EVOAD: SELF-EVOLVING AUTONOMOUS DRIVING VIA LATENT WORLD-MODEL GUIDED POST-HOC REFLECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

End-to-end autonomous driving systems have achieved rapid progress by learning unified policies that map sensor observations directly to driving trajectories. However, once deployed, these systems typically remain frozen: they may repeat similar decisions in similar scenes, and failures can only be addressed through costly data collection and retraining. This sharply contrasts with human drivers, who improve lifelong through experience, memory, and reflection. In this work, we ask whether autonomous driving systems can achieve similar post-deployment self-evolution to human drivers. We propose EvoAD, a self-evolving autonomous driving framework via latent world-model guided post-hoc reflection. EvoAD stores driving experiences in a bird's-eye-view feature memory pool, retrieves scene-relevant precedents according to geometric and semantic similarity, and uses a latent world model to evaluate the future consequences of candidate trajectories in a compact planning-oriented space. Instead of treating memory as external textual contexts or using world models only for pre-action simulation, EvoAD transforms executed driving outcomes into reusable improvement signals for future decisions. This enables the system to diagnose how prior trajectories could have been improved and to refine behavior in similar future scenarios without retraining the base driving policy. Experiments on NAVSIM v1 and v2 show consistent improvements across reflection rounds and state-of-the-art performance, demonstrating the potential of world models as reflective mechanisms for post-deployment self-evolution in autonomous driving.

1 INTRODUCTION

End-to-end (E2E) autonomous driving (AD) has advanced rapidly as a unified data-driven paradigm, where a single neural policy is trained on large-scale driving data to map raw sensor observations directly to driving trajectories (Hu et al., 2023; Jiang et al., 2023). Yet after training, these systems are typically deployed as frozen models: given the same scene, they are likely to repeat the same decision, and any failure observed after deployment can only be corrected through another costly cycle of data collection, labeling, and retraining (Chen et al., 2024; Caesar et al., 2021). In stark contrast, human drivers exhibit a fundamentally different form of competence. Even after obtaining a license, they continue to improve lifelong: they remember previously encountered situations, mentally revisit decisions that felt unsafe or suboptimal, and adjust their future behavior when similar scenarios arise (Groeger, 2013; Fuller, 2005). This contrast exposes a central limitation of current E2E AD systems: driving competence is never achieved once and for all; it continually evolves through experience, memory, and reflection.

A growing line of work has sought to recover this missing adaptability by moving self-improvement into the test-time loop, including online or test-time fine-tuning (Wang et al., 2020; Hanselmann et al., 2022), knowledge- and retrieval-augmented driving (Yuan et al., 2024b; Wen et al., 2023), and score-based candidate re-ranking (Li et al., 2024b; 2025c). However, retrieval and memory in these methods are typically used as external context, such as textual knowledge, high-level experience, or demonstration priors. They rarely provide feedback grounded in the actual consequences of the trajectory just produced by the planner. More recently, world models promise to bridge this gap by creating an imaginative space for action evaluation, where each candidate trajectory can be rolled

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

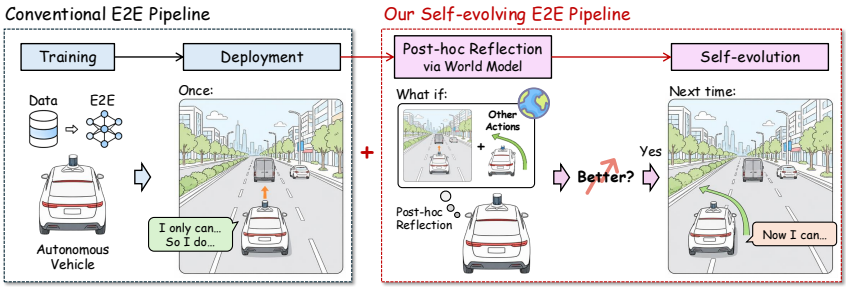


Figure 1: **From Conventional to Self-Evolving E2E AD**. Conventional E2E pipelines typically execute fixed policies after deployment and thus tend to repeat similar decisions in similar scenarios. We introduce a world-model guided post-hoc reflection mechanism that enables the system to imagine alternative actions, identify potentially better outcomes, and update future behavior.

forward into its possible visual or latent consequences (Wang et al., 2024a; Gao et al., 2023; Lu et al., 2024). Consequently, many AD approaches have incorporated world models as forward simulators, using them to imagine future outcomes before selecting an action, such as World4Drive (Zheng et al., 2025), and Drive-WM (Wang et al., 2024b). In these systems, world models behave like a forward-looking advisor: they imagine possible futures to guide action selection before the vehicle acts. This dominant use of world models, however, obscures a broader question about their role in AD: can their imaginative capacity support human-like self-evolution through post-hoc reflection?

In this study, we propose **EvoAD**, a self-evolving AD framework based on latent world-model guided post-hoc reflection. As shown in Fig. 1, rather than treating a trained E2E model as a fixed policy, EvoAD converts past driving experiences into reusable improvement signals for future decision-making. First, we introduce bird's-eye-view (BEV) feature-based memory to store reflected driving experiences, including scene representations, trajectories, future latents, and evaluation feedback. Compared with text-only memory or high-level action descriptions, BEV-level memory better preserves the geometric layout and spatial relations of the driving scene, making retrieval more aligned with the structure of AD decisions. Second, we use a latent world model to imagine the future consequences of candidate trajectories in a compact planning-oriented latent space, which enables safer or more efficient trajectory evaluation without expensive pixel-level future reconstruction. Third, EvoAD performs outcome-aware post-hoc reflection based on trajectory evaluation feedback, so the reflection is not merely past-action evidence, but a targeted improvement based on the previous weakness. In doing so, EvoAD enables a trained base model to accumulate experience, retrieve relevant prior reflections, and refine future behavior without retraining.

In summary, our contributions are threefold: 1) We propose a self-evolving AD framework based on post-hoc reflection, enabling AD systems to improve after deployment without retraining the base model; 2) We redefine the role of the world model as a post-hoc reflective evaluator, which explores better alternatives in a latent imaginative space; and 3) We introduce BEV-feature-based memory and retrieval, which stores scenario-level driving experiences and retrieves relevant ones to support future decision refinement. Experiments on NAVSIM v1 and v2 show that EvoAD improves WoTE from 88.3 to 93.4 PDMS and from 84.2 to 91.5 EPDMS, respectively, achieving state-of-the-art performance on both benchmarks.

2 Related Work

2.1 Post-hoc Reflection and Retrieval Memory for Autonomous Driving

Recent vision-language and multimodal driving models increasingly use reasoning, explanation, retrieval, and memory to support driving decisions. Existing studies have explored scene-graph-based explanations for auditable decision making (Belmecheri et al., 2024), retrieval-augmented expert demonstrations for control and explanation (Yuan et al., 2024b), reward-aligned reasoning for trajectory or meta-action prediction (Li et al., 2026), and diagnostic analyses of the reasoning-planning disconnect in driving vision-language models. In parallel, retrieval and memory have been introduced to incorporate traffic regulations, expert trajectories, commonsense knowledge, and past experience into driving systems (Cai et al., 2026; Fu et al., 2025; Ye et al., 2025). Yet prior

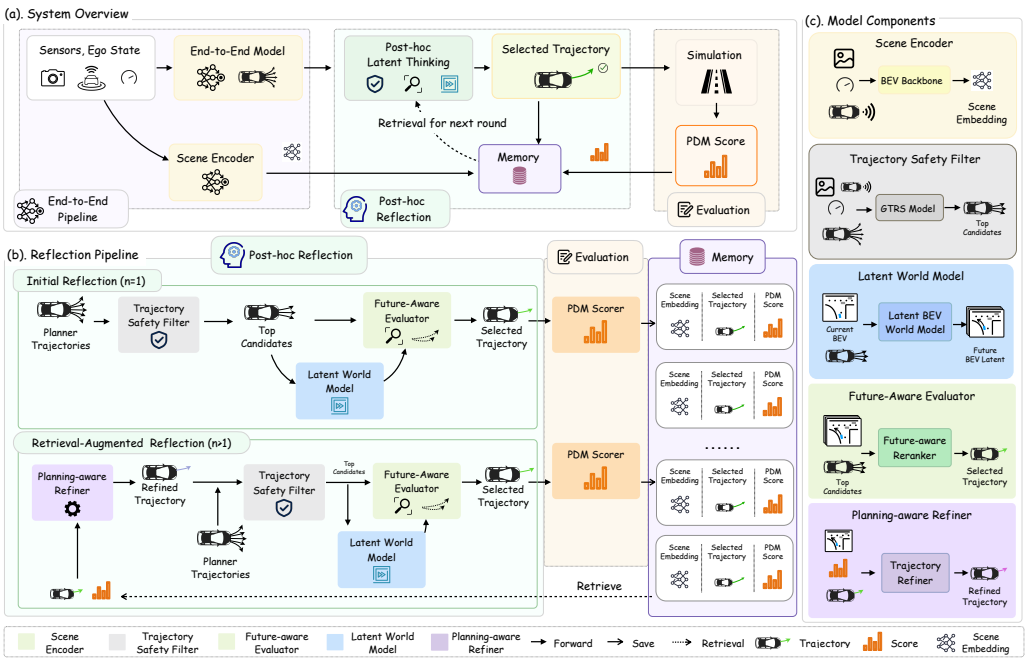


Figure 2: **Overview of EvoAD.** (a) **System Overview**: an E2E planner augmented with post-hoc latent reflection, evaluation, and memory. (b) **Reflection Pipeline**: safe, future-aware trajectories are selected and then refined with retrieved experiences. (c) **Model Components**: scene encoder, safety filter (SF), latent world model with future-aware evaluator (FAE), and planning-aware refiner (PAR).

methods mainly use post-hoc reasoning for explanation or retrieval as auxiliary knowledge, external demonstrations, or planner priors, lacking a closed-loop mechanism that reflects on the model's own driving outcomes and turns them into reusable improvement signals. EvoAD fills this gap by using latent world-model feedback to evaluate its own trajectory consequences, store reflected experiences in BEV-feature memory, and retrieve relevant experiences to refine future decisions.

2.2 Latent World Modeling in Autonomous Driving

World models provide a natural substrate for deliberative AD by simulating possible future scene evolutions before action execution. Early driving world models such as Drive-WM couple multiview future visual forecasting with planning by rolling out candidate futures and selecting actions according to predicted outcomes (Wang et al., 2024b). Recent methods increasingly move from pixel-level generation to planning-oriented latent representations: LAW predicts future scene features as self-supervision for E2E driving (Li et al., 2024a); World4Drive learns intention-aware physical latent dynamics for multimodal trajectory selection and closed-loop performance (Zheng et al., 2025); WorldRFT fine-tunes latent world models with reinforcement signals to better align them with planning objectives (Yang et al., 2026); and DriveWorld-VLA unifies vision-language-action planning and latent-space world modeling within a shared latent state (Liu et al., 2026). These studies establish latent world models as planning-aware future simulators. EvoAD builds on this trend but uses the latent world model differently: instead of merely predicting future states or selecting actions, it uses latent future feedback for post-hoc reflection and iterative self-evolution of the driving policy.

3 Methodology

3.1 Problem Formulation

We formulate post-hoc trajectory improvement for E2E AD as an iterative process followed by E2E model inference and evaluation results. At each timestamp t, the system receives a driving observation $\mathcal{O}_t = \{\mathcal{I}_t, \mathbf{s}_t^{\text{raw}}, \mathcal{G}_t, c_t\}$, where $\mathcal{I}_t$ denotes sensor observations, $\mathbf{s}_t^{\text{raw}}$ is the raw ego state, $\mathcal{G}_t$ represents map context, and c_t is the navigation command. Given $\mathcal{O}_t$, a base planner π generates

a set of candidate future trajectories

$$\mathcal{Y}_t^{(1)} = \pi(\mathcal{O}_t) = \{\mathbf{y}_{t,k}^{(1)}\}_{k=1}^{K}. \tag{1}$$

Instead of replacing the base planner, EvoAD aims to improve its output through a multi-round post-hoc reflection process,

$$\mathbf{y}_t^\star = \mathcal{F}(\mathcal{O}_t, \mathcal{Y}_t^{(1)}, \mathcal{M}), \tag{2}$$

where $\mathcal{F}$ denotes the proposed reflection procedure and $\mathcal{M}$ is an episodic memory that stores previous scene-level planning experiences.

As shown in Fig. 2, each reflection round chains a reusable SF, an FAE backed by a latent world model, and a post-hoc feedback signal on the executed trajectory; from the second round onward, the PAR additionally consumes memory-retrieved experiences to correct the previously selected trajectory before re-evaluation. The whole loop can be abstracted as

$$\mathbf{y}_t^{(n)}, r_t^{(n)}, \mathcal{M} = \mathcal{R}\Big(\mathcal{O}_t,\, \mathcal{Y}_t^{(n)},\, \mathcal{M}\Big), \qquad n = 1, \ldots, N, \tag{3}$$

where $\mathcal{R}$ denotes one reflection step, $\mathbf{y}_t^{(n)}$ the selected trajectory at round n, and $r_t^{(n)}$ its post-hoc feedback; $\mathbf{y}_t^\star = \mathbf{y}_t^{(N)}$ is returned after N rounds. Here, $r_t^{(n)}$ is observed only after execution and is never used for selection. It can be derived from real-world feedback signals, such as takeovers, jerk, or lane deviation; in our experiments, we use NAVSIM's PDMS, a composite metric capturing safety, progress, and comfort.

3.2 Future-Latent Guided Post-hoc Reflection

A trajectory that appears feasible under the current observation may still lead to unsafe or inefficient future interactions. EvoAD therefore augments post-hoc trajectory evaluation with candidate-conditioned future latents from a latent world model. At reflection round n, the score-based SF first prunes evidently infeasible candidates from $\mathcal{Y}_t^{(n)}$, resulting in $\tilde{\mathcal{Y}}_t^{(n)}$. For each remaining candidate $\mathbf{y}_{t,k}^{(n)}$, we obtain a future latent representation

$$\hat{\mathbf{z}}_{t,k}^{(n)} = \textsc{LatentWM}\left(\mathcal{O}_t, \mathbf{y}_{t,k}^{(n)}\right), \tag{4}$$

which summarizes the predicted future scene evolution conditioned on executing that candidate trajectory. EvoAD does not directly decode this latent representation or use the world model as a planner. Instead, the future latent is used as a compact trajectory-conditioned prior to provide the future trend for downstream evaluation. We form a future-latent augmented candidate representation:

$$\mathbf{h}_{t,k}^{(n)} = \mathrm{Concat}\left(\mathbf{s}_t, \mathbf{y}_{t,k}^{(n)}, \hat{\mathbf{z}}_{t,k}^{(n)}\right), \tag{5}$$

where $\mathbf{h}_{t,k}^{(n)}$ is the concatenated feature of the ego status, the candidate trajectory, and its predicted future latent. This representation allows the evaluator to compare candidates not only by their geometric feasibility, but also by their potential future consequences from latent features. The same future-latent interface is used for both initial planner candidates and memory-refined proposals, providing a unified bridge between the latent world model and iterative post-hoc reflection.

3.3 Future-Aware Trajectory Evaluator

Since score-based methods like GTRS (Li et al., 2025c) only evaluate the performance of trajectories under current scene feature without considering the potential future latent signal, we introduce the FAE to rerank the candidate proposals. In each reflection round, the FAE scores the safety-filtered candidates and selects the one EvoAD will execute. Building on $\mathbf{h}_{t,k}^{(n)}$ from Sec. 3.2, it (i) encodes each candidate into a compact feature, (ii) jointly grounds the compact feature in the current scene and compares it against the other candidates, and (iii) decodes a ranking signal by dense score supervision of each candidate.

Intra-candidate encoding. For each candidate trajectory $\mathbf{y}_{t,k}^{(n)}$, the evaluator receives its future-latent augmented representation $\mathbf{h}_{t,k}^{(n)}$, then encoded into a compact feature through a self-attention module, which summarizes the future evidence of a single proposal:

$$\mathbf{c}_{t,k}^{(n)} = \mathrm{SelfAttn}\left(\mathbf{h}_{t,k}^{(n)}\right). \tag{6}$$

Current context fusion. To further relate this future-aware proposal feature $\mathbf{c}_{t,k}^{(n)}$ to the current driving scene, we reuse the backbone scene feature $\mathbf{F}_t \in \mathbb{R}^{L\times d}$ as a shared scene context and update each candidate feature through cross-attention:

$$\mathbf{c}_{t,k}^{\mathrm{ctx},(n)} = \mathrm{CrossAttn}\left(\mathbf{c}_{t,k}^{(n)}, \mathbf{F}_t\right) + \alpha \mathbf{c}_{t,k}^{(n)}. \tag{7}$$

This operation allows each future-aware proposal feature to query the current scene evidence, so that the model can incorporate predicted future evolution while preserving an explicit understanding of the present traffic context. We apply this query update with a residual connection and a learnable gate α, so that the evaluator can adaptively balance the original future-aware candidate representation and the current-scene contextual feature.

Set-wise candidate interaction. Considering the fact that proposal quality is relative within the same scene, the evaluator should compare candidates under shared context rather than score each trajectory independently. After context fusion, all candidates are jointly processed by set-wise self-attention to interact with each other for further ranking:

$$[\hat{\mathbf{c}}_{t,1}^{(n)}, \ldots, \hat{\mathbf{c}}_{t,K}^{(n)}] = \mathrm{SelfAttn}\left([\mathbf{c}_{t,1}^{\mathrm{ctx},(n)}, \ldots, \mathbf{c}_{t,K}^{\mathrm{ctx},(n)}]\right). \tag{8}$$

The set-wise interaction therefore helps capture redundancy, relative safety, progress trade-offs, and fine-grained differences among similar proposals.

Training loss. Given the candidate feature $\hat{\mathbf{c}}_{t,k}^{(n)}$, the evaluator uses a ranking head to decode the ranking signal $\hat{s}_{t,k}^{(n)}$ and auxiliary score heads to predict trajectory score and sub-items $\hat{r}_{t,k}^{(n)}$, $\hat{\mathbf{u}}_{t,k}^{(n)}$:

$$\hat{s}_{t,k}^{(n)} = h^{\mathrm{rank}}(\hat{\mathbf{c}}_{t,k}^{(n)}), \qquad \hat{\mathbf{u}}_{t,k}^{(n)} = h^{\mathrm{sub}}(\hat{\mathbf{c}}_{t,k}^{(n)}), \qquad \hat{r}_{t,k}^{(n)} = h^{\mathrm{score}}(\hat{\mathbf{c}}_{t,k}^{(n)}). \tag{9}$$

The ranking score $\hat{s}_{t,k}^{(n)}$ is used to select the final proposal. Then, the evaluator is trained jointly with

$$\mathcal{L}_{\mathrm{eval}} = \lambda_{\mathrm{list}}\,\mathcal{L}_{\mathrm{list}} + \lambda_{\mathrm{sub}}\,\mathcal{L}_{\mathrm{sub}} + \lambda_{\mathrm{score}}\,\mathcal{L}_{\mathrm{score}}, \tag{10}$$

where $\mathcal{L}_{\mathrm{list}}$ optimizes the within-scene proposal ordering on $\hat{s}_{t,k}^{(n)}$ by the maximum-likelihood pairwise loss, $\mathcal{L}_{\mathrm{score}}$ and $\mathcal{L}_{\mathrm{sub}}$ denote the loss of predicted scores (Guo et al., 2025).

3.4 Retrieval-Guided Planning-aware Refiner

From Round 2 onward, EvoAD looks up similar past experiences in an episodic memory and uses them to correct the previously selected trajectory. With the last-round decision and its performance retrieved from the memory, the refiner aims to consume the warm-start trajectory together with its real post-hoc feedback and per-trajectory future latent, and then emit a small, feedback-conditioned correction rather than re-planning. By targeting the failure mode observed in the previous round (dangerous interaction, low progress, out-of-drivable-area, etc.), the refinement is specific for trajectory deficiencies rather than searching for a higher score.

Episodic memory. EvoAD maintains an episodic memory $\mathcal{M}$ that accumulates evaluated planning experiences. Each entry stores the round's selected candidate as

$$m_{t,k}^{(n)} = \left(\mathbf{e}_t,\, \mathbf{y}_{t,k}^{(n)},\, r_{t,k}^{(n)},\, \mathbf{u}_{t,k}^{(n)},\, \hat{\mathbf{z}}_{t,k}^{(n)}\right), \tag{11}$$

where $\mathbf{e}_t$ is the scene embedding from the scene encoder, $\mathbf{y}_{t,k}^{(n)}$ the selected trajectory, $(r_{t,k}^{(n)}, \mathbf{u}_{t,k}^{(n)})$ its measured feedback total and sub-scores, and $\hat{\mathbf{z}}_{t,k}^{(n)}$ its per-trajectory future latent. Each scene keeps at

most one record, overwritten only on improvement, so the memory does not grow with the number of rounds. Nor does it need to grow with the number of scenes: once the pool is sufficiently rich it can be frozen, or even built entirely from another domain (Sec. 4.2). Given $\mathbf{e}_t$, the related scenes are retrieved by cosine similarity,

$$\mathcal{N}_t^{(n)} = \mathrm{TopK}_{m_{t',k'}^{(n')} \in \mathcal{M}} \, \mathrm{sim}\big(\mathbf{e}_t, \mathbf{e}_{t'}\big), \tag{12}$$

yielding scene-conditioned priors over previously observed planning patterns.

Planning-aware refinement model. Let $\mathbf{y}_t^w$ denote the last-round selected trajectory retrieved from the memory $\mathcal{N}_t^{(n)}$ to be refined, and k_w is the search index. Our planning-aware refinement model takes $\mathbf{y}_t^w$, the per-trajectory future feature $\hat{\mathbf{z}}_{t,k_w}^{(n)}$, the ego-status feature $\mathbf{s}_t$, and the planning feedback vector $\mathbf{q}_t = [r_t, \mathbf{u}_t]$, where r_t and $\mathbf{u}_t$ are the measured post-hoc total and sub-score feedback. Rather than generating a new plan from scratch, the model is designed to perform a conservative, feedback-driven correction of the selected proposal.

The trajectory, scene feature, ego-status feature, and planning feedback are first embedded into four tokens and fused by a lightweight self-attention block H. Then we inject the planning feedback directly into the trajectory representation $\mathbf{t}_H^{\mathrm{traj}}$ through FiLM (Perez et al., 2018) modulation, so that the refinement is explicitly conditioned on why the current trajectory is weak:

$$[\boldsymbol{\gamma}, \boldsymbol{\beta}] = \psi_{\mathrm{FiLM}}(\mathbf{q}_t), \qquad \mathbf{t}_t' = (1 + \boldsymbol{\gamma}) \odot \mathbf{t}_H^{\mathrm{traj}} + \boldsymbol{\beta}, \tag{13}$$

where $\mathbf{t}_H^{\mathrm{traj}}$ denotes the trajectory embedding after attention fusion. The modulated feature set is then decoded through MLPs to predict both a residual trajectory correction $\Delta\mathbf{y}_t$ and a learnable confidence gate g_t, and the refined trajectory is obtained by a gated residual update,

$$\bar{\mathbf{y}}_t = \mathbf{y}_t^w + g_t \Delta\mathbf{y}_t. \tag{14}$$

This structure keeps the refinement local and stable, because the model edits the selected trajectory according to explicit planning feedback, while the gate controls the magnitude of the correction.

Training and inference. For training the refiner to generate better trajectories, we construct weak/strong trajectory pair $(\mathbf{y}_t^-, \mathbf{y}_t^+)$ dataset from trajectory vocabularies (Li et al., 2025c). Given a trajectory, we sample a slightly better target trajectory from the candidate pool using two principles: improving the weakest planning metric, and taking small score-improving steps in the ranked trajectory set. This pair construction teaches the model to perform realistic local improvement, instead of learning arbitrary long-range trajectory jumps. Given a trajectory-pair $\mathbf{y}_t^-, \mathbf{y}_t^+$ ($\mathbf{y}_t^-$ plays the role of $\mathbf{y}_t^w$), the model is trained to match the stronger trajectory through a weighted regression objective:

$$\mathcal{L}_{\mathrm{refine}} = \mathcal{L}_{\mathrm{xy}}(\bar{\mathbf{y}}_t, \mathbf{y}_t^+) + \lambda_h \mathcal{L}_{\mathrm{heading}}(\bar{\mathbf{y}}_t, \mathbf{y}_t^+), \tag{15}$$

where $\mathcal{L}_{\mathrm{xy}}$ means the L1 loss and $\mathcal{L}_{\mathrm{heading}}$ is the heading loss with the weight λ_h. We further randomly drop the feedback during training to enable classifier-free guidance (CFG) at inference (Ho & Salimans, 2022). This reduces over-reliance on retrieved feedback and improves robustness when cross-domain memories are only partially aligned with the current scene, allowing the refiner to retain scene-conditioned correction ability. At test time, the model refines the current champion trajectory under multiple guidance strengths and outputs several corrected candidates, which are then fed back into the downstream evaluation pipeline for selection.

3.5 Iterative Self-Evolving Inference

This section ties the previous components into the test-time inference algorithm, i.e., the concrete instantiation of the abstract reflection step $\mathcal{R}$ introduced in Sec. 3.1. Each round chains SF, FAE, post-hoc feedback, and memory writing; from $n > 1$ the PAR additionally consumes retrieved memory to correct the previously selected trajectory before re-evaluation, so memory accumulation and trajectory improvement compound across rounds.

Given $\mathcal{O}_t$, the base planner produces $\mathcal{Y}_t^{(1)} = \pi(\mathcal{O}_t)$. At each round n EvoAD applies the SF, extracts future latents for the remaining candidates, and uses the FAE to select the best proposal:

$$\mathbf{y}_t^{(n)} = \arg \max_{\mathbf{y}_{t,k}^{(n)} \in \tilde{\mathcal{Y}}_t^{(n)}} \hat{s}_{t,k}^{(n)} \quad \text{with} \quad \hat{s}_{t,k}^{(n)} = \textsc{Eval}\big(\mathbf{h}_{t,k}^{(n)}\big). \tag{16}$$

Table 1: Quantitative comparison on the `navtest` split of NAVSIM v1 (PDMS) and v2 (EPDMS). EvoAD is built on WoTE, and Rn denotes the n-th reflection round. NC: no at-fault collision; DAC: drivable area compliance; TTC: time-to-collision; Comf.: comfort; EP: ego progress. The best results are in bold. *Original v2 evaluation protocol.

Method	Venue	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑	EPDMS ↑
UniAD (Hu et al., 2023)	CVPR'23	97.8	91.9	92.9	**100**	78.8	83.4	–
PARA-Drive (Weng et al., 2024)	CVPR'24	97.9	92.4	93.0	99.8	79.3	84.0	–
DRAMA (Yuan et al., 2024a)	ISRR'24	98.0	93.1	94.8	**100**	80.1	85.5	–
TransFuser (Chitta et al., 2022)	TPAMI'22	97.7	92.8	92.8	**100**	79.2	84.0	76.7*
Hydra-MDP++ (Li et al., 2025a)	arXiv'25	98.6	**98.6**	95.1	**100**	85.7	91.0	85.1*
DiffusionDrive (Liao et al., 2025)	CVPR'25	98.2	96.2	94.7	**100**	82.2	88.1	84.8*
WoTE (Li et al., 2025b)	ICCV'25	98.5	96.8	94.9	99.9	81.9	88.3	84.2
iPad (Guo et al., 2025)	arXiv'25	98.6	98.3	94.9	**100**	88.0	91.7	84.1
DrivoR (Kirby et al., 2026)	CVPR'26	98.9	98.3	96.2	**100**	89.1	93.1	88.4
SparseDriveV2 (Sun et al., 2026)	ECCV'26	98.5	98.4	95.0	99.9	88.6	92.0	90.1
EvoAD (R1)	–	99.2	96.4	97.0	99.9	85.7	90.6	87.0
EvoAD (R2)	–	**99.3**	97.8	97.3	99.9	88.3	92.6	89.6
EvoAD (R5)	–	**99.3**	98.1	**97.6**	99.9	**89.4**	**93.4**	**91.5**

After $\mathbf{y}_t^{(n)}$ is executed, its outcome yields the feedback $(r_t^{(n)}, \mathbf{u}_t^{(n)})$. For $n > 1$, retrieved neighbours $\mathcal{N}_t^{(n)}$ and the previous round's feedback condition the PAR to produce a corrected proposal,

$$\bar{\mathbf{y}}_t^{(n)} = \textsc{Refine}_\phi\Big(\mathbf{y}_t^{(n-1)}, \mathcal{N}_t^{(n)}, r_t^{(n-1)}, \mathbf{u}_t^{(n-1)}\Big), \tag{17}$$

which is inserted back into the candidate pool and re-evaluated alongside the planner proposals, so the original candidates remain as fallbacks while feedback-aware corrections are introduced progressively. The memory is updated by overwriting any earlier record of scene t only when the new trajectory strictly improves on the stored driving score. After N rounds EvoAD outputs $\mathbf{y}_t^\star = \mathbf{y}_t^{(N)}$, closing a self-evolving loop in which past post-hoc reflections become priors for future refinement.

4 Experiments

We first compare EvoAD with representative E2E driving methods on NAVSIM (Dauner et al., 2024) to assess its overall planning performance. We then conduct module-level ablations and progressive analysis to understand where the improvement comes from and how self-evolving reflection enhances trajectory quality. To further examine generalization, we test EvoAD across different base planners and memory sources from different data splits. Finally, qualitative studies visualize how future-aware evaluation and memory-guided refinement correct unsafe or inefficient trajectories.

4.1 Benchmark and Implementation Settings

We conduct experiments on NAVSIM v1 and v2 (Cao et al., 2025), large-scale AD benchmarks with 100K scenarios, evaluated by PDMS and EPDMS, respectively. All trainable modules are trained on the `navtrain` split (85k) and evaluated on `navtest` (12k), where evaluated trajectories are accumulated as episodic memory for subsequent reflection rounds. We use WoTE (Li et al., 2025b) as the base planner, which generates 256 candidate trajectories per scene. Implementation details are in Appendix B.

4.2 Main Results

Comparison with baselines. Table 1 compares EvoAD with existing E2E driving methods on NAVSIM `navtest`. Based on the WoTE planner, EvoAD (R1) first applies the SF and FAE to rerank the initial trajectory candidates, improving PDMS from 88.3 to 90.6. This shows that future-latent evaluation alone already enhances the base planner. Starting from R2, EvoAD introduces the retrieval-guided PAR, which uses previous reflection feedback to improve weak aspects of the selected trajectory. This brings a clear gain from 90.6 to 92.6 PDMS, with EP increasing from 85.7 to 88.3 and DAC from 96.4 to 97.8. Through self-evolution, EvoAD achieves the best performance at R5, reaching a PDMS of 93.4 on NAVSIM v1 and an EPDMS of 91.5 on NAVSIM v2.

Table 2: Module-level ablation studies on NAVSIM v1 `navtest` (PDMS). ✓ marks the enabled modules; WoTE and GTRS + Memory are baselines without our modules.

Method	SF	FAE	PAR	R1	R2	R5
WoTE	–	–	–	88.33	88.33	88.33
GTRS + Memory	–	–	–	90.27	90.35	90.44
WoTE + SF + Memory	✓	–	–	89.05	89.94	90.87
+ only FAE	✓	✓	–	90.58	91.58	92.76
+ FAE and PAR	✓	✓	✓	**90.58**	**92.61**	**93.36**

Table 3: Generalization studies on different base planners. Ori. denotes the original single-modality output; R2 is the mean of four refined candidates generated with four CFG strengths in Round 2.

Planner	Ori.	R2	NC	DAC	EP	TTC
TransFuser	84.24	89.79	98.48	96.80	85.14	95.10
DiffusionDrive	88.19	91.33	98.62	97.86	86.39	96.09
WoTE	88.33	91.24	98.94	98.01	85.61	96.46
iPad	**91.91**	**94.02**	**99.05**	**99.13**	**90.85**	**96.38**

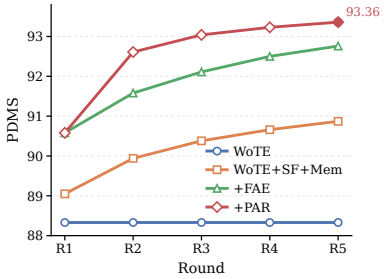


Figure 3: Improvement across reflection rounds.

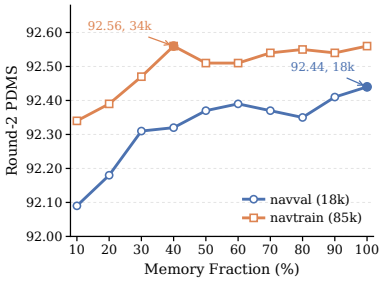


Figure 4: Effect of retrieved memory scale on trajectory refinement.

Ablation studies. Table 2 reports the module-level ablation of EvoAD. Starting from the WoTE+SF+Memory baseline, adding FAE improves the final R5 PDMS from 90.87 to 92.76, bringing the largest gain of +1.89. This indicates that trajectory-conditioned future latents provide an effective reranking signal beyond safety filtering alone. Further adding PAR improves R5 PDMS from 92.76 to 93.36, showing that feedback-guided residual refinement can further enhance the selected trajectory.

Multi-round self-evolution analysis. Fig. 3 illustrates the improvement across reflection rounds. Without the proposed FAE and refinement modules, the WoTE baseline remains unchanged and the WoTE+SF+Memory variant improves only gradually from 89.05 to 90.87, showing that memory alone is insufficient for self-evolution. The full EvoAD further improves from 90.58 at R1 to 92.61 at R2 and 93.36 at R5, demonstrating that retrieval-guided planning-aware refinement starts to take effect from the second round and brings further gains in later rounds.

Effectiveness on different base planners. Table 3 shows that the PAR produces useful improvement candidates across different base planners. The mean R2 PDMS improves over the original planning trajectory, with larger gains on weaker planners, e.g., +5.55 for TransFuser and +3.14 for DiffusionDrive. The improvement remains positive for stronger planners, including +2.91 for WoTE and +2.11 for iPad. This indicates that the PAR is not planner-specific; it captures reusable correction patterns that generalize across different trajectory distributions.

Cross-domain generalization. We study whether memory accumulated from other splits can transfer to `navtest`. As shown in Fig. 4, both `navval` and `navtrain` memories improve R2 performance as the memory fraction increases. The `navval` memory improves the R2 PDMS from 92.09 to 92.44 when maintaining 18k val-split experience, while the larger `navtrain` memory reaches 92.56 at the scale of 34k samples. This suggests that EvoAD learns transferable trajectory-feedback priors rather than relying on split-specific memorization.

Reactive closed-loop evaluation. We further evaluate EvoAD on NavHard, where surrounding agents respond to the ego vehicle. EvoAD improves across reflection rounds, with EPDMS increasing from 31.31 at R1 to 41.37 at R11 (Appendix C.5).

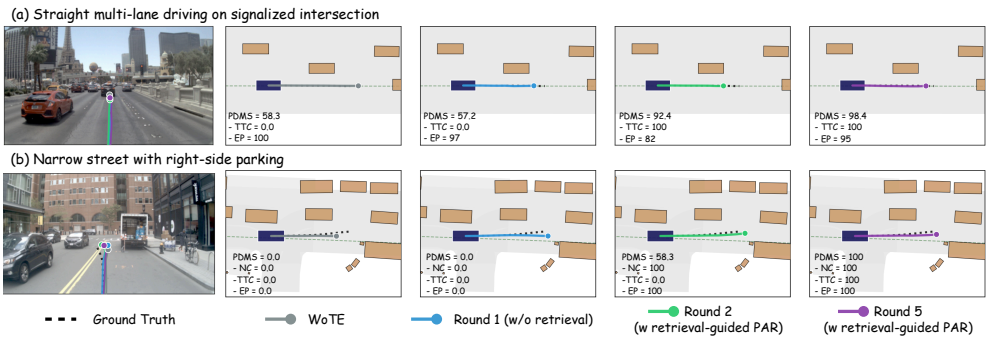


Figure 5: Qualitative improvement of EvoAD across reflection rounds.

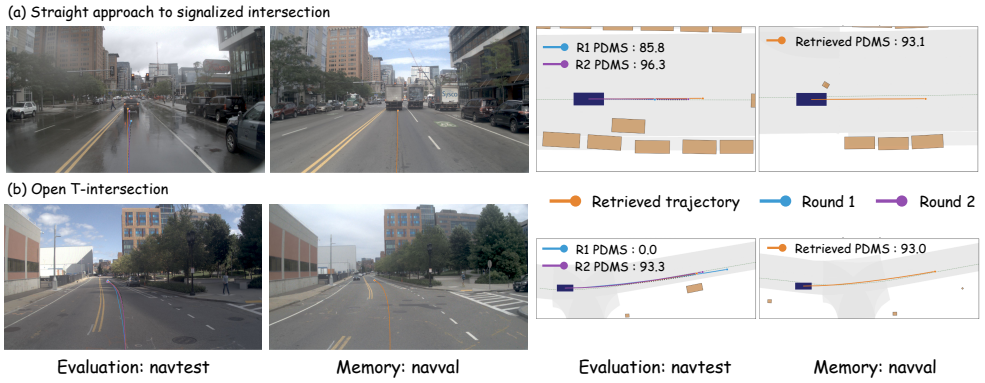


Figure 6: Generalization of memory-guided refinement across related driving scenes.

4.3 Qualitative Studies

As shown in Fig. 5, EvoAD improves trajectories through reflection rounds. In the signalized multi-lane intersection, reflection first removes the TTC risk and then improves efficiency, increasing PDMS from 58.3 to 98.4. In the narrow street with right-side parking, EvoAD corrects an initially unsafe or stalled trajectory into a collision-free and TTC-safe plan with full progress, raising PDMS from 0.0 to 100.0. Fig. 6 further shows that retrieved memories can transfer across related scenes. Even when the retrieved case differs in appearance or comes from a different split, its decision-outcome pattern provides useful refinement guidance. This improves the current trajectory from 85.8 to 96.3 in the signalized intersection and from 0.0 to 93.3 in the open T-intersection.

5 Conclusion and Limitations

We propose **EvoAD**, contributing a post-deployment self-evolution framework to E2E AD that enables vehicles to improve from past driving experiences without retraining or replacing the base model. EvoAD uses latent BEV imagination to evaluate trajectory outcomes, stores reflected experiences in memory, and retrieves relevant experiences to refine future plans. Experiments on NAVSIM v1 and v2 demonstrate the effectiveness of EvoAD, which achieves gains through multi-round reflection and generalizes across memory sources and base planners. EvoAD also has limitations. We have validated it on NAVSIM v1, v2, and the reactive NavHard split. In future work, we will extend it to more benchmarks and simulators such as CARLA (Dosovitskiy et al., 2017). Stronger latent world models and uncertainty-aware retrieval may further improve refinement reliability.

AI use statement

Generative AI tools were used for language editing and limited assistance with code development and debugging. They were also occasionally used to help organize or clarify discussions related to

experimental design and result interpretation. All AI-assisted outputs were independently reviewed and verified by the authors. The authors made all research, methodological, experimental, and interpretative decisions and take full responsibility for the content of this work.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. All experiments are conducted on the publicly released NAVSIM benchmarks, which are built on the OpenScene/nuPlan driving logs and used under their respective licenses; no new data were collected and no human subjects were involved. EvoAD is a research method for improving E2E planners through post-hoc reflection, and all evaluations are performed on simulation-based benchmarks. Since AD is safety-critical, improvements on these benchmarks do not by themselves guarantee safe behavior on public roads; any real-world deployment would require extensive additional validation, including closed-loop and on-road safety testing, as well as compliance with applicable regulations.

REPRODUCIBILITY STATEMENT

We describe the problem formulation and all components of EvoAD in Sec. 3.1–Sec. 3.5. Appendix A details the benchmarks, data splits, evaluation protocols, and the reused foundation components with their released checkpoints. Appendix B specifies the module architectures, loss functions, training data construction, training hyperparameters, inference configuration, and memory organization. Per-round results, ablations, and per-module latency are reported in Appendix C, and the notation is summarized in Appendix E. All experiments use publicly available benchmarks with their official evaluation code.

REFERENCES

Nassim Belmecheri, Arnaud Gotlieb, Nadjib Lazaar, and Helge Spieker. Trustworthy automated driving through qualitative scene understanding and explanations. In *Transport Research Arena Conference*, pp. 490–496. Springer, 2024.

Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

Tianhui Cai, Yifan Liu, Zewei Zhou, Haoxuan Ma, Seth Z Zhao, Zhiwen Wu, Xu Han, Zhiyu Huang, and Jiaqi Ma. Driving with regulation: Trustworthy and interpretable decision-making for autonomous driving with retrieval-augmented reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 38287–38295, 2026.

Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, et al. Pseudo-simulation for autonomous driving. *arXiv preprint arXiv:2506.04218*, 2025.

Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10164–10183, 2024.

Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE transactions on pattern analysis and machine intelligence*, 45(11):12878–12895, 2022.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, et al. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. volume 37, pp. 28706–28719, 2024.

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

Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Dingkang Liang, Chong Zhang, Dingyuan Zhang, Hongwei Xie, Bing Wang, and Xiang Bai. Orion: A holistic end-to-end autonomous driving framework by vision-language instructed action generation. pp. 24823–24834, 2025.

Ray Fuller. Towards a general theory of driver behaviour. *Accident analysis & prevention*, 37(3): 461–472, 2005.

Ruiyuan Gao, Kai Chen, Enze Xie, Lanqing Hong, Zhenguo Li, Dit-Yan Yeung, and Qiang Xu. Magicdrive: Street view generation with diverse 3d geometry control. 2023.

John A Groeger. *Understanding driving: Applying cognitive psychology to a complex everyday task*. Routledge, 2013.

Ke Guo, Haochen Liu, Xiaojun Wu, Jia Pan, and Chen Lv. ipad: Iterative proposal-centric end-to-end autonomous driving. *arXiv preprint arXiv:2505.15111*, 2025.

Niklas Hanselmann, Katrin Renz, Kashyap Chitta, Apratim Bhattacharyya, and Andreas Geiger. King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In *European Conference on Computer Vision*, pp. 335–352. Springer, 2022.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 17853–17862, 2023.

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

Ellington Kirby, Alexandre Boulch, Yihong Xu, Yuan Yin, Gilles Puy, Éloi Zablocki, Andrei Bursuc, Spyros Gidaris, Renaud Marlet, Florent Bartoccioni, et al. Driving on registers. 2026.

Kailin Li, Zhenxin Li, Shiyi Lan, Yuan Xie, Zhizhong Zhang, Jiayi Liu, Zuxuan Wu, Zhiding Yu, and Jose M Alvarez. Hydra-mdp++: Advancing end-to-end driving via expert-guided hydra-distillation. *arXiv preprint arXiv:2503.12820*, 2025a.

Yingyan Li, Lue Fan, Jiawei He, Yuqi Wang, Yuntao Chen, Zhaoxiang Zhang, and Tieniu Tan. Enhancing end-to-end autonomous driving with latent world model. *arXiv preprint arXiv:2406.08481*, 2024a.

Yingyan Li, Yuqi Wang, Yang Liu, Jiawei He, Lue Fan, and Zhaoxiang Zhang. 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, 2025b.

Yue Li, Meng Tian, Dechang Zhu, Jiangtong Zhu, Zhenyu Lin, Zhiwei Xiong, and Xinhai Zhao. Drive-r1: Bridging reasoning and planning in vlms for autonomous driving with reinforcement learning. 40(8):6708–6716, 2026.

Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, et al. Hydra-mdp: End-to-end multimodal planning with multi-target hydra-distillation. *arXiv preprint arXiv:2406.06978*, 2024b.

Zhenxin Li, Wenhao Yao, Zi Wang, Xinglong Sun, Joshua Chen, Nadine Chang, Maying Shen, Zuxuan Wu, Shiyi Lan, and Jose M Alvarez. Generalized trajectory scoring for end-to-end multimodal planning. 2025c.

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

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Lin Liu, Ziying Song, Caiyan Jia, Hangjun Ye, Xiaoshuai Hao, Long Chen, et al. Driveworld-vla: Unified latent-space world modeling with vision-language-action for autonomous driving. *arXiv preprint arXiv:2602.06521*, 2026.

Jiachen Lu, Ze Huang, Zeyu Yang, Jiahui Zhang, and Li Zhang. Wovogen: World volume-aware diffusion for controllable multi-camera driving scene generation. In *European conference on computer vision*, pp. 329–345. Springer, 2024.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Wenchao Sun, Xuewu Lin, Keyu Chen, Zixiang Pei, Xiang Li, Yining Shi, and Sifa Zheng. Sparsedrivev2: Scoring is all you need for end-to-end autonomous driving. In *European Conference on Computer Vision*, pp. 446–463. Springer, 2026.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. 2020.

Xiaofeng Wang, Zheng Zhu, Guan Huang, Xinze Chen, Jiagang Zhu, and Jiwen Lu. Drivedreamer: Towards real-world-drive world models for autonomous driving. In *European conference on computer vision*, pp. 55–72. Springer, 2024a.

Yuqi Wang, Jiawei He, Lue Fan, Hongxin Li, Yuntao Chen, and Zhaoxiang Zhang. Driving into the future: Multiview visual forecasting and planning with world model for autonomous driving. pp. 14749–14759, 2024b.

Licheng Wen, Daocheng Fu, Xin Li, Xinyu Cai, Tao Ma, Pinlong Cai, Min Dou, Botian Shi, Liang He, and Yu Qiao. Dilu: A knowledge-driven approach to autonomous driving with large language models. *arXiv preprint arXiv:2309.16292*, 2023.

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

Pengxuan Yang, Ben Lu, Zhongpu Xia, Chao Han, Yinfeng Gao, Teng Zhang, Kun Zhan, XianPeng Lang, Yupeng Zheng, and Qichao Zhang. Worldrft: Latent world model planning with reinforcement fine-tuning for autonomous driving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 11649–11657, 2026.

Hao Ye, Mengshi Qi, Zhaohong Liu, Liang Liu, and Huadong Ma. Safedriverag: Towards safe autonomous driving with knowledge graph-based retrieval-augmented generation. pp. 11170–11178, 2025.

Chengran Yuan, Zhanqi Zhang, Jiawei Sun, Shuo Sun, Zefan Huang, Christina Dao Wen Lee, Dongen Li, Yuhang Han, Anthony Wong, Keng Peng Tee, et al. Drama: An efficient end-to-end motion planner for autonomous driving with mamba. 2024a.

Jianhao Yuan, Shuyang Sun, Daniel Omeiza, Bo Zhao, Paul Newman, Lars Kunze, and Matthew Gadd. Rag-driver: Generalisable driving explanations with retrieval-augmented in-context learning in multi-modal large language model. *arXiv preprint arXiv:2402.10828*, 2024b.

Yupeng Zheng, Pengxuan Yang, Zebin Xing, Qichao Zhang, Yuhang Zheng, Yinfeng Gao, Pengfei Li, Teng Zhang, Zhongpu Xia, Peng Jia, et al. World4drive: End-to-end autonomous driving via intention-aware physical latent world model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 28632–28642, 2025.

APPENDIX

This appendix collects the implementation-level details and supplementary experiments that support the main text. Sec. A describes the benchmark and implementation details. Sec. B specifies the model architectures, training and test configuration of the FAE and the PAR. Sec. C provides complementary ablations and latency studies, as well as the results on NAVSIM v2 and NavHard. Additional qualitative case studies are shown in Sec. D, and Sec. F finally discusses the feedback source, asynchronous reflection, the stability of multi-round reflection, and the refined candidates seen by the evaluator.

A DATASET AND SPLITS

NAVSIM v1 benchmark. Our main experiments are conducted on NAVSIM v1 (Dauner et al., 2024), a large-scale, non-reactive closed-loop benchmark built on top of the OpenScene/nuPlan logs. Each scenario provides 8 surround cameras and a 360° LiDAR sweep at the current frame, the past 1.5 s of ego state, a high-definition map context, and a high-level navigation command. The planner predicts a 2 Hz, 4 s future trajectory in the ego frame, which is then evaluated by the official Predictive Driver Model Score (PDMS), aggregated as $\mathrm{PDMS} = \mathrm{NC} \times \mathrm{DAC} \times (5\,\mathrm{EP} + 5\,\mathrm{TTC} + 2\,\mathrm{Comf.})/12$. Driving-direction compliance (DDC) is additionally reported as a sub-score. All PDMS values in this paper follow the official NAVSIM v1 scoring code with the default sub-metric weights.

NAVSIM v2 benchmark. We additionally evaluate on NAVSIM v2 (Cao et al., 2025), which extends NAVSIM v1 with the Extended PDMS (EPDMS). We report EPDMS on the same `navtest` split (12,146 scenes) with the official NAVSIM v2 evaluation code; results marked with * in Table 1 were reported under the original v2 evaluation protocol, before the human-behavior filtering fix.

NavHard benchmark. To examine EvoAD under reactive traffic, we further evaluate on the `navhard` split of NAVSIM v2.2. Surrounding vehicles react to the motion of the ego vehicle, and the evaluation follows a two-stage protocol in which 225 first-stage scenes expand into 5,912 scored tokens; the final EPDMS is the product of the stage-1 and stage-2 scores. All NavHard numbers follow the corrected official metric. Since all trainable modules are trained only on `navtrain`, which has no overlap with NavHard, this evaluation is a zero-shot transfer. We run $N{=}11$ reflection rounds on this split.

Splits used in this work. We train EvoAD's two trainable modules (FAE and PAR) on `navtrain` (85,109 scenes), and report all PDMS numbers on `navtest` (12,146 scenes). For internal validation during training we additionally hold out an 18,179-scene split of `navtrain` as `navval`, which is also reused as an in-distribution memory pool in the cross-split memory generalisation study of the main text.

Foundation components. EvoAD reuses two off-the-shelf models as frozen foundation modules: WoTE (Li et al., 2025b) as the E2E base planner that supplies the candidate trajectories and the latent world model in the main text, and GTRS (Li et al., 2025c) as the reusable SF described in Sec. 3.2. Both are used with their official released checkpoints and are not finetuned.

B IMPLEMENTATION DETAILS

B.1 MODULE ARCHITECTURES.

Table 4 summarises the input and output, model layers and parameter counts of the two trainable modules. Both use a token dimension of $d{=}256$ with 4 attention heads, and the PAR additionally uses FiLM modulation (Perez et al., 2018) to inject the planning feedback $\mathbf{q}_t$ into the trajectory token. The abstract notation ($\mathbf{F}_t$, H, ψ_{FiLM}, h^{rank}, h^{sub}, h^{score}, h_{Δ}, h_g) follows Sec. 3.3 and Sec. 3.4.

Trajectory safety filter. The SF is a GTRS-type trajectory scoring module whose architecture follows GTRS (Li et al., 2025c). It scores the 256 candidates produced by the base planner and

Table 4: Architecture summary of EvoAD's two trainable modules. Both use token dimension d=256 with 4 attention heads.

	FAE	PAR
Per-candidate inputs	$\mathbf{y}_{t,k}^{(n)}$, $\hat{\mathbf{z}}_{t,k}$	$\mathbf{y}_t^w$, $\hat{\mathbf{z}}_{t,k_w}^{(n)}$
Shared inputs	$\mathbf{s}_t$, $\mathbf{F}_t$	$\mathbf{s}_t$, $\mathbf{q}_t$
Layer stack	1 intra + 2 scene-cross + 2 cand-self	1 self-attn fusion + 1 AttnPool
Per-scene outputs	$\{\hat{s}_{t,k}^{(n)}, \hat{\mathbf{u}}_{t,k}^{(n)}, \hat{r}_{t,k}^{(n)}\}_{k=1}^{K}$	$\bar{\mathbf{y}}_t$
Parameters	$\approx 4.0\,\mathrm{M}$	$\approx 3.2\,\mathrm{M}$

discards the low-scoring ones, so that the FAE performs future-latent evaluation only on a higher-quality top-K subset.

B.2 Loss Details.

Future-aware evaluator losses. $\mathcal{L}_{\text{list}}$ is a pairwise margin hinge over the predicted ranking scores: for two candidates i, j in the same scene with ground-truth PDMS values r_i, r_j,

$$\mathcal{L}_{\text{list}} = \mathbb{E}_{(i,j)}\big[\max\big(0,\ -\text{sign}(r_i - r_j)\,(\hat{s}_i - \hat{s}_j) + m_{ij}\big)\big]\,, \tag{18}$$

with margin m_{ij} proportional to the PDMS gap $|r_i - r_j|$. $\mathcal{L}_{\text{sub}}$ is the standard focal loss (Lin et al., 2017; Guo et al., 2025) on the predicted sub-scores with (α, γ)=$(0.75, 1.0)$. $\mathcal{L}_{\text{score}} = \|\hat{r}_{t,k}^{(n)} - r_{t,k}^{(n)}\|^2$ is an L2 regression of the predicted total to the ground-truth PDMS.

Planning-aware refiner losses. $\mathcal{L}_{\text{xy}} = \frac{1}{T}\sum_{\tau=1}^{T}\big\|(\bar{x}_\tau, \bar{y}_\tau) - (x_\tau^+, y_\tau^+)\big\|_1$ is an L1 loss between the refined trajectory $\bar{\mathbf{y}}_t$ and the strong target $\mathbf{y}_t^+$. $\mathcal{L}_{\text{heading}} = \frac{1}{T}\sum_{\tau=1}^{T}\big(1 - \cos(\bar{\theta}_\tau - \theta_\tau^+)\big)$ is a wrap-safe cosine loss on the heading channel; we set λ_h=0.5.

B.3 Training Configuration

Training settings and hyperparameters are summarised in Table 5.

Table 5: Training configuration of EvoAD's two trainable modules based on 4× NVIDIA RTX 4090 GPUs.

Setting	FAE	PAR
Optimiser	AdamW	AdamW
Learning rate	3×10^{-4}	3×10^{-4}
Weight decay	1×10^{-4}	1×10^{-4}
Learning-rate schedule	Cosine annealing	Cosine annealing
Batch size (per GPU)	64 scenes (K=20 each)	256 weak/strong pairs
Mixed precision	bfloat16	bfloat16
Gradient clip	0.5	1.0
Train / val samples	80,854 / 4,255 scenes	3,842,297 / 202,315 pairs

Future-aware evaluator training data. For each `navtrain` scene we run the WoTE base planner once to obtain 256 candidate trajectories together with their per-candidate future BEV latents, GTRS-rank them, and keep the top-K = 20 as the candidate set seen by the evaluator. Each retained candidate is then scored offline by the official NAVSIM scoring simulator, providing the total score $r_{t,k}^{(n)}$ and six sub-scores $\mathbf{u}_{t,k}^{(n)}$ that serve as the regression targets.

Refiner training data. For each `navtrain` scene we score every trajectory in a per-scene candidate pool with the scoring simulator and form 40 weak/strong pairs $(\mathbf{y}_t^-, \mathbf{y}_t^+)$ by combining two complementary principles: (i) preferring transitions that improve the weakest sub-score, and (ii) restricting the gap to small score-improving steps in the ranked pool. This produces realistic local edits and prevents the refiner from learning arbitrary long-range trajectory jumps. During training, the feedback branch is additionally dropped to its null value with probability $p = 0.1$ to enable the multi-strength CFG inference described in Sec. 3.4.

B.4 Inference Configuration

CFG-based refiner inference. The refiner consumes the planning feedback $\mathbf{q}_t = [r_t^w, \mathbf{u}_t^w]$ as an explicit cue about which axes the previous trajectory was weak on. In same-scene refinement this cue is well aligned, since $\mathbf{q}_t$ is the target scene's own previous-round PDMS. In cross-scene refinement, however, the last-round trajectory and its $\mathbf{q}_t$ are drawn from a retrieved memory entry whose PDMS was computed on a different scene, so the feedback may not actually describe the target scene's weaknesses. To handle both regimes with a single network we adopt CFG (Ho & Salimans, 2022): the feedback branch is randomly masked to its null value during training (Sec. B.3), so that the same parameters learn both a *conditional* response that exploits $\mathbf{q}_t$ and an *unconditional* response that ignores it.

At inference time we run the refiner twice — once with the unmasked $\mathbf{q}_t$ to obtain $\mathrm{pred}_{\mathrm{cond}}$, and once with $\mathbf{q}_t$ masked to the null value to obtain $\mathrm{pred}_{\mathrm{uncond}}$ — and linearly extrapolate at four guidance strengths,

$$\mathrm{pred}_g \;=\; \mathrm{pred}_{\mathrm{uncond}} \;+\; g \cdot (\mathrm{pred}_{\mathrm{cond}} - \mathrm{pred}_{\mathrm{uncond}}), \qquad g \in \{0.5, 1.0, 1.5, 2.0\}. \tag{19}$$

This produces 4 refined candidates per scene from only 2 network forwards. All four are returned to the candidate pool and re-evaluated by the FAE, so the post-selection automatically picks whichever guidance strength best matches the current $\mathbf{q}_t$. The resulting same-domain and cross-domain comparison against single-forward baselines is reported in Sec. C.1.

Memory organization and size. The episodic memory $\mathcal{M}$ keeps at most one record per scene, and a record is overwritten rather than appended, only when a new trajectory strictly improves the stored PDMS. The memory size is therefore bounded by the number of scenes and independent of the number of reflection rounds. Each record stores a 256-d scene embedding $\mathbf{e}_t$, an 8×3 trajectory, its total score, six sub-scores, and a 256-d future latent, i.e., 543 scalars; the largest evaluated pool ($85{,}109$ `navtrain` entries) thus occupies about $176\,$MiB in FP32 ($\approx 88\,$MiB in FP16) before indexing overhead. During planning the memory is a read-only lookup table, and it holds no images, LiDAR point clouds, or full BEV feature maps. With memory retrieval included, EvoAD adds only about 8% latency to the base planner (Sec. C.3).

C Supplementary Results

C.1 Classifier-Free Guidance Ablation

We compare our CFG sweep against two single-forward baselines that share the same first-round selection and refiner checkpoint: (a) a single unconditional forward (g=0, feedback masked to null) and (b) a single conditional forward (g=1, feedback fed unchanged). Table 6 shows that the CFG sweep brings two benefits over either single forward: it generates four diverse corrections from only two network forwards, and the post-evaluator selection over those four candidates lifts R2 PDMS by $+0.56$ in same-domain refinement and $+0.18$ in cross-domain refinement.

Table 6: CFG ablation on NAVSIM `navtest`, comparing same-domain and cross-domain refiner queries under different guidance settings.

Refiner protocol at n=2	#cand	#fwd	Same-domain R2	Cross-domain R2
Single unconditional forward (g=0)	1	1	91.93	92.26
Single conditional forward (g=1)	1	1	92.05	92.22
CFG sweep, $g \in \{0.5, 1, 1.5, 2\}$ (ours)	**4**	**2**	**92.61**	**92.44**
Δ(CFG sweep − best single)	—	—	$+0.56$	$+0.18$

C.2 Full Multi-Round Reflection Trajectory

Table 7 extends the main-text ablation by reporting the intermediate rounds R3 and R4, while Table 8 provides the full PDMS sub-score decomposition for EvoAD across R1–R5. These results clarify the roles of the two trainable modules: without the FAE, the PAR-only variant quickly plateaus around 90.2, whereas the FAE-only variant still improves from 90.58 to 92.76 over five rounds.

This suggests that the FAE drives most of the early gain, while the PAR contributes mainly to the later-stage improvement when combined with reliable future-aware selection.

Table 7: Per-round PDMS on `navtest` (%), where − means the component is absent.

Variant	R1	R2	R3	R4	R5
EvoAD (full)	90.58	92.61	93.04	93.23	**93.36**
− PAR (FAE only)	90.58	91.58	92.11	92.50	92.76
− FAE (PAR only)	89.05	89.93	90.12	90.21	90.23
− FAE − PAR	89.05	89.94	90.38	90.66	90.87
Pure GTRS + memory (no WoTE)	90.27	90.35	90.39	90.39	90.44
Pure WoTE + memory (no GTRS)	88.33	88.33	88.33	88.33	88.33

Table 8: Complete progression study on `navtest`.

Round	PDMS	NC	DAC	DDC	EP	TTC	Comf.
R1	90.58	99.24	96.43	99.98	85.70	97.00	99.92
R2	92.61	99.31	97.79	99.99	88.28	97.32	99.94
R3	93.04	99.31	97.97	99.99	88.91	97.49	99.94
R4	93.23	99.32	98.05	99.99	89.20	97.55	99.94
R5	**93.36**	**99.32**	**98.11**	99.99	**89.39**	**97.60**	**99.94**

The headline gains across rounds concentrate on EP ($+3.69$) and DAC ($+1.68$), with smaller improvements on the safety axes (NC $+0.08$, TTC $+0.60$). The DDC and Comf. axes are already saturated in $\mathbf{y}_t^{(1)}$ and remain so under reflection. Across R1→R5, the successive gains of the full model are $+2.03$, $+0.43$, $+0.19$, and $+0.13$ PDMS, showing that the improvement gradually converges to a plateau.

C.3 Per-Module Latency

Table 9: Per-module forward latency studies on a single NVIDIA RTX 4090.

Module	Mean (ms)
SF (GTRS)	54.37
WoTE Latent World Model	9.45
FAE	3.46
PAR	2.99
Memory retrieval — `navval` 18,179 (CPU)	0.04
Memory retrieval — `navval` 18,179 (GPU)	0.07
Memory retrieval — `navtrain` 85,109 (CPU)	0.23
Memory retrieval — `navtrain` 85,109 (GPU)	0.16

The per-scene cost decomposes as: *(i)* a one-time forward of the reused foundation components, i.e., the WoTE planner (image backbone and latent world model, the latter costing ≈ 9 ms) and the GTRS-based SF (≈ 54 ms); *(ii)* per round, ≈ 6.6 ms total for the FAE, the PAR CFG sweep, and memory retrieval. Measured end to end, a single full forward pass of EvoAD takes about 84 ms, of which the two modules we introduce (FAE and PAR) account for only 6.5 ms ($\approx 8\%$); the remainder comes from the reused base planner and SF. Cosine retrieval on the larger 85k pool stays sub-millisecond (0.16 ms on GPU), negligible next to the GTRS scorer and the visual backbone, so the choice of memory pool size is a quality decision rather than a latency decision.

Multiple reflection rounds are not performed repeatedly before executing the current action. Each round therefore corresponds to one improvement that takes effect the next time a similar scene is encountered, and at deployment a scene incurs only one full forward pass. Our NAVSIM study uses repeated reflection on the benchmark scenarios as a controlled way to quantify this self-evolution, rather than as a real-time evaluation of a full vehicle stack. The 93.4 PDMS reported in the main text is what the system converges to after five such rounds. Measured on the same RTX 4090 under an identical protocol, the architecture used by Hydra-MDP++ (Li et al., 2025a) is in the same range (71.9 ms) but reaches only 91.0 PDMS, while DrivoR (Kirby et al., 2026) raises the cost to 147.3 ms

($\approx 1.75\times$) with 93.1 PDMS. The overhead added by our modules, which lifts PDMS from 88.3 to 93.4, is therefore a favorable trade.

C.4 Results on NAVSIM v2

Table 10 reports the per-round EPDMS of EvoAD on the NAVSIM v2 `navtest` split. As on NAVSIM v1, EPDMS improves over rounds, from 87.01 at R1 to 91.52 at R5, i.e., +7.32 over the WoTE planner (84.2). The largest gain again appears at R2 (+2.55), when the retrieval-guided PAR is first applied. For reference, the public GTRS-Dense checkpoint (Li et al., 2025c) measures 90.27 PDMS on v1 and 85.17 EPDMS on v2 `navtest` in our evaluation, both below EvoAD (93.4 and 91.52).

Table 10: Per-round EPDMS of EvoAD on NAVSIM v2 `navtest`.

Variant	R1	R2	R3	R4	R5
EvoAD (full)	87.01	89.56	90.47	91.21	**91.52**
Δ w.r.t. previous round	–	+2.55	+0.91	+0.74	+0.31

C.5 Multi-Round Reflection on NavHard

We further run EvoAD on the reactive `navhard` split (Sec. A). Table 11 shows that the reflection mechanism remains effective under reactive traffic: EPDMS rises from 31.31 at R1 to 41.23 at R7 and then converges, with a net change of only +0.14 from R7 to R11 (41.37). A single regression occurs at R8 ($41.23 \rightarrow 40.44$), where the evaluator still judged the selected trajectory to be improving. Once the measured feedback of that trajectory becomes available in the next round, the loop recovers and surpasses the previous peak at R9 (41.33).

Table 11: Per-round EPDMS of EvoAD on NavHard (`navhard`, NAVSIM v2.2).

Round	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
EPDMS	31.31	31.79	32.63	34.15	36.86	39.01	41.23	40.44	41.33	41.25	**41.37**

C.6 Cross-Domain Memory on NAVSIM v2 and NavHard

Table 12: Cross-domain memory on NAVSIM v2 and NavHard. Same-domain memory is accumulated on the evaluation split itself.

Benchmark	Evaluation split	Memory source	R2 EPDMS
NAVSIM v2	`navtest`	`navtest` (same-domain)	89.56
	`navtest`	`navval` (cross-domain)	89.47
NavHard	`navhard`	`navhard` (same-domain)	31.79
	`navhard`	`navtrain` (cross-domain)	**34.18**

Sec. 4.2 shows that memories built on `navval` and `navtrain` transfer to NAVSIM v1 `navtest`. Table 12 shows the same behavior on the two additional benchmarks. On NAVSIM v2, using the `navval` memory for `navtest` inference gives 89.47 EPDMS at R2, on par with 89.56 for the same-domain memory. On NavHard, the memory is built entirely on `navtrain`, so all trajectories and feedback used by the refiner come from retrieved other scenes, and none of the target scene's own feedback is used. This cross-domain memory reaches 34.18 EPDMS at R2, higher than 31.79 for the same-domain multi-round reflection. Since NavHard is limited in scale (5,912 tokens), the much larger `navtrain` pool offers more relevant precedents for reflection. This suggests that the memory stores transferable decision–outcome patterns rather than scene-specific records, and that self-evolution does not require memory accumulated on the target domain.

C.7 Additional Ablations

Effect of the safety filter. Removing the SF and letting the FAE rank the full candidate set directly yields 89.6 PDMS at R1, below the 90.58 obtained with the SF in place. When the candidate set is large and contains many low-quality trajectories, the ranking of the evaluator is diluted, and the coarse pre-filter makes the subsequent future-latent evaluation more effective.

Refiner with baseline scorers. Table 3 shows that the refiner produces improved candidates for TransFuser, DiffusionDrive, WoTE, and iPad. When these refined candidates are instead selected by the own scorer of each baseline, none of the four planners obtains an R2 improvement, since these scorers are trained on their own candidate pools and do not recognize the new candidates produced by the refiner. Together with the memory-equipped baselines in Table 7, which barely improve across rounds, this indicates that memory alone is not sufficient for self-evolution: the gain requires both the refiner that generates improved candidates and the FAE that recognizes them.

D Qualitative Case Studies

This section provides qualitative `navtest` examples that visualize EvoAD's reflective behaviors: multi-round trajectory refinement on a single scene, cross-split memory retrieval that imports a useful trajectory from `navval`, and a multi-scene gallery covering diverse traffic configurations.

D.1 Five-Round Reflection Example

The leftmost panel shows the front camera; the BEV panels visualize the selected trajectory at successive reflection rounds together with its PDMS and sub-scores. The EP sub-score ("prog") fluctuates while the refiner trades progress against the other axes, and the loop ultimately converges to a trajectory that attains the maximum PDMS on this scene.

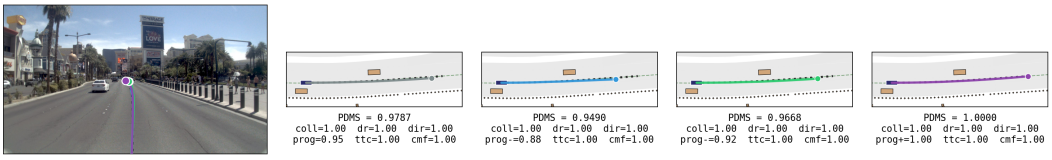


Figure 7: Five-round reflection on a free-space straight driving road.

D.2 Cross-Split Memory Retrieval Example

For the target scene (left two camera panels and the third BEV panel), the retrieval module fetches a similar trajectory from the `navval` memory pool (rightmost BEV); the PAR uses this retrieved trajectory, lifting the R1 selection (blue) to a refined R2 (purple) much closer to the human ground truth (dotted black). The retrieved `navval` entry, originally evaluated on its own scene, successfully transfers planning structure across splits.

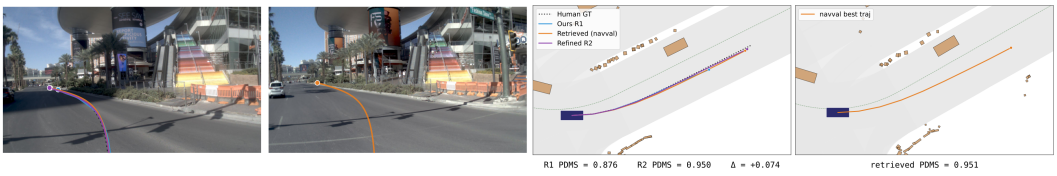


Figure 8: Cross-split memory retrieval on left-turn scenario.

D.3 Multi-Scene Reflection Showcase

For each scene, we visualize the front-camera view with the trajectories selected at successive reflection rounds projected onto the image, together with their PDMS. Across these scenes, EvoAD

corrects the initial selection over rounds and reaches a higher-PDMS trajectory, indicating that the reflection mechanism transfers across diverse driving situations rather than fitting a particular scene type.



Figure 9: Improvement on scenes covering different traffic configurations.

E NOTATION

Table 13 consolidates the core symbols of Sec. 3.1–Sec. 3.5. Shapes omit the scene/batch index; "–" marks abstract objects (procedures, sets) without a tensor shape.

Table 13: Core notation used throughout the paper.

Notation	Shape & Value	Description
Indices and dimensions		
N	5	total number of reflection rounds
K	20	evaluator candidate set size at training
L	1024	length of the scene keyval sequence $\mathbf{F}_t$
d	256	token dimension shared by both trainable modules
Scene-level features		
$\mathbf{F}_t$	$L \times d$	scene keyval reused from the GTRS backbone
$\mathbf{s}_t$	d	encoded ego-status feature
$\mathbf{e}_t$	d	scene embedding used for memory retrieval
Per-candidate features (round n, candidate k)		
$\mathbf{y}_{t,k}^{(n)}$	8×3	candidate trajectory (x, y, θ)
$\hat{\mathbf{z}}_{t,k}^{(n)}$	d	candidate-conditioned future BEV latent from LATENTWM
$\mathbf{y}_t^{(n)}$	8×3	trajectory selected by the evaluator at round n
FAE outputs		
$\hat{s}_{t,k}^{(n)}$	1	predicted ranking score
$\hat{\mathbf{u}}_{t,k}^{(n)}$	6	predicted PDMS sub-scores (NC, DAC, DDC, EP, TTC, Comf.)
$\hat{r}_{t,k}^{(n)}$	1	predicted total PDMS
Driving evaluator (PDMS) targets		
$r_{t,k}^{(n)}$	1	ground-truth total PDMS
$\mathbf{u}_{t,k}^{(n)}$	6	ground-truth PDMS sub-scores
PAR		
$\mathbf{y}_t^w$	8×3	warm-start trajectory retrieved from $\mathcal{N}_t^{(n)}$
$\mathbf{q}_t$	7	planning feedback $[r_t^w, \mathbf{u}_t^w]$
$\Delta\mathbf{y}_t$	8×3	predicted residual correction
g_t	$(0, 1)$	sigmoid update gate
$\tilde{\mathbf{y}}_t$	8×3	refined trajectory $\mathbf{y}_t^w + g_t\, \Delta\mathbf{y}_t$
Memory and retrieval		
$\mathcal{M}$	–	episodic memory pool
$m_{t,k}^{(n)}$	–	memory entry $(\mathbf{e}_t, \mathbf{y}_{t,k}^{(n)}, r_{t,k}^{(n)}, \mathbf{u}_{t,k}^{(n)}, \hat{\mathbf{z}}_{t,k}^{(n)})$
$\mathcal{N}_t^{(n)}$	–	cosine top-K' neighbours of $\mathbf{e}_t$ in $\mathcal{M}$
Training and inference		
$\mathbf{y}_t^+$	8×3	strong trajectory in a weak/strong refiner pair
$\mathbf{y}_t^-$	8×3	weak trajectory in a weak/strong refiner pair (training counterpart of $\mathbf{y}_t^w$)
g	$\{0.5, 1, 1.5, 2\}$	CFG strength at $n{>}1$ inference

F DISCUSSION

Feedback source. In this study, since NAVSIM serves as our benchmark, we use PDMS as the ground-truth feedback signal for post-hoc reflection. However, EvoAD does not depend on this metric: other signals observable in real-world driving can also serve as the post-hoc reflection signal, such as interventions, comfort, lane deviation, or progress. This signal is only used to update memory for future decisions, not to select or rank the current candidates.

Asynchronous reflection. Driving inference and post-hoc reflection operate asynchronously. During real-time inference, EvoAD selects a trajectory from the current planner proposals together with memory-guided refined candidates. After execution, EvoAD stores the executed trajectory and its feedback in memory, and later retrieves relevant experience to reflect on the decision and generate refined trajectories for future decisions.

Stability of multi-round reflection. EvoAD can prevent errors from propagating across reflection rounds. At each round, the latent world model re-evaluates the current candidate from the original observation rather than rolling forward a previously predicted future. Moreover, only experiences with improved measured feedback overwrite the stored memory. Therefore, an occasional suboptimal decision does not replace the best previous experience and can be corrected in later rounds.

Refined candidates and the evaluator. EvoAD uses PAR to refine trajectories with retrieved experience and feedback. FAE then evaluates both the refined candidates and the original planner proposals. Refined trajectories may differ from those seen during FAE training. To reduce this potential distribution shift, we keep the original proposals in the candidate pool as fallbacks. Adding PAR improves R5 PDMS from 92.76 to 93.36 (Table 7), showing that FAE can effectively select useful refined candidates.