SparseDriveV2: Scoring is All You Need for End-to-End Autonomous Driving

Wenchao Sun[1,2], Xuewu Lin[3], Keyu Chen[1], Zixiang Pei[2], Xiang Li[2], Yining Shi[1], and Sifa Zheng[1]

[1] Tsinghua University, Beijing, China
swc21@mails.tsinghua.edu.cn
zsf@mail.tsinghua.edu.cn
[2] Horizon Continental Technology, Beijing, China
[3] Horizon, Beijing, China

Abstract. End-to-end multi-modal planning has been widely adopted to model the uncertainty of driving behavior, typically by scoring candidate trajectories and selecting the optimal one. Existing approaches generally fall into two categories: scoring a large static trajectory vocabulary, or scoring a small set of dynamically generated proposals. While static vocabularies often suffer from coarse discretization of the action space, dynamic proposals provide finer-grained precision and have shown stronger empirical performance on existing benchmarks. However, it remains unclear whether dynamic generation is fundamentally necessary, or whether static vocabularies can already achieve comparable performance when they are sufficiently dense to cover the action space. In this work, we start with a systematic scaling study of Hydra-MDP, a representative scoring-based method, revealing that performance consistently improves as trajectory anchors become denser, without exhibiting saturation before computational constraints are reached. Motivated by this observation, we propose SparseDriveV2 to push the performance boundary of scoring-based planning through two complementary innovations: (1) a scalable vocabulary representation with a factorized structure that decomposes trajectories into geometric paths and velocity profiles, enabling combinatorial coverage of the action space, and (2) a scalable scoring strategy with coarse factorized scoring over paths and velocity profiles followed by fine-grained scoring on a small set of composed trajectories. By combining these two techniques, SparseDriveV2 scales the trajectory vocabulary to be 32× denser than prior methods, while still enabling efficient scoring over such super-dense candidate set. With a lightweight ResNet-34 as backbone, SparseDriveV2 achieves 92.0 PDMS and 90.1 EPDMS on NAVSIM, with 89.15 Driving Score and 70.00 Success Rate on Bench2Drive. Code and model are released at https://github.com/swc-17/SparseDriveV2.

Keywords: End-to-end Autonomous Driving · Scoring-based Planning

Table 1: Systematic scaling study of static vocabulary density based on Hydra-MDP [18]. We report EPDMS on NAVSIM v2 and training memory consumption on NVIDIA L20 GPU with 48 GB memory. EPDMS improvements are computed relative to the previous scale.

#Anchors	EPDMS	Memory (MB)
1024	85.02	9531
2048	85.80 (+0.78)	11451
4096	86.33 (+0.53)	15513
8192	86.78 (+0.45)	23261
16384	87.35 (+0.57)	38877
32768	–	OOM

1 Introduction

End-to-end autonomous driving has recently emerged as a promising paradigm that unifies all driving tasks into a single holistic model, enabling direct optimization toward planning objectives [8, 14, 28]. Owing to the inherent uncertainty and non-deterministic nature of driving behaviors, multi-modal planning has been widely adopted in end-to-end driving [1, 18], which typically scores a set of candidate trajectories and selects the optimal one.

Based on how candidate trajectories are constructed, existing methods can be broadly categorized into two classes. The first class relies on a static trajectory vocabulary [1, 16, 18, 35], typically obtained by clustering trajectories from large-scale driving datasets. Given a predefined vocabulary, the planner simply scores all candidates and selects the trajectory with the highest score. However, its performance is often constrained by the coarse discretization of the action space. Due to memory and computational limits, the vocabulary size is usually restricted to only several thousand anchors, which fails to adequately cover the action space in complex driving scenarios.

The second class seeks to overcome this limitation through dynamic trajectory generation, such as regression-based methods [6] [33] or diffusion-based approaches [22]. By generating trajectories conditioned on scene information, these methods enable more flexible and expressive candidate proposals, allowing for finer-grained adaption and broader coverage of the action space. When further combined with advanced scoring modules [35], dynamic generation methods have demonstrated stronger empirical performance on existing benchmarks [39]. Nevertheless, these gains typically come at the cost of increased model complexity, including additional network components or iterative denoising procedures.

This naturally raises a fundamental yet largely underexplored question: is dynamic trajectory generation truly necessary for high-performance end-to-end planning, or equivalently, can static trajectory vocabularies already achieve comparable performance when they are sufficiently dense and properly scored?

In this work, we revisit scoring-based planning from a scaling perspective. We conduct a systematic study on the effect of vocabulary density with Hydra-

MDP [18] as a representative method. As shown in Tab. 1, as the number of trajectory anchors increases, planning performance consistently improves, without exhibiting saturation before computational constraints are reached. This finding suggests that the perceived limitations of static vocabularies are not intrinsic, but instead stem from insufficient coverage of the action space under computational constraint. Consequently, the key challenge for advancing scoring-based planning lies in how to (1) represent an extremely dense trajectory vocabulary to adequately cover the action space, and (2) efficiently score such a super-dense vocabulary under computational budgets.

Motivated by this observation, we propose SparseDriveV2 to push the performance boundary of scoring-based planning through a scalable vocabulary representation and a scalable scoring strategy. Rather than simply increasing vocabulary size, we introduce a scalable vocabulary representation that factorizes spatiotemporal trajectories into spatial-level geometric paths and temporal-level velocity profiles. The factorized representation enables combinatorial composition, substantially expanding trajectory coverage while preserving a compact structure of vocabulary. To make scoring over such super-dense vocabulary computationally feasible, we further design a scalable scoring strategy that first performs coarse factorized scoring independently over paths and velocity profiles to prune low-quality candidates, followed by fine-grained scoring on a small set of composed trajectories. This strategy enables precise spatiotemporal reasoning to be performed only on a small subset of high-quality trajectories, regardless of the full vocabulary size.

By combining above design, SparseDriveV2 scales the trajectory vocabulary to be 32× denser than prior methods (1024×256 vs. 8192), while still enabling efficient scoring over such super-dense candidate set. As a result, SparseDriveV2 achieves SOTA performance on NAVSIM [3] and Bench2Drive [12] benchmarks.

Our contributions are summarized as follows:

- We revisit scoring-based planning from a scaling perspective with a systematic study, revealing the bottlenecks that limit the performance of existing methods.
- We propose a scalable vocabulary representation that factorizes trajectories into geometric paths and velocity profiles, enabling combinatorial composition to expand trajectory coverage while preserving a compact structure of vocabulary.
- We introduce a scalable scoring strategy that performs coarse factorized scoring over paths and velocity profiles followed by fine-grained scoring on a small set of composed trajectories, making it feasible to score over super-dense vocabulary efficiently.
- Combining scalable vocabulary and scalable scoring, we present SparseDriveV2, achieving SOTA performance on challenging benchmarks and setting a new record for scoring-based method.

2 Related Works

2.1 Scoring-Based Planning Methods

Scoring-based methods learn a scorer to select an optimal trajectory from a predefined set of candidates. A prominent example is VADv2 [1], which formulates end-to-end autonomous driving as a probabilistic planning problem and learns the action distribution from large-scale demonstration data. The Hydra-MDP [16, 18] series extends single-target imitation learning to multi-target settings by distilling knowledge from both human experts and rule-based teachers. DriveSuprim [35] further enhances fine-grained discrimination capability through a coarse-to-fine refinement module. While being conceptually simple, the performance of these methods is often limited by the coarse discretization of the trajectory space, since the computational cost of scoring increases with the size of the trajectory vocabulary.

2.2 Dynamic Trajectory Generation Methods

Dynamic trajectory generation approaches aim to enhance the expressiveness of the candidate set by generating trajectory proposals conditioned on scene information, rather than relying on a fixed trajectory vocabulary. One class of methods adopts a regression-based paradigm, in which trajectory proposals are directly predicted or iteratively refined through regression modules. For instance, ipad [6] centers the planning pipeline around a sparse set of candidate proposals and progressively refines them via proposal-anchored attention, followed by a scoring module to select the optimal trajectory.

Another class of approaches leverage generative models to sample diverse and fine-grained trajectory candidates. Diffusion-based planners, such as DiffusionDrive [22], employs a truncated diffusion policy to denoise trajectory samples from an anchored Gaussian distribution, enabling the modeling of complex multimodal behaviors beyond fixed anchors. DiffusionDriveV2 further incorporates reinforcement learning to suppress low-quality modes while encouraging the exploration of higher-quality trajectories. By integrating advanced scoring modules [35], DiffusionDriveV2 outperforms scoring-based methods relying on static vocabularies. GoalFlow [34], a representative flow-matching approach, constrains the generation process using goal points to produce high-quality multimodal trajectories. Despite their strong empirical performance, these methods typically require additional network components dedicated to trajectory generation, which increases the overall model complexity.

2.3 Hybrid Methods

One notable method is GTRS [20], which investigates the limitations of both static and dynamic candidate sets in end-to-end multimodal planning. GTRS observes that static vocabularies, while providing coarse discretization, lack the

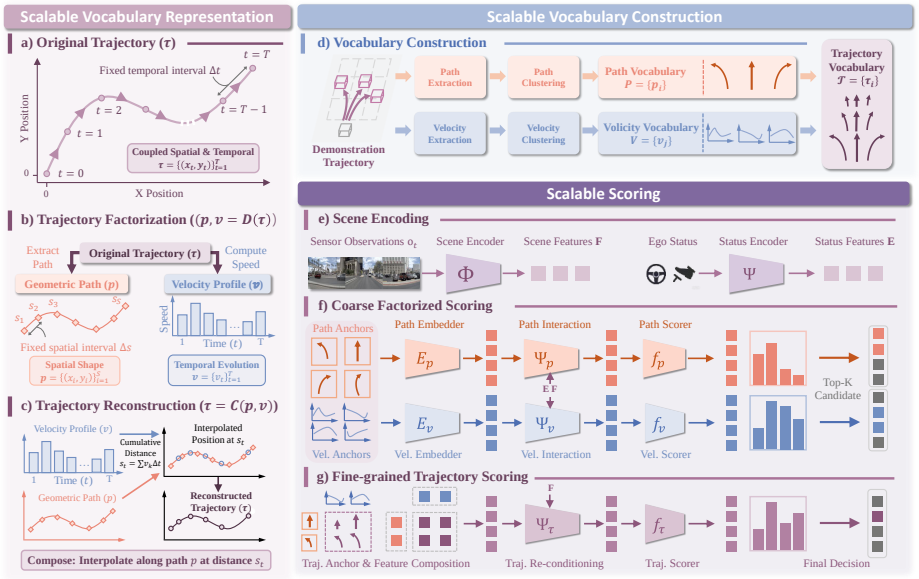

Fig. 1: Overview of the SparseDriveV2 framework. SparseDriveV2 factorizes (a) a spatiotemporal trajectory into (b) a geometric path and a velocity profile, and reconstructs the trajectory by (c) composing the two components. This representation enables (d) a super-dense trajectory vocabulary constructed from a compact set of paths and velocity profiles. Conditioned on (e) scene features, the scalable scoring strategy first performs (f) coarse factorized scoring over paths and velocity profiles to select top-k candidates, followed by (g) fine-grained scoring over the composed trajectories to produce the final planning decision.

flexibility to adapt to the fine-grained variations required in diverse driving scenarios. Conversely, a small number of dynamic proposals often fail to generalize to unseen trajectories and fail to capture broader trajectory distribution. To mitigate these issues, GTRS proposes a generalized scoring framework that combines diffusion-based dynamic proposals and static vocabulary, with a robust scorer trained using vocabulary dropout and sensor augmentation. While these innovations yield strong performance and achieve superior result, they also introduce additional complexity through multiple specialized sub-modules. In contrast, our approach adheres to a purely scoring-based paradigm without dynamic trajectory generation.

3 Methods

3.1 Problem Formulation

End-to-End Planning. An end-to-end autonomous driving system takes raw sensor observations o_t as input and predicts a future trajectory for the ego ve-

hicle. A trajectory is represented as a sequence of waypoints sampled at a fixed temporal interval Δt:

$$\tau = \{(x_t, y_t)\}_{t=1}^{T}, \tag{1}$$

where T denotes the planning horizon, and (x_t, y_t) represents the location of the ego vehicle at time step t in the current ego-centric coordinate system.

Scoring-Based Planning. Scoring-based planning predefines a trajectory vocabulary $\mathcal{T} = \{\tau_i\}_{i=1}^{N}$, where N denotes the number of trajectory anchors in the vocabulary. Given a specific driving scenario, a scorer evaluates the quality of each candidate trajectory according to predefined metrics, such as the ℓ_2 distance to human demonstrations, safety-related metrics, or traffic rule adherence. The scorer produces a trajectory score s_i for each candidate, and the optimal trajectory with the highest score is selected for vehicle control:

$$\tau^* = \arg\max_{\tau \in \mathcal{T}} s(\tau, o_t). \tag{2}$$

In this work, our method is also formulated as a scoring-based planning framework, as shown in Fig. 1, and we will introduce the three core components in the following sections.

3.2 Scalable Vocabulary Representation

A trajectory inherently couples spatial geometry and temporal evolution, describing where the vehicle goes and how fast it progresses over time. Covering the full action space with monolithic spatiotemporal trajectories therefore requires an extremely large number of anchors, which quickly becomes impractical under memory and computational constraints. To address this challenge, we introduce a vocabulary representation with factorized structure, which decouples trajectory into a spatial component and a temporal component, enabling combinatorial coverage of the action space with a compact vocabulary.

Trajectory Factorization. We factorize a trajectory into a geometric path and a velocity profile. A path is defined as a sequence of spatial waypoints

$$p = \{(x_i, y_i)\}_{i=1}^{S}, \tag{3}$$

where consecutive waypoints are sampled at a fixed spatial interval Δs along the path, and S denotes the number of spatial waypoints. The path encodes the geometric shape of motion but does not contain any temporal information. In contrast, a velocity profile is defined as a temporal sequence of scalar speeds

$$v = \{v_t\}_{t=1}^{T}, \tag{4}$$

where each v_t represents the average speed over a fixed temporal interval Δt. The velocity profile specifies how fast the vehicle progresses along a path over time, independent of the spatial geometry.

From Trajectory to Path and Velocity. Given a trajectory, we extract the corresponding path and velocity profile:

$$(p, v) = \mathcal{D}(\tau). \tag{5}$$

To obtain the path, we first specify a target spatial horizon $S_{\max}$ that defines the desired maximum path length. We then compute the cumulative traveled distance along the trajectory and resample trajectory positions at a fixed spatial interval Δs using interpolation. This results in a geometric path representation that captures the spatial shape of motion independent of timing. If the available trajectory does not cover the target spatial horizon $S_{\max}$, a validity mask is applied to indicate missing path segments. The velocity profile is obtained by computing the average speed between consecutive trajectory points. Let $(x_0, y_0) = (0, 0)$ denote the current ego position, the velocity profile is defined as

$$v_t = \frac{\|(x_t, y_t) - (x_{t-1}, y_{t-1})\|}{\Delta t}, \quad t = 1, \ldots, T. \tag{6}$$

From Path and Velocity to Trajectory. Given a path and a velocity profile, a spatiotemporal trajectory can be reconstructed by composing the two components. Specifically, the cumulative traveled distance at time step t is computed as

$$s_t = \sum_{k=1}^{t} v_k \, \Delta t. \tag{7}$$

The trajectory position at time t is then obtained by interpolating along the path at distance s_t. This composition defines a trajectory

$$\tau = \mathcal{C}(p, v), \tag{8}$$

where $\mathcal{C}(\cdot)$ denotes the path–velocity composition operator.

3.3 Scalable Vocabulary Construction

We construct the trajectory vocabulary from large-scale human driving demonstrations. Leveraging the proposed factorized representation, we separately build a path vocabulary and a velocity vocabulary, which can then be composed to form a dense set of trajectories.

Path Vocabulary. To construct the path vocabulary, we extract future trajectories from the training data with a sufficiently long temporal horizon to ensure coverage of the target spatial horizon $S_{\max}$. Each trajectory is then converted into a geometric path of length $S_{\max}$ following the procedure described in the previous section. The extracted paths are clustered using the K-Means algorithm to obtain a set of representative path anchors, forming the path vocabulary.

$$\mathcal{P} = \{p_i\}_{i=1}^{N_p}, \tag{9}$$

where N_p denotes the number of path anchors.

Velocity Vocabulary. The velocity vocabulary is constructed in a similar manner. For each trajectory, we compute its velocity profile according to Eq. (6), resulting in a sequence of scalar speeds over the planning horizon T. These velocity profiles are then clustered to obtain a set of representative velocity anchors, forming the velocity vocabulary

$$\mathcal{V} = \{v_j\}_{j=1}^{N_v}, \tag{10}$$

where N_v denotes the number of velocity anchors.

Trajectory Vocabulary via Composition. Given the path and velocity vocabularies, we obtain the full trajectory vocabulary by composing every path anchor with every velocity anchor:

$$\mathcal{T} = \{\tau_{i,j} \mid \tau_{i,j} = \mathcal{C}(p_i, v_j),\ p_i \in \mathcal{P},\ v_j \in \mathcal{V}\}. \tag{11}$$

This factorized construction enables combinatorial coverage of the trajectory space, allowing us to form a super-dense trajectory vocabulary from compact path and velocity sets. As a result, the size of the trajectory vocabulary scales as $|\mathcal{T}| = N_p \times N_v$, which is significantly larger than that of conventional monolithic vocabularies, while remaining computationally manageable under the proposed scoring framework.

3.4 Scalable Scoring

Scene Encoding. We first extract scene features from raw sensor observations o_t and encode the ego status e_t into status features:

$$\mathbf{F} = \Phi(o_t), \qquad \mathbf{E} = \Psi(e_t), \tag{12}$$

where $\Phi(\cdot)$ and $\Psi(\cdot)$ denote the scene encoder and the status encoder, respectively. Following the design philosophy of SparseDrive [28], the scene encoder directly extracts multi-view image features for subsequent interaction, without explicitly constructing BEV representations [9, 21].

Conditioned on the scene features $\mathbf{F}$ and status features $\mathbf{E}$, a straightforward approach to scoring-based planning treats each trajectory as a monolithic unit and directly scores all trajectory candidates. However, the computational complexity of such trajectory-level scoring grows linearly with the number of trajectory anchors. When scaling to super-dense vocabularies containing hundreds of thousands of trajectories, this approach becomes computationally prohibitive for both training and inference.

Coarse Factorized Scoring. To enable efficient scoring over the super-dense combinatorial trajectory vocabulary, we exploit the factorized structure and perform coarse-grained scoring independently at the path and velocity levels.

Path scoring. For each path anchor $p_i \in \mathcal{P}$, we first encode it into a path embedding using a lightweight MLP:

$$\mathbf{e}_i^p = E_p(p_i), \qquad \mathbf{e}_i^p \in \mathbb{R}^d. \tag{13}$$

We then incorporate scene context by interacting path embedding with the scene features $\mathbf{F}$ and the status features $\mathbf{E}$:

$$\widetilde{\mathbf{e}}_i^p = \Psi_p(\mathbf{e}_i^p + \mathbf{E}, \mathbf{F}), \tag{14}$$

where $\Psi_p(\cdot)$ denotes a path–scene interaction module. Since a path provides explicit spatial geometry, Ψ_p can be instantiated either as multi-head cross-attention [30] between path embeddings and scene features, or as deformable

aggregation [23] that samples scene features along the path geometry. Finally, a coarse path score is predicted directly from each contextualized path embedding:

$$s_i^p = f_p(\widetilde{\mathbf{e}}_i^p). \tag{15}$$

Velocity scoring. Similarly, for each velocity anchor $v_j \in \mathcal{V}$, we obtain a velocity embedding

$$\mathbf{e}_j^v = E_v(v_j), \qquad \mathbf{e}_j^v \in \mathbb{R}^d, \tag{16}$$

and incorporate scene context via:

$$\widetilde{\mathbf{e}}_j^v = \Psi_v(\mathbf{e}_j^v + \mathbf{E}, \mathbf{F}), \tag{17}$$

where $\Psi_v(\cdot)$ is implemented with cross-attention since velocity profiles do not contain explicit spatial geometry. A coarse velocity score is then predicted as

$$s_j^v = f_v(\widetilde{\mathbf{e}}_j^v). \tag{18}$$

Top-K selection. Although path and velocity are scored independently, they capture complementary aspects of high-level driving intent and are effective at filtering out a large number of low-quality trajectory candidates. For example, high-speed velocity profiles are unlikely to be valid in emergency or congested scenarios, while turning paths can be safely discarded in straight-lane driving scenario. By performing coarse scoring at the path and velocity levels, many implausible combinations can be eliminated before trajectory composition.

Specifically, based on the coarse path scores $\{s_i^p\}$ and velocity scores $\{s_j^v\}$, we select the top-K_p paths and top-K_v velocity profiles, respectively. The composition of these selected components yields a high-quality trajectory candidate set

$$\mathcal{T}_{\text{coarse}} = \{\mathcal{C}(p_i, v_j) \mid p_i \in \mathcal{P}_{K_p},\ v_j \in \mathcal{V}_{K_v}\}, \tag{19}$$

with a vocabulary size of $|\mathcal{T}_{\text{coarse}}| = K_p \times K_v$, which is several orders of magnitude smaller than the full combinatorial vocabulary.

Fine-Grained Trajectory Scoring. After coarse filtering, we perform fine-grained scoring on the composed trajectory candidates. Each trajectory anchor is obtained by composing a path anchor and a velocity anchor, i.e., $\tau_{i,j} = \mathcal{C}(p_i, v_j)$. Correspondingly, a trajectory feature can be naturally constructed by combining the associated path embedding and velocity embedding:

$$\mathbf{e}_{i,j}^{\tau} = \widetilde{\mathbf{e}}_i^p + \widetilde{\mathbf{e}}_j^v. \tag{20}$$

This fusion operation yields a trajectory feature that jointly encodes the spatial information from the path and the temporal information from the velocity profile, and can be directly used for trajectory-level scoring.

However, this formulation implicitly assumes independence between path and velocity, which does not always hold in real-world driving. For instance, a sharply turning path is typically incompatible with very high speeds. To capture such

spatiotemporal dependencies and enable joint reasoning over path, velocity, and scene context, we introduce a trajectory re-conditioning mechanism.

Specifically, the trajectory feature $\mathbf{e}^{\tau}_{i,j}$ further interacts with the scene features $\mathbf{F}$ to obtain a re-conditioned trajectory embedding:

$$\widetilde{\mathbf{e}}^{\tau}_{i,j} = \Psi_{\tau}(\mathbf{e}^{\tau}_{i,j}, \mathbf{F}), \tag{21}$$

where $\Psi_{\tau}(\cdot)$ denotes a trajectory–scene interaction module implemented as deformable aggregation. This re-conditioning step allows the model to perform fine-grained spatiotemporal reasoning conditioned on the scene. Finally, a fine-grained trajectory score is predicted as

$$s^{\tau}_{i,j} = f_{\tau}(\widetilde{\mathbf{e}}^{\tau}_{i,j}). \tag{22}$$

3.5 Training and Inference

Training Objectives. Our training objective follows a scoring-based formulation and supervises different stages of the scoring pipeline with appropriate targets. We apply distance-based soft classification losses for coarse factorized scoring and trajectory-level imitation learning, together with metric supervision distilled from a rule-based teacher system.

Coarse Factorized Scoring. For the path-level scoring, given a ground-truth geometric path $\hat{p}$, we compute the masked average squared distance between $\hat{p}$ and each path anchor $p_i \in \mathcal{P}$:

$$d^{p}_{i} = \frac{1}{|\mathcal{M}|} \sum_{k \in \mathcal{M}} \|p_i(k) - \hat{p}(k)\|_2^2 \,, \tag{23}$$

where $\mathcal{M}$ denotes the set of valid path points and $|\mathcal{M}|$ is the number of valid points. We then construct a soft target distribution and apply a cross-entropy loss:

$$\mathcal{L}_{\text{path}} = \text{CE}(s^{p},\ \text{Softmax}(-\lambda_p\, d^{p}))\,, \tag{24}$$

where $s^{p} \in \mathbb{R}^{N_p}$ denotes the predicted path scores and λ_p is a scaling factor.

Similarly, for velocity-level scoring, we compute the L_1 distance between the ground-truth velocity profile $\hat{v}$ and each velocity anchor $v_j \in \mathcal{V}$:

$$d^{v}_{j} = \sum_{t=1}^{T} |v_j(t) - \hat{v}(t)|\,, \tag{25}$$

and supervise velocity scoring with a soft classification loss:

$$\mathcal{L}_{\text{vel}} = \text{CE}(s^{v},\ \text{Softmax}(-\lambda_v\, d^{v}))\,, \tag{26}$$

where $s^{v} \in \mathbb{R}^{N_v}$ denotes the predicted velocity scores and λ_v is a scaling factor.

Fine-Grained Trajectory Scoring. After coarse filtering, we supervise fine-grained scoring on composed trajectory candidates. Given a ground-truth trajectory $\hat{\tau}$ and a trajectory anchor τ_k, we compute their squared L_2 distance:

$$d_k^\tau = \sum_{t=1}^{T} \|\tau_k(t) - \hat{\tau}(t)\|_2^2 . \tag{27}$$

Trajectory-level scores are trained using the following soft classification loss:

$$\mathcal{L}_{\text{traj}} = \text{CE}(s^\tau,\ \text{Softmax}(-\lambda_\tau\, d^\tau)) , \tag{28}$$

where s^τ denotes the predicted trajectory scores and λ_τ is a scaling factor.

Following Hydra-MDP [18], we further incorporate metric supervision at the trajectory level using a rule-based teacher system. The teacher evaluates each composed trajectory under a set of predefined driving metrics and produces metric sub-scores $\{\hat{s}^{(m)}\}$, reflecting safety, progress, comfort, and rule compliance. The scoring network directly predicts the corresponding sub-scores $\{s^{(m)}\}$ for each trajectory, and is trained using binary cross-entropy loss:

$$\mathcal{L}_{\text{metric}} = \sum_m \text{BCE}\Big(s^{(m)},\ \hat{s}^{(m)}\Big) . \tag{29}$$

Overall Objective. The final training objective is a weighted sum of all losses:

$$\mathcal{L} = \mathcal{L}_{\text{path}} + \mathcal{L}_{\text{vel}} + \mathcal{L}_{\text{traj}} + \alpha\, \mathcal{L}_{\text{metric}}, \tag{30}$$

where α controls the contribution of rule-based metric supervision.

Inference. At inference time, trajectory candidates are assigned final scores by aggregating the predicted sub-scores using the same weighting scheme as in the benchmark evaluation protocol. The planner then selects the trajectory with the highest aggregated score as the final driving decision.

4 Experiments

4.1 Dataset and Metrics

We conduct experiments on NAVSIM [3] and Bench2Drive [12]benchmarks. NAVSIM is split into `navtrain` (1192 training scenes) and `navtest` (136 evaluation scenes), with two evalution protocols, which is NAVSIM v1 and NAVSIM v2. NAVSIM v1 includes 5 metrics: no collision (NC), drivable area compliance (DAC), ego progress (EP), time-to-collision (TTC), and comfort (C). PDM score (PDMS) is computed by weighted aggregation of these metrics:

$$PDMS = NC \times DAC \times \tfrac{(5\times TTC+2\times C+5\times EP)}{12} \tag{31}$$

NAVSIM v2 introduces 4 extended metrics, including driving direction compliance (DDC), traffic light compliance (TL), lane keeping (LK) and extended

Table 2: Performance on NAVSIM v1 `navtest` leaderboard.

Method	Img. Backbone	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
VADv2 [1]	ResNet-34	97.2	89.1	91.6	**100**	76.0	80.9
UniAD [8]	ResNet-34	97.8	91.9	92.9	**100**	78.8	83.4
Transfuser [2]	ResNet-34	97.7	92.8	92.8	**100**	79.2	84.0
PARA-Drive [31]	ResNet-34	97.9	92.4	93.0	99.8	79.3	84.0
DRAMA [36]	ResNet-34	98.0	93.1	94.8	**100**	80.1	85.5
GoalFlow [34]	ResNet-34	98.3	93.8	94.3	**100**	79.8	85.7
Hydra-MDP [18]	ResNet-34	98.3	96.0	94.6	**100**	78.7	86.5
Hydra-MDP++ [16]	ResNet-34	97.6	96.0	93.1	**100**	80.4	86.6
ARTEMIS [5]	ResNet-34	98.3	95.1	94.3	**100**	81.4	87.0
FUMP [24]	ResNet-34	98.1	96.2	94.2	**100**	82.0	87.8
DiffusionDrive [22]	ResNet-34	98.2	96.2	94.7	**100**	82.2	88.1
WoTE [17]	ResNet-34	98.5	96.8	94.9	99.9	81.9	88.3
DIVER [26]	ResNet-34	98.5	96.5	94.9	**100**	82.6	88.3
DriveSuprim [35]	ResNet-34	97.8	97.3	93.6	**100**	86.7	89.9
DiffusionDriveV2 [39]	ResNet-34	98.3	97.9	94.8	99.9	87.5	91.2
ipad [6]	ResNet-34	**98.6**	98.3	94.9	**100**	88.0	91.7
Hydra-MDP [18]	V2-99	98.4	97.8	93.9	**100**	86.5	90.3
GoalFlow [34]	V2-99	98.4	98.3	94.6	**100**	85.0	90.3
SparseDriveV2	ResNet-34	98.5	**98.4**	**95.0**	99.9	**88.6**	**92.0**

comfort (EC). The Extended PDM Score (EPDMS) is aggregated by:

$$EPDMS = NC \times DAC \times DDC \times TL \times \frac{(5 \times EP + 5 \times TTC + 2 \times LK + 2 \times C + 2 \times EC)}{16} \tag{32}$$

Bench2Drive is a closed-loop benchmark based on CARLA simulator [4], with 220 test routes across 44 interactive scenarios. The official metrics include Driving Score, Success Rate, Efficiency and Comfortness.

Table 3: Performance on NAVSIM v2 `navtest` leaderboard. EPDMS* denotes results computed using the original NAVSIM v2 evaluation code, in which human-behavior filtering was not applied when calculating EPDMS. This version is reported for fair comparison with prior works. EPDMS denotes results computed using the updated official implementation after the issue was resolved.

Method	Img. Backbone	NC ↑	DAC ↑	DDC ↑	TL ↑	EP ↑	TTC ↑	LK ↑	HC ↑	EC ↑	EPDMS* ↑	EPDMS ↑
Ego Status MLP	ResNet-34	93.1	77.9	92.7	99.6	86.0	91.5	89.4	**98.3**	85.4	64.0	–
Transfuser [2]	ResNet-34	96.9	89.9	97.8	99.7	87.1	95.4	92.7	**98.3**	87.2	76.7	–
Hydra-MDP++ [16]	ResNet-34	97.2	97.5	99.4	99.6	83.1	96.5	94.4	98.2	70.9	81.4	–
DriveSuprim [35]	ResNet-34	97.5	96.5	99.4	99.6	88.4	96.6	95.5	**98.3**	77.0	83.1	–
ARTEMIS [5]	ResNet-34	98.3	95.1	98.6	99.8	81.5	97.4	96.5	**98.3**	**98.3**	83.1	–
DiffusionDriveV2	ResNet-34	97.7	96.6	99.2	99.8	88.9	97.2	96.0	97.8	91.0	85.5	87.5
Hydra-MDP++ [16]	V2-99	**98.4**	98.0	99.4	99.8	87.5	**97.7**	95.3	**98.3**	77.4	85.1	–
DriveSuprim [35]	V2-99	97.8	97.9	99.5	**99.9**	90.6	97.1	96.6	**98.3**	77.9	86.0	–
SparseDriveV2	ResNet-34	98.1	**98.1**	**99.6**	99.8	**91.1**	97.3	**96.9**	98.2	78.4	**86.7**	**90.1**

Table 4: Closed-loop planning performance on Bench2Drive. * denotes expert feature distillation.

Method	Driving Score ↑	Success Rate (%) ↑	Driving Efficiency ↑	Comfort ↑
UniAD [8]	45.81	16.36	129.21	43.58
VAD [14]	42.35	15.00	157.94	46.01
SparseDrive [28]	44.54	16.71	170.21	**48.63**
GenAD [38]	44.81	15.90	-	-
DiFSD [27]	52.02	21.00	178.30	-
DriveTransformer [13]	63.46	35.01	100.64	20.78
Hydra-NeXt [19]	73.86	50.00	197.76	20.68
SimLingo [25]	86.02	67.27	**259.23**	33.67
HiP-AD [29]	86.77	69.09	203.12	19.36
TCP-traj* [32]	59.90	30.00	76.54	18.08
ThinkTwice* [11]	62.44	31.23	69.33	16.22
DriveAdapter* [10]	64.22	33.08	70.22	16.01
SparseDriveV2	**89.15**	**70.00**	199.84	18.32

4.2 Implementation Details

We present the implementation details for NAVSIM in this section, while those for Bench2Drive are deferred to the appendix due to space limitations.

For NAVSIM v1, we construct a factorized trajectory vocabulary consisting of 1024 path anchors and 256 velocity anchors. Path anchors are sampled with a spatial interval of 1 m and a maximum spatial horizon of 50 m. Velocity anchors are defined with a temporal interval of 0.5 s and a temporal horizon of 4 s. The combinatorial composition results in $1024 \times 256 = 262{,}144$ trajectory anchors, which is 32× denser than the commonly used 8192 anchors in prior scoring-based methods [18, 35].

To enable efficient inference over this super-dense vocabulary, we adopt a progressive filtering strategy with two decoder layers. The first layer retains the top 128 path anchors and 64 velocity anchors. The second layer further refines the candidates to 20 paths and 20 velocities, resulting in 400 trajectory hypotheses for fine-grained scoring. Metric supervision is applied only to these 400 filtered trajectories. For NAVSIM v2, the only difference is that the number of velocity anchors in the second layer is reduced to 10 in order to accelerate metric ground-truth computation.

For scene feature extraction, we use a ResNet-34 [7] backbone. Images from cameras l_0, f, and r_0 are used as input, with a resolution of 256×512. We train all models on the `navtrain` split using 8 NVIDIA L20 GPUs, with a total batch size of 128 for 10 epochs. The learning rate is set to 1×10^{-4} and the weight decay is set to 0.

4.3 Main Results

Results on NAVSIM v1. As shown in Tab. 2, SparseDriveV2 achieves a PDMS of 92.0, outperforming both prior scoring-based planners and dynamic trajectory generation methods. Notably, with only a lightweight ResNet-34 backbone (21.8M parameters), SparseDriveV2 still surpasses GoalFlow and Hydra-MDP

Table 5: Multi-Ability results on Bench2Drive.* denotes expert feature distillation.

Method	Ability (%) ↑					
	Merging	Overtaking	Emergency Brake	Give Way	Traffic Sign	**Mean**
AD-MLP [37]	0.00	0.00	0.00	0.00	4.35	0.87
UniAD [8]	14.10	17.78	21.67	10.00	14.21	15.55
VAD [14]	8.11	24.44	18.64	20.00	19.15	18.07
DriveTransformer [13]	17.57	35.00	48.36	40.00	52.10	38.60
Hydra-NeXt [19]	40.00	64.44	61.67	**50.00**	50.00	53.22
HiP-AD [29]	50.00	**84.44**	**83.33**	40.00	**72.10**	65.98
TCP-traj* [32]	12.50	22.73	52.72	40.00	46.63	34.92
ThinkTwice* [11]	13.72	22.93	52.99	**50.00**	47.78	37.48
DriveAdapter* [10]	14.55	22.61	54.04	**50.00**	50.45	38.33
SparseDriveV2	**66.25**	75.55	75.00	**50.00**	71.57	**67.67**

equipped with the significantly larger V2-99 backbone [15] (96.9M parameters), demonstrating the effectiveness of the proposed scalable vocabulary and scoring framework.

Results on NAVSIM v2. Tab. 3 reports the results on the NAVSIM v2 `navtest` split. Under the original evaluation protocol, SparseDriveV2 achieves the best performance among methods using the same backbone and also outperforms strong scoring-based approaches such as Hydra-MDP++ and DriveSuprim that adopt larger backbones. In particular, the substantial improvement in the EP metric indicates the advantage of the proposed scalable vocabulary in providing broader coverage of the action space. We additionally report the corrected EPDMS metric to encourage the community to adopt the updated evaluation protocol. Under this metric, SparseDriveV2 achieves a EPDMS of 90.1, surpassing DiffusionDriveV2 by 2.6 EPDMS.

Results on Bench2Drive. The closed-loop planning performance on Bench2Drive is reported in Tab. 4 and Tab. 5. SparseDriveV2 achieves a Driving Score of 89.15 and a Success Rate of 70.00%, together with a multi-ability score of 67.67. These results consistently outperform prior approaches, demonstrating the strong generalization ability of SparseDriveV2 in complex closed-loop driving scenarios.

4.4 Ablation Studies

Ablation for Scalable Vocabulary. We study the effect of vocabulary scaling by varying the numbers of path anchors (N_p) and velocity anchors (N_v), as shown in Tab. 6 (left). Increasing the vocabulary size consistently improves planning performance, indicating that a denser vocabulary provides broader action-space coverage and enables the scorer to select higher-quality trajectories.

Ablation for Scalable Scoring. We further evaluate the proposed scalable scoring design in Tab. 6 (right), including the path–scene interaction mechanism and the trajectory re-conditioning module. Replacing standard multi-head cross-attention with deformable aggregation consistently improves performance, suggesting that it captures more informative spatial cues along path anchors. Moreover, trajectory re-conditioning improves results under both attention types,

Table 6: Ablation studies on NAVSIM v2. **Left:** Effect of scaling the factorized vocabulary size by varying the number of path anchors (N_p) and velocity anchors (N_v). **Right:** Effect of the scalable scoring design, including path–scene interaction type and trajectory re-conditioning. MHA denotes multi-head cross-attention and DFA denotes deformable aggregation.

Scalable Vocabulary			**Scalable Scoring**		
N_p	N_v	EPDMS ↑	Path Attn	Re-cond.	EPDMS ↑
512	128	88.7	MHA	✗	87.7
512	256	89.2	MHA	✓	89.9
1024	128	89.5	DFA	✗	89.9
1024	256	**90.1**	DFA	✓	**90.1**

highlighting the importance of modeling the spatiotemporal dependency of composed trajectories.

5 Conclusion

In this work, we propose SparseDriveV2, which introduces a scalable trajectory vocabulary representation and a scalable scoring strategy for reasoning over super-dense candidate trajectories. Extensive experiments demonstrate that SparseDriveV2 achieves SOTA performance among both scoring-based and dynamic generation methods, suggesting that with sufficiently dense vocabularies and efficient scoring mechanisms, scoring-based planning can serve as a simple yet powerful paradigm for end-to-end autonomous driving.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant 52572497 and Grant 52221005.

References

1. 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)
2. Chitta, K., Prakash, A., Jaeger, B., Yu, Z., Renz, K., Geiger, A.: Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. IEEE transactions on pattern analysis and machine intelligence **45**(11), 12878–12895 (2022)
3. Dauner, D., Hallgarten, M., Li, T., Weng, X., Huang, Z., Yang, Z., Li, H., Gilitschenski, I., Ivanovic, B., Pavone, M., et al.: Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. Advances in Neural Information Processing Systems **37**, 28706–28719 (2024)
4. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: Conference on robot learning. pp. 1–16. PMLR (2017)

5. Feng, R., Xi, N., Chu, D., Wang, R., Deng, Z., Wang, A., Lu, L., Wang, J., Huang, Y.: Artemis: Autoregressive end-to-end trajectory planning with mixture of experts for autonomous driving. IEEE Robotics and Automation Letters **11**(1), 226–233 (2025)
6. Guo, K., Liu, H., Wu, X., Pan, J., Lv, C.: ipad: Iterative proposal-centric end-to-end autonomous driving. arXiv preprint arXiv:2505.15111 (2025)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
8. 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: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17853–17862 (2023)
9. Huang, J., Huang, G., Zhu, Z., Ye, Y., Du, D.: Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. arXiv preprint arXiv:2112.11790 (2021)
10. 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: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7953–7963 (2023)
11. Jia, X., Wu, P., Chen, L., Xie, J., He, C., Yan, J., Li, H.: Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21983–21994 (2023)
12. Jia, X., Yang, Z., Li, Q., Zhang, Z., Yan, J.: Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. Advances in Neural Information Processing Systems **37**, 819–844 (2024)
13. Jia, X., You, J., Zhang, Z., Yan, J.: Drivetransformer: Unified transformer for scalable end-to-end autonomous driving. arXiv preprint arXiv:2503.07656 (2025)
14. 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: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8340–8350 (2023)
15. Lee, Y., Hwang, J.w., Lee, S., Bae, Y., Park, J.: An energy and gpu-computation efficient backbone network for real-time object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. pp. 0–0 (2019)
16. Li, K., Li, Z., Lan, S., Xie, Y., Zhang, Z., Liu, J., Wu, Z., Yu, Z., Alvarez, J.M.: Hydra-mdp++: Advancing end-to-end driving via expert-guided hydra-distillation. arXiv preprint arXiv:2503.12820 (2025)
17. 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: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 27137–27146 (2025)
18. 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)
19. Li, Z., Wang, S., Lan, S., Yu, Z., Wu, Z., Alvarez, J.M.: Hydra-next: Robust closed-loop driving with open-loop training. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 27305–27314 (2025)

20. 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)
21. Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Yu, Q., Dai, J.: Bevformer: learning bird's-eye-view representation from lidar-camera via spatiotemporal transformers. IEEE Transactions on Pattern Analysis and Machine Intelligence **47**(3), 2020–2036 (2024)
22. Liao, B., Chen, S., Yin, H., Jiang, B., Wang, C., Yan, S., Zhang, X., Li, X., Zhang, Y., Zhang, Q., et al.: Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 12037–12047 (2025)
23. Lin, X., Lin, T., Pei, Z., Huang, L., Su, Z.: Sparse4d: Multi-view 3d object detection with sparse spatial-temporal fusion. arXiv preprint arXiv:2211.10581 (2022)
24. Liu, L., Jia, C., Song, Z., Pan, H., Liao, B., Sun, W., Zhang, Y., Yang, L., Luo, Y.: Fully unified motion planning for end-to-end autonomous driving. arXiv preprint arXiv:2504.12667 (2025)
25. Renz, K., Chen, L., Arani, E., Sinavski, O.: Simlingo: Vision-only closed-loop autonomous driving with language-action alignment. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 11993–12003 (2025)
26. Song, Z., Liu, L., Pan, H., Liao, B., Guo, M., Yang, L., Zhang, Y., Xu, S., Jia, C., Luo, Y.: Diver: Reinforced diffusion breaks imitation bottlenecks in end-to-end autonomous driving. arXiv preprint arXiv:2507.04049 (2025)
27. Su, H., Wu, W., Yan, J.: Difsd: Ego-centric fully sparse paradigm with uncertainty denoising and iterative refinement for efficient end-to-end self-driving. arXiv preprint arXiv:2409.09777 (2024)
28. Sun, W., Lin, X., Shi, Y., Zhang, C., Wu, H., Zheng, S.: Sparsedrive: End-to-end autonomous driving via sparse scene representation. In: 2025 IEEE International Conference on Robotics and Automation (ICRA). pp. 8795–8801. IEEE (2025)
29. Tang, Y., Xu, Z., Meng, Z., Cheng, E.: Hip-ad: Hierarchical and multi-granularity planning with deformable attention for autonomous driving in a single decoder. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 25605–25615 (2025)
30. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
31. Weng, X., Ivanovic, B., Wang, Y., Wang, Y., Pavone, M.: Para-drive: Parallelized architecture for real-time autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15449–15458 (2024)
32. Wu, P., Jia, X., Chen, L., Yan, J., Li, H., Qiao, Y.: Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. Advances in Neural Information Processing Systems **35**, 6119–6132 (2022)
33. Wu, Y., Zhang, H., He, F., Wu, R., Shan, Y., Qiu, C., Gao, L., Ke, W., Zhang, T.: Alignndrive: Aligned lateral-longitudinal planning for end-to-end autonomous driving. arXiv preprint arXiv:2601.01762 (2026)
34. 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: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 1602–1611 (2025)
35. 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)

36. 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)
37. Zhai, J.T., Feng, Z., Du, J., Mao, Y., Liu, J.J., Tan, Z., Zhang, Y., Ye, X., Wang, J.: Rethinking the open-loop evaluation of end-to-end autonomous driving in nuscenes. arXiv preprint arXiv:2305.10430 (2023)
38. Zheng, W., Song, R., Guo, X., Zhang, C., Chen, L.: Genad: Generative end-to-end autonomous driving. In: European Conference on Computer Vision. pp. 87–104. Springer (2024)
39. 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)