chitta, K., Prakash, A., Jaeger, B., Yu, Z., Renz, K., Geiger, A.: Transfuser: Imitation with transformer-based sensor fusion for autonomous driving **45**(11), 12878–12895 (2022)
7. Dauner, D., Hallgarten, M., Li, T., Weng, X., Huang, Z., Yang, Z., Li, H., Gilitschenski, I., Ivanovic, B., Pavone, M., Geiger, A., Chitta, K.: Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. In: NeurIPS. vol. 37, pp. 28706–28719 (2024)
8. Gafni, O., Polyak, A., Ashual, O., Sheynin, S., Parikh, D., Taigman, Y.: Make-a-scene: Scene-based text-to-image generation with human priors. In: ECCV. p. 89–106 (2022)
9. Gao, H., Chen, S., Jiang, B., Liao, B., Shi, Y., Guo, X., Pu, Y., Yin, H., Li, X., Zhang, X., Zhang, Y., Liu, W., Zhang, Q., Wang, X.: Rad: Training an end-to-end driving policy via large-scale 3dgs-based reinforcement learning. arXiv preprint arXiv:2502.13144 (2025)
10. Gao, Y., Jiang, A., Wang, Y., Jijun, W., Jiang, H., Sun, Z., Yuwen, H., Shuo, W., Zhao, H., Hao, S.: Diffvla++: Bridging cognitive reasoning and end-to-end driving through metric-guided alignment. arXiv preprint arXiv:2510.17148 (2025)

11. Hegde, D., Yasarla, R., Cai, H., Han, S., Bhattacharyya, A., Mahajan, S., Liu, L., Garrepalli, R., Patel, V.M., Porikli, F.: Distilling Multi-Modal Large Language Models for Autonomous Driving. In: CVPR. pp. 27575–27585 (2025)
12. Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., et al.: Planning-oriented autonomous driving. In: CVPR. pp. 17853–17862 (2023)
13. Jia, X., Gao, Y., Chen, L., Yan, J., Liu, P.L., Li, H.: Driveadapter: Breaking the coupling barrier of perception and planning in end-to-end autonomous driving. In: ICCV. pp. 7953–7963 (2023)
14. Jia, X., You, J., Zhang, Z., Yan, J.: Drivetransformer: Unified transformer for scalable end-to-end autonomous driving. In: ICLR (2025)
15. Jiang, A., Gao, Y., Sun, Z., Wang, Y., Wang, J., Chai, J., Cao, Q., Heng, Y., Jiang, H., Dong, Y., et al.: Diffvla: Vision-language guided diffusion planning for autonomous driving. arXiv preprint arXiv:2505.19381 (2025)
16. Jiang, B., Chen, S., Xu, Q., Liao, B., Chen, J., Zhou, H., Zhang, Q., Liu, W., Huang, C., Wang, X.: Vad: Vectorized scene representation for efficient autonomous driving. In: ICCV. pp. 8340–8350 (2023)
17. Jiang, C., Cornman, A., Park, C., Sapp, B., Zhou, Y., Anguelov, D.: Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In: CVPR. pp. 9644–9653 (June 2023)
18. Jiao, S., Qian, K., Ye, H., Zhong, Y., Luo, Z., Jiang, S., Huang, Z., Fang, Y., Miao, J., Fu, Z., Wang, Y., Jiang, K., Yang, D., Fan, R., Peng, B.: Evadrive: Evolutionary adversarial policy optimization for end-to-end autonomous driving. arXiv preprint arXiv:2508.09158 (2025)
19. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
20. Li, H., Li, T., Yang, J., Tian, H., Wang, C., Shi, L., Shang, M., Lin, Z., Wu, G., Hao, Z., et al.: Plannerrft: Reinforcing diffusion planners through closed-loop and sample-efficient fine-tuning. arXiv preprint arXiv:2601.12901 (2026)
21. Li, K., Li, Z., Lan, S., Liu, J., Xie, Y., zhizhong zhang, Wu, Z., Yu, Z., Alvarez, J.M.: Hydra-MDP++: Advancing end-to-end driving via hydra-distillation with expert-guided decision analysis (2024)
22. Li, L., Zhou, T., Wang, W., Li, J., Yang, Y.: Deep hierarchical semantic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1246–1257 (2022)
23. Li, Y., Fan, L., He, J., Wang, Y., Chen, Y., Zhang, Z., Tan, T.: Enhancing end-to-end autonomous driving with latent world model. In: ICLR (2025)
24. Li, Y., Wang, Y., Liu, Y., He, J., Fan, L., Zhang, Z.: End-to-end driving with online trajectory evaluation via bev world model. In: ICCV. pp. 27137–27146 (2025)
25. Li, Y., Xiong, K., Guo, X., Li, F., Yan, S., Xu, G., Zhou, L., Chen, L., Sun, H., Wang, B., et al.: Recogdrive: A reinforced cognitive framework for end-to-end autonomous driving. arXiv preprint arXiv:2506.08052 (2025)
26. Li, Z., Li, K., Wang, S., Lan, S., Yu, Z., Ji, Y., Li, Z., Zhu, Z., Kautz, J., Wu, Z., et al.: Hydra-mdp: End-to-end multimodal planning with multi-target hydra-distillation. arXiv preprint arXiv:2406.06978 (2024)
27. Li, Z., Wang, S., Lan, S., Yu, Z., Wu, Z., Alvarez, J.M.: Hydra-next: Robust closed-loop driving with open-loop training. In: ICCV. pp. 27305–27314 (October 2025)
28. Li, Z., Yao, W., Wang, Z., Sun, X., Chen, J., Chang, N., Shen, M., Song, J., Wu, Z., Lan, S., et al.: Ztrs: Zero-imitation end-to-end autonomous driving with trajectory scoring. arXiv preprint arXiv:2510.24108 (2025)

29. Li, Z., Yao, W., Wang, Z., Sun, X., Chen, J., Chang, N., Shen, M., Wu, Z., Lan, S., Alvarez, J.M.: Generalized trajectory scoring for end-to-end multimodal planning. arXiv preprint arXiv:2506.06664 (2025)
30. Liao, B., Chen, S., Yin, H., Jiang, B., Wang, C., Yan, S., Zhang, X., Li, X., Zhang, Y., Zhang, Q., Wang, X.: Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. In: CVPR (June 2025)
31. Liu, S.Y., Dong, X., Lu, X., Diao, S., Belcak, P., Liu, M., Chen, M.H., Yin, H., Wang, Y.C.F., Cheng, K.T., et al.: Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. arXiv preprint arXiv:2601.05242 (2026)
32. Liu, S., Chen, W., Li, W., Wang, Z., Yang, L., Huang, J., Zhang, Y., Huang, Z., Cheng, Z., Yang, H.: Bridgedrive: Diffusion bridge policy for closed-loop trajectory planning in autonomous driving. arXiv preprint arXiv:2509.23589 (2025)
33. Phan-Minh, T., Grigore, E.C., Boulton, F.A., Beijbom, O., Wolff, E.M.: Covernet: Multimodal behavior prediction using trajectory sets. In: CVPR (2020)
34. Philion, J., Fidler, S.: Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In: ECCV (2020)
35. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125 (2022)
36. Razavi, A., van den Oord, A., Vinyals, O.: Generating diverse high-fidelity images with vq-vae-2. In: NeurIPS (2019)
37. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) NeurIPS (2015)
38. Shao, H., Hu, Y., Wang, L., Song, G., Waslander, S.L., Liu, Y., Li, H.: Lmdrive: Closed-loop end-to-end driving with large language models. In: CVPR. pp. 15120–15130 (June 2024)
39. Shao, H., Wang, L., Chen, R., Waslander, S.L., Li, H., Liu, Y.: Reasonnet: End-to-end driving with temporal and global reasoning. In: CVPR. pp. 13723–13733 (2023)
40. Shi, C., Zhang, Y., Yang, B., Tang, J., Ma, Y., Yang, S.: Part2object: Hierarchical unsupervised 3d instance segmentation. In: ECCV. pp. 1–18. Springer (2024)
41. Sima, C., Chitta, K., Yu, Z., Lan, S., Luo, P., Geiger, A., Li, H., Alvarez, J.M.: Centaur: Robust end-to-end autonomous driving with test-time training. arXiv preprint arXiv:2503.11650 (2025)
42. Skalse, J., Howe, N., Krasheninnikov, D., Krueger, D.: Defining and characterizing reward gaming. Advances in Neural Information Processing Systems **35**, 9460–9471 (2022)
43. Sun, W., Lin, X., Shi, Y., Zhang, C., Wu, H., Zheng, S.: Sparsedrive: End-to-end autonomous driving via sparse scene representation. pp. 8795–8801 (2025)
44. Tian, K., Jiang, Y., Yuan, Z., PENG, B., Wang, L.: Visual autoregressive modeling: Scalable image generation via next-scale prediction. In: NeurIPS (2024)
45. Toromanoff, M., Wirbel, E., Moutarde, F.: End-to-end model-free reinforcement learning for urban driving using implicit affordances. In: CVPR. pp. 7153–7162 (2020)
46. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)

47. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
48. Wang, Z., Teng, J., Xiang, C., Chen, K., Pan, X., Deng, L., Gu, W.: Cognitive-hierarchy guided end-to-end planning for autonomous driving. arXiv preprint arXiv:2505.21581 (2025)
49. Wang, Z., Lan, S., Sun, X., Chang, N., Li, Z., Yu, Z., Alvarez, J.M.: Enhancing autonomous driving safety with collision scenario integration. arXiv preprint arXiv:2503.03957 (2025)
50. Weng, X., Ivanovic, B., Wang, Y., Wang, Y., Pavone, M.: Para-drive: Parallelized architecture for real-time autonomous driving. In: CVPR. pp. 15449–15458 (2024)
51. Xing, Z., Zhang, X., Hu, Y., Jiang, B., He, T., Zhang, Q., Long, X., Yin, W.: Goalflow: Goal-driven flow matching for multimodal trajectories generation in end-to-end autonomous driving. In: CVPR. pp. 1602–1611 (June 2025)
52. Yang, Z., Jia, X., Li, Q., Yang, X., Yao, M., Yan, J.: Raw2drive: Reinforcement learning with aligned world models for end-to-end autonomous driving (in carla v2). arXiv preprint arXiv:2505.16394 (2025)
53. Yao, W., Li, Z., Lan, S., Wang, Z., Sun, X., Alvarez, J.M., Wu, Z.: Drivesuprim: Towards precise trajectory selection for end-to-end planning. arXiv preprint arXiv:2506.06659 (2025)
54. Yin, L., Ju, R., Guo, G., Cheng, E.: Diffrefiner: Coarse to fine trajectory planning via diffusion refinement with semantic interaction for end to end autonomous driving. arXiv preprint arXiv:2511.17150 (2025)
55. Yuan, C., Zhang, Z., Sun, J., Sun, S., Huang, Z., Lee, C.D.W., Li, D., Han, Y., Wong, A., Tee, K.P., et al.: Drama: An efficient end-to-end motion planner for autonomous driving with mamba. arXiv preprint arXiv:2408.03601 (2024)
56. Zhang, H., Li, F., Liu, S., Zhang, L., Su, H., Zhu, J., Ni, L., Shum, H.Y.: DINO: DETR with improved denoising anchor boxes for end-to-end object detection. In: ICLR (2023)
57. Zheng, W., Song, R., Guo, X., Zhang, C., Chen, L.: Genad: Generative end-to-end autonomous driving. In: ECCV (2024)
58. Zheng, Y., Liang, R., ZHENG, K., Zheng, J., Mao, L., Li, J., Gu, W., Ai, R., Li, S.E., Zhan, X., Liu, J.: Diffusion-based planning for autonomous driving with flexible guidance. In: ICLR (2025)
59. Zhou, H., Lin, L., Wang, J., Lu, Y., Bai, D., Liu, B., Wang, Y., Geiger, A., Liao, Y.: Hugsim: A real-time, photo-realistic and closed-loop simulator for autonomous driving. arXiv preprint arXiv:2412.01718 (2024)
60. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. In: ICLR (2021)
61. Zou, J., Chen, S., Liao, B., Zheng, Z., Song, Y., Zhang, L., Zhang, Q., Liu, W., Wang, X.: Diffusiondrivev2: Reinforcement learning-constrained truncated diffusion modeling in end-to-end autonomous driving. arXiv preprint arXiv:2512.07745 (2025)