DREAMERAD: EFFICIENT REINFORCEMENT LEARNING VIA LATENT WORLD MODEL FOR AUTONOMOUS DRIVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Training reinforcement learning (RL) policies through real-world interaction is costly and risky. Video-generation-based world models can provide realistic video rollouts in place of real-world interaction, but multi-step denoising and RGB decoding make these imagined rollouts computationally expensive. We introduce DreamerAD, a framework for efficient reinforcement learning through latent-space imagination for autonomous driving. Our approach leverages denoised latent features from video generation models through three key mechanisms: (1) shortcut forcing that distills multi-step flow updates into low-step latent rollouts, accelerating world model inference by up to $80\times$, (2) an autoregressive dense reward model that predicts step-wise rewards directly from latent representations, providing temporally dense feedback for policy optimization, and (3) Gaussian vocabulary sampling for GRPO that constrains exploration to physically plausible trajectories. DreamerAD achieves 91.4 EPDMS on NAVSIM v2 and 55.0 HD-Score on zero-shot HUGSIM, establishing state-of-the-art performance across both benchmarks. Across both the driving-specific Epona world model and the general-purpose Wan2.2 video generation backbone, reinforcement learning through latent-space imagination improves zero-shot closed-loop driving performance on HUGSIM.

Keywords: World Model, Reward Model, Reinforcement Learning

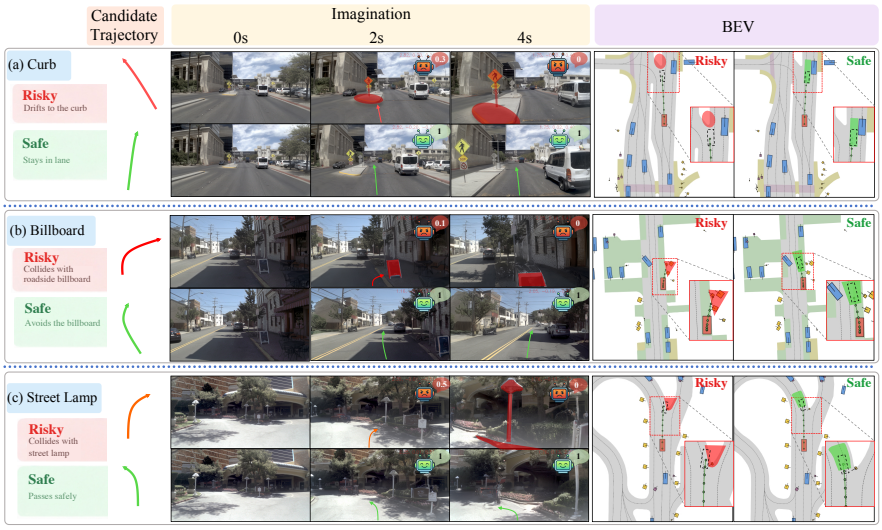

Figure 1: **Trajectory-conditioned imagination and reward evaluation in DreamerAD.** The world model imagines future states conditioned on risky (top) or safe (bottom) trajectories, while the reward model predicts step-wise rewards. Risky trajectories receive decreasing rewards, whereas safe alternatives maintain high rewards. BEV views show the corresponding outcomes.

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.

1 INTRODUCTION

Reinforcement learning (RL) is widely recognized as an effective approach for addressing long-tail problems in autonomous driving, such as distribution shift and causal confusion (Gao et al., 2025). Training RL policies through real-world interaction incurs prohibitive trial-and-error costs and unacceptable safety risks. Conventional simulator-based methods introduce additional sim-to-real discrepancies (Li et al., 2024). Video world models offer an alternative by generating realistic future scenes for policy learning without real-world interaction (Jiang et al., 2026; 2025). Using these models for RL nevertheless presents two challenges: multi-step denoising and RGB decoding incur substantial computational costs, and generating plausible future scenes does not directly provide the trajectory evaluations needed for policy optimization. Efficient imagination therefore requires both low-cost future prediction and reward modeling of the imagined outcomes.

To address these challenges, we propose DreamerAD, a framework that performs reinforcement learning through latent-space imagination. Our key insight is to leverage the denoised features from video generation models to construct a latent world model and to represent and evaluate imagined outcomes. As shown in Fig. 2, the denoised latent features from Video DiT exhibit well-structured spatial information and semantic coherence, motivating reward prediction directly in latent space. Specifically, DreamerAD predicts future latent states conditioned on candidate trajectories and evaluates their imagined outcomes with a learned reward model. For visualization, we decode the imagined latents into RGB frames in Fig. 1, showing how different candidate trajectories under the same driving context lead to distinct imagined outcomes and corresponding reward predictions.

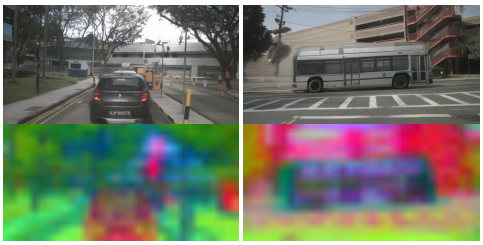

Figure 2: PCA visualization of denoised latent features, demonstrating strong spatial and semantic coherence.

Concretely, DreamerAD integrates three complementary components. First, Shortcut Forcing distills multi-step flow updates into as few as one sampling step, enabling efficient latent rollouts for RL. The predicted latents remain decodable into RGB frames for visual inspection, while RGB decoding is bypassed during RL training. Second, we learn an autoregressive dense reward model that directly evaluates trajectory-conditioned imagined latent states across future prediction horizons. By predicting step-wise driving scores, the reward model provides temporally dense feedback for policy optimization. Third, we introduce Gaussian vocabulary sampling for GRPO to generate candidate trajectories for latent imagination and reward evaluation. Complete trajectories from a predefined vocabulary are sampled according to their Gaussian likelihood around the current policy prediction. Compared with random Gaussian point sampling in prior work (Yang et al., 2026b), this approach preserves the motion structure of complete trajectories while retaining diversity through probabilistic sampling.

We instantiate DreamerAD with the pretrained driving-specific world model Epona (Zhang et al., 2025) and the general-purpose video generation backbone Wan2.2 (Wan et al., 2025), with the latter initialized from DriveVA (Liu et al., 2026). We evaluate both implementations on NAVSIM (Cao et al., 2025) and further assess their zero-shot closed-loop driving performance on HUGSIM (Zhou et al., 2025). DreamerAD accelerates world model inference by up to $80\times$ with Epona and consistently improves planning performance over both backbone-specific baselines. The Wan2.2-based implementation achieves 91.4 EPDMS on NAVSIM v2 and improves the HUGSIM HD-Score from 46.68 to 54.96 without HUGSIM-specific training. These results demonstrate the effectiveness of training policies through latent-space imagination in improving closed-loop driving performance.

Our contributions are summarized as follows:

- We propose **DreamerAD**, a framework for reinforcement learning through latent-space imagination in autonomous driving, which enables trajectory-conditioned future prediction and policy optimization directly from imagined latent representations.

- We introduce three complementary components for efficient and effective latent-space reinforcement learning: **Shortcut Forcing World Model (SF-WM)** for low-cost latent imagination, **Autoregressive Dense Reward Model (AD-RM)** for providing temporally dense feedback through step-wise reward prediction, and **Gaussian vocabulary sampling** for maintaining trajectory coherence while supporting diverse exploration under GRPO.
- Extensive experiments on NAVSIM and zero-shot closed-loop HUGSIM demonstrate consistent improvements across both Epona and Wan2.2 backbones. DreamerAD–Wan2.2 achieves **91.4 EPDMS on NAVSIM v2** and **55.0 HD-Score on HUGSIM**, establishing state-of-the-art performance and demonstrating substantial gains in zero-shot closed-loop driving capability.

2 Related Works

2.1 Autonomous Driving World Models

Vision-centric world models (Babaeizadeh et al., 2017; Ha & Schmidhuber, 2018) have gained attention due to their sensor flexibility and data accessibility. One line of work explores controllable driving video generation, including GAIA-1 (Hu et al., 2023a), DriveDreamer (Wang et al., 2024), and Epona (Zhang et al., 2025). Other approaches explore alternative representations for prediction and planning. OccWorld (Zheng et al., 2024) predicts scene evolution and ego motion in 3D occupancy space. Latent-WAM (Wang et al., 2026b) and ReasonPlan (Liu et al., 2025b) support planning through latent world-state prediction and next-scene prediction with decision reasoning, respectively. For diffusion-based video world models, multi-step sampling makes repeated imagined rollouts computationally expensive. We use shortcut forcing to enable single-step latent prediction for efficient imagination-based RL.

2.2 Reinforcement Learning in World Models

Reducing real-world trial-and-error cost has motivated RL training inside world model imagination spaces. ReSim (Yang et al., 2026a) uses trajectory-conditioned video synthesis for evaluative feedback. The Dreamer series (Hafner et al., 2019; 2020; 2023; 2025) optimizes policies through latent rollouts over multiple future time steps. In autonomous driving, model-based RL has been used to learn driving policies with predictive individual world models (Gao et al., 2024), while pseudo-simulation-based RL enables rendering-free closed-loop policy optimization (Gao et al., 2026b). Recent post-training methods also use takeover data and scenario dreaming to improve driving VLA policies (Gao et al., 2026a). RAD (Gao et al., 2025) employs 3D Gaussian Splatting world modeling, while AD-R1 (Yan et al., 2026) explores RL within occupancy-based representations. However, these methods face limitations in efficiency, annotation dependency, and underutilization of latent world model features. Our framework performs RL entirely within latent imagination space, generating temporally dense feedback for policy optimization.

3 Method

3.1 Overall

As shown in Fig. 3, DreamerAD performs reinforcement learning through trajectory-conditioned latent imagination. The RL training pipeline consists of three stages: **Policy Generation and Sampling**, **Latent Imagination and Reward Modeling**, and **Policy Optimization**. Given the observed driving history, the policy uses the backbone's flow-based trajectory-generation branch to predict a base trajectory, around which Gaussian vocabulary sampling constructs a group of candidate trajectories. For each candidate, the Shortcut Forcing World Model (SF-WM) efficiently rolls out future latent states conditioned on the sampled trajectory. The Autoregressive Dense Reward Model (AD-RM) evaluates the imagined states across future prediction horizons and produces step-wise reward predictions, which are temporally aggregated into a trajectory-level reward. Finally, the trajectory rewards are converted into group-relative advantages and used to optimize the policy with GRPO.

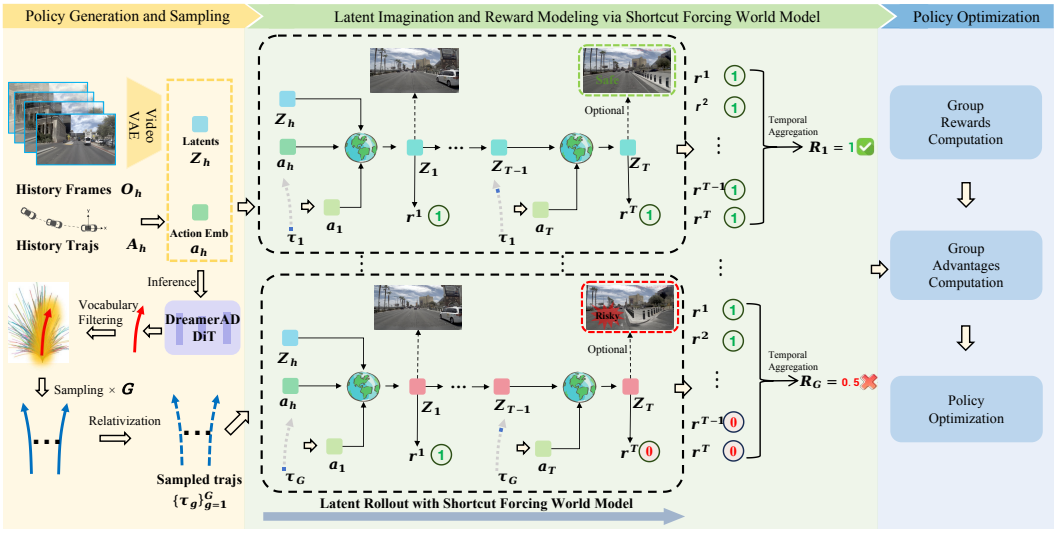

Figure 3: **Overview of the DreamerAD RL training framework.** The pipeline contains three stages: (1) **Policy Generation and Sampling (yellow)**, where the policy predicts a base trajectory and Gaussian vocabulary sampling generates candidate trajectories; (2) **Latent Imagination and Reward Modeling (green)**, where candidate trajectories are rolled out efficiently with the Shortcut Forcing World Model, step-wise reward predictions are temporally aggregated into trajectory rewards, and RGB decoding is optional and used only for visual inspection; and (3) **Policy Optimization (blue)**, where trajectory rewards are converted into group-relative advantages for GRPO-based policy optimization.

3.2 Efficient Latent Imagination with Shortcut Forcing

Shortcut Forcing World Model. Iterative flow-based generation incurs substantial computational cost when the world model repeatedly generates imagined outcomes for candidate trajectories during reinforcement learning. Inspired by shortcut models (Frans et al., 2024) and Dreamer 4 (Hafner et al., 2025), SF-WM learns to approximate two successive fine-grained flow updates with a single larger update, enabling latent prediction with as few as one sampling step.

Let x_1 denote the clean future visual latent, $x_0 \sim \mathcal{N}(0, I)$ the initial noise, and c the historical context together with trajectory conditioning. For flow time $t \in [0, 1]$, we interpolate between noise and data as $x_t = tx_1 + (1-t)x_0$, with velocity $v = \frac{dx_t}{dt} = x_1 - x_0$ serving as the standard flow-matching target. The student network $\phi_\theta(x_t, t, d; c)$ predicts the visual latent flow velocity over an integration interval of size d. During training, we sample d over multiple dyadic resolutions, allowing the same model to support different sampling budgets.

We maintain an exponential moving average (EMA) teacher $\phi_{\bar{\theta}}$. For an interval of size d, the teacher performs two consecutive half-step updates:

$$\begin{aligned} v_1^{\text{teach}} &= \phi_{\bar{\theta}}(x_t, t, d/2; c), \qquad x_{\text{mid}} = x_t + \frac{d}{2} v_1^{\text{teach}}, \\ v_2^{\text{teach}} &= \phi_{\bar{\theta}}(x_{\text{mid}}, t + d/2, d/2; c). \end{aligned} \tag{1}$$

The two half-step predictions define the supervision for a single full-step student update. Let $d_{\min}$ denote the finest training resolution. At $d = d_{\min}$, where no finer shortcut target is available, we retain the standard flow-matching target. The training target is therefore defined as

$$v_{\text{target}} = \begin{cases} v, & d = d_{\min}, \\ \text{sg}\left(\dfrac{v_1^{\text{teach}} + v_2^{\text{teach}}}{2} \right), & d > d_{\min}. \end{cases} \tag{2}$$

where sg denotes stop-gradient.

The student is optimized using

$$\mathcal{L}_{\mathrm{SF}} = \mathbb{E}\left[\left\|\phi_\theta(x_t, t, d; c) - v_{\mathrm{target}}\right\|_2^2\right], \tag{3}$$

During shortcut training, distillation is applied only to the visual latent flow, while the trajectory-generation branch retains flow-matching supervision using ground-truth trajectories. At inference, setting $d = 1/K$ yields a K-step latent prediction. Backbone-specific shortcut implementations and training settings are provided in Appendix B and Appendix C.

3.3 Autoregressive Dense Reward Modeling

Reward Annotation and Labeling. World models trained primarily on expert demonstrations may produce unreliable imagined states when conditioned on trajectories that deviate substantially from the data distribution. To reduce such out-of-distribution imagination while retaining diverse driving behaviors, we construct reward-model supervision from a spatially constrained subset of a trajectory vocabulary from GTRS (Li et al., 2025). We retain trajectories satisfying $|\Delta x| \le x_{\mathrm{th}}$, $|\Delta y| \le y_{\mathrm{th}}$, and $\Delta\theta \le \theta_{\mathrm{th}}$, where Δx and Δy denote longitudinal and lateral endpoint deviations from the expert trajectory, and $\Delta\theta$ is the smallest absolute heading difference at the endpoint.

To preserve diversity in lateral driving behaviors, the retained trajectories are sorted according to $|\Delta y|$, from which M candidates are selected at evenly spaced indices. Each selected trajectory is simulated and scored in NAVSIM to obtain supervision for 8 driving metrics:

$$y^j = \left\{y^j_{\mathrm{nc}}, y^j_{\mathrm{dac}}, y^j_{\mathrm{ddc}}, y^j_{\mathrm{tlc}}, y^j_{\mathrm{ep}}, y^j_{\mathrm{ttc}}, y^j_{\mathrm{lk}}, y^j_{\mathrm{hc}}\right\}. \tag{4}$$

These metrics cover safety, traffic-rule compliance, progress, and comfort; their definitions are provided in Appendix A.

Instead of assigning only a single reward to the complete trajectory, we evaluate each candidate at T future prediction steps separated by Δt. Here, $j \in \{1, \ldots, T\}$ denotes the future step, with step j corresponding to physical prediction time $j\Delta t$. The resulting step-wise labels provide temporally resolved supervision for learning how driving quality evolves along the imagined trajectory, enabling dense reward prediction for subsequent reinforcement learning.

Autoregressive Dense Reward Prediction. Given a candidate trajectory τ, the world model predicts its corresponding imagined future latent states. At future step j, AD-RM predicts driving scores from the observed history, candidate trajectory, and imagined visual context. Epona uses step-wise latent prefixes, while Wan2.2 uses context organized into temporally compressed blocks. The following equations describe Epona's step-wise formulation; the block-wise context and reward queries used by Wan2.2 are detailed in Appendix C.3.

$$H_j = \mathrm{Compress}\left([z_{1-N_h}, \ldots, z_0, \hat{z}_1, \ldots, \hat{z}_j]\right), \tag{5}$$

where $H_j, z, \hat{z}$ denote the latent context at future step j, the observed latent history, and the imagined future latents, respectively. Here, N_h denotes the number of observed latent states, including the current state. The query-based compressor reduces the latent representation to a compact set of context tokens for reward prediction.

$$Q_j = Q_{\mathrm{base}} + \mathrm{MLP}_{\mathrm{traj}}(\tau_{1:j}) + \mathrm{Emb}_{\mathrm{step}}(j). \tag{6}$$

Here, Q_{base} distinguishes the 8 driving dimensions, $\mathrm{MLP}_{\mathrm{traj}}(\tau_{1:j})$ encodes the candidate trajectory prefix up to step j, and $\mathrm{Emb}_{\mathrm{step}}(j)$ identifies the corresponding future step.

$$\mathbf{p}^j = \mathrm{sigmoid}\left(\mathrm{MLP}\left(\mathrm{CrossAttention}(Q_j, H_j)\right)\right) \in [0,1]^8, \tag{7}$$

where $\mathbf{p}^j$ represents the predicted reward scores for the 8 driving dimensions at future step j.

$$\mathcal{L}_{\mathrm{reward}} = \sum_{j=1}^{T} \gamma_j \sum_{k=1}^{8} \omega_k \,\mathrm{BCE}\left(p^j_k, y^j_k\right), \tag{8}$$

where k indexes the 8 reward dimensions, and γ_j and ω_k denote the temporal and reward-dimension weighting factors, respectively. The resulting step-wise predictions $\{\mathbf{p}^j\}_{j=1}^T$ provide dense reward signals for the reinforcement learning procedure described in Sec. 3.4.

3.4 Reinforcement Learning with Gaussian Vocabulary Sampling

Reward Design for RL. AD-RM provides 8 driving scores $\mathbf{p}^j \in [0,1]^8$ at each future step j. Following the PDM-style aggregation in NAVSIM v2, we treat $\mathcal{S} = \{\mathrm{NC}, \mathrm{DAC}, \mathrm{DDC}, \mathrm{TLC}\}$ as safety-compliance metrics and $\mathcal{P} = \{\mathrm{EP}, \mathrm{TTC}, \mathrm{LK}, \mathrm{HC}\}$ as task-performance metrics. For each sampled trajectory, the step-wise reward and its temporal aggregation are

$$r^j = \log\left[\prod_{k\in\mathcal{S}} (p_k^j)^{w_k} \cdot \sum_{k\in\mathcal{P}} w_k p_k^j\right], \qquad R = \sum_{j=1}^{T} \lambda_j r^j. \tag{9}$$

Here, w_k and λ_j denote the metric and temporal weights, respectively. The multiplicative safety terms assign a strong penalty to low safety-compliance scores, while temporal aggregation retains the dense step-wise feedback provided by AD-RM.

Gaussian Vocabulary Sampling. To keep exploration within plausible motion patterns, we sample complete trajectories from the spatially constrained vocabulary subset described in Sec. 3.3, rather than perturbing individual waypoints directly. Let τ_{act} denote the current policy prediction and $\Gamma = \{\gamma_n\}_{n=1}^N$ this filtered subset. We assign each vocabulary trajectory a sampling probability according to its Gaussian likelihood around τ_{act}:

$$P_n = \mathrm{softmax}_n\left(-\frac{\|\gamma_n - \tau_{\mathrm{act}}\|_2^2}{2\sigma^2}\right), \qquad n = 1, \ldots, N. \tag{10}$$

Here, σ controls the exploration range, and the distance is computed over the complete trajectory. We then sample G candidate trajectories according to $\{P_n\}_{n=1}^N$ ($G = 8$ by default) and evaluate them through trajectory-conditioned latent rollouts and AD-RM. This standard probabilistic sampling favors trajectories near the current policy prediction while preserving the smooth motion structure encoded by the vocabulary.

Policy Optimization. Given the trajectory rewards $\{R_i\}_{i=1}^G$, GRPO normalizes them within each sampled group to obtain relative advantages. Since candidate trajectories are sampled from a discrete vocabulary, we use a Gaussian surrogate likelihood $\tilde{\pi}_\theta(\tau_i) = \mathcal{N}(\tau_i; \tau_{\mathrm{act}}, \sigma^2 I)$ centered at the current policy prediction. The group advantage and policy ratio are

$$A_i = \frac{R_i - \mathrm{mean}(R_{1:G})}{\mathrm{std}(R_{1:G})}, \qquad \rho_i = \frac{\tilde{\pi}_\theta(\tau_i)}{\tilde{\pi}_{\mathrm{old}}(\tau_i)}. \tag{11}$$

Here, ρ_i serves as a surrogate policy-update ratio rather than the exact importance ratio of the discrete vocabulary-sampling process. The policy is optimized with the clipped GRPO objective together with behavioral cloning:

$$\mathcal{L}_{\mathrm{total}} = -\mathbb{E}_i[\min(\rho_i A_i, \mathrm{clip}(\rho_i, 1-\epsilon, 1+\epsilon)A_i)] + \lambda_{\mathrm{bc}}\|\tau_{\mathrm{act}} - \tau_{\mathrm{gt}}\|_1. \tag{12}$$

The behavioral cloning term preserves the supervised driving prior during reinforcement learning.

4 Experiment

4.1 Experimental Setup

We evaluate DreamerAD on NAVSIM v1 (Dauner et al., 2024) and NAVSIM v2 (Cao et al., 2025), reporting PDMS and EPDMS, respectively. We instantiate DreamerAD on two different world-model backbones. Epona (Zhang et al., 2025) is a driving-specific autoregressive world model. We further apply the complete framework to Wan2.2 (Wan et al., 2025), initialized from DriveVA (Liu et al., 2026), to evaluate whether DreamerAD generalizes to a different large-scale video world-model backbone. To evaluate closed-loop driving capability beyond NAVSIM, we further conduct zero-shot evaluation on HUGSIM (Zhou et al., 2025), a photo-realistic interactive driving simulator, without HUGSIM-specific training or fine-tuning. Dataset descriptions and metric definitions are provided in Appendix A. The implementation details of Epona and Wan2.2 are provided in Appendices B and C, respectively.

4.2 Main Results

Table 1: **NAVSIM v2 (Cao et al., 2025) results with extended metrics.** + and - denote improvement/degradation relative to the corresponding base model. † The Wan2.2 baseline is initialized from DriveVA.

Method	Venue	NC ↑	DAC ↑	DDC ↑	TLC ↑	EP ↑	TTC ↑	LK ↑	HC ↑	EC ↑	EPDMS ↑
ReCogDrive (Li et al., 2026b)	ICLR 2026	98.3	95.2	99.5	99.8	87.1	97.5	96.6	98.3	86.5	83.6
MeanFuser (Wang et al., 2026a)	CVPR 2026	98.3	97.2	99.6	99.8	87.6	97.4	97.3	98.3	**88.2**	89.5
DriveFine (Dang et al., 2026)	ECCV 2026	98.7	97.3	99.5	99.8	88.7	97.8	97.7	**98.4**	83.8	89.7
SparseDriveV2 (Sun et al., 2026)	ECCV 2026	98.1	98.1	99.6	99.8	**91.1**	97.3	96.9	98.2	78.4	90.1
WM Methods											
World4Drive (Zheng et al., 2025)	ICCV 2025	97.8	96.3	99.4	99.8	88.3	97.1	97.7	98.0	53.9	84.8
WorldRFT (Yang et al., 2026b)	AAAI 2026	97.8	96.5	99.5	99.8	88.5	97.0	97.4	98.1	69.1	86.7
DriveVLA-W0 (Li et al., 2026a)	ICLR 2026	98.5	**99.1**	98.0	99.7	86.4	98.1	93.2	97.9	58.9	86.1
DriveWorld-VLA (Jia et al., 2026)	ICML 2026	98.6	**99.1**	99.6	99.8	87.4	97.9	97.0	97.8	78.6	86.8
Latent-WAM (Wang et al., 2026b)	ACMMM 2026	98.1	97.3	99.6	99.8	87.7	97.3	97.6	98.1	87.3	89.3
Epona (Base) (Zhang et al., 2025)	ICCV 2025	97.1	95.7	99.3	99.7	88.6	96.3	97.0	98.0	67.8	85.1
DreamerAD–Epona	–	98.0$_{+0.9}$	97.2$_{+1.5}$	99.5$_{+0.2}$	99.8$_{+0.1}$	87.9$_{-0.7}$	97.4$_{+1.1}$	97.4$_{+0.4}$	98.2$_{+0.2}$	73.5$_{+5.7}$	87.8$_{+2.7}$
Wan2.2 (Base)† (Liu et al., 2026)	ECCV 2026	99.0	97.6	99.7	99.8	87.5	98.6	97.8	98.0	74.8	89.1
DreamerAD–Wan2.2	–	**99.6**$_{+0.6}$	98.7$_{+1.1}$	**99.8**$_{+0.1}$	**99.9**$_{+0.1}$	87.6$_{+0.1}$	**99.2**$_{+0.6}$	**98.5**$_{+0.7}$	98.3$_{+0.3}$	79.2$_{+4.4}$	**91.4**$_{+2.3}$

As shown in Tables 1 and 2, DreamerAD consistently improves both world-model backbones across NAVSIM v1 and v2. On NAVSIM v2, DreamerAD improves Epona from 85.1 to 87.8 EPDMS (+2.7) and Wan2.2 from 89.1 to 91.4 (+2.3). On NAVSIM v1, the corresponding PDMS scores improve from 86.2 to 88.8 (+2.6) and from 90.2 to 92.3 (+2.1). DreamerAD–Wan2.2 achieves 91.4 EPDMS on NAVSIM v2 and 92.3 PDMS on NAVSIM v1, establishing state-of-the-art performance among existing world-model-based methods. The higher absolute scores of the Wan2.2-based model may partly reflect the greater representational capacity afforded by its larger backbone (5B parameters) and large-scale video pretraining. These consistent improvements demonstrate that DreamerAD is compatible with different world-model backbones.

Table 2: **NAVSIM v1 (Dauner et al., 2024) results.** † The Wan2.2 baseline is initialized from DriveVA.

Method	NC↑	DAC↑	TTC↑	Comf.↑	EP↑	PDMS↑
SparseDriveV2 (Sun et al., 2026)	98.5	98.4	95.0	99.9	**88.6**	92.0
ReCogDrive (Li et al., 2026b)	97.9	97.3	94.9	**100.0**	87.3	90.8
DriveFine (Dang et al., 2026)	98.8	98.6	96.2	**100.0**	86.9	91.8
WM Methods						
DriveVLA-W0 (Li et al., 2026a)	98.7	96.2	95.5	**100.0**	82.2	88.4
DriveWorld-VLA (Jia et al., 2026)	99.1	98.2	96.1	**100.0**	85.9	91.3
Epona (Base) (Zhang et al., 2025)	97.9	95.1	93.8	99.9	80.4	86.2
DreamerAD–Epona	98.0	97.2	94.2	**100.0**	83.2	88.8
Wan2.2 (Base)† (Liu et al., 2026)	99.2	97.7	96.2	**100.0**	83.9	90.2
DreamerAD–Wan2.2	**99.6**	**98.7**	**97.9**	**100.0**	85.6	**92.3**

DreamerAD also improves key safety and driving-quality metrics on NAVSIM v2. With Epona, NC, DAC, and TTC increase by 0.9, 1.5, and 1.1 points, respectively; with Wan2.2, they improve by 0.6, 1.1, and 0.6 points. Extended Comfort (EC) measures motion consistency between predicted trajectories across frames. EC increases from 67.8 to 73.5 for Epona and from 74.8 to 79.2 for Wan2.2.

4.3 Zero-shot Closed-loop Evaluation on HUGSIM

As shown in Table 3, DreamerAD–Wan2.2 achieves a state-of-the-art HD-Score of **54.96** with an average route completion of **66.87**. More importantly, DreamerAD consistently improves both backbones: the average HD-Score increases from 29.71 to 32.40 (+2.69) with Epona and from 46.68 to 54.96 (+8.28) with Wan2.2. The gains remain evident across different scenario difficulties. For DreamerAD–Wan2.2, HD-Score improves by 8.64, 9.96, 10.68, and 3.20 points on the Easy, Medium, Hard, and Extreme subsets, respectively. Together with the NAVSIM results, the zero-shot improvements on HUGSIM show that latent-space reinforcement learning improves closed-loop driving performance beyond the benchmark used for training.

4.4 Ablation Studies

4.4.1 Impact of Shortcut Forcing Method

Comparing IDs 4 and 6 in Table 4 isolates the contribution of SF-WM while keeping AD-RM and vocabulary sampling unchanged. ID 4 applies one-step denoising to the world model without shortcut training. SF-WM improves EPDMS from 86.4 to 87.8 (+1.4) with Epona and from 90.5 to 91.4 (+0.9) with Wan2.2. The consistent gains across both backbones show that aggressive

Table 3: **Zero-shot closed-loop results on HUGSIM (Zhou et al., 2025).** E/M/H/X denote Easy/Medium/Hard/Extreme. Bold indicates the best result in each column.

Method	RC ↑					HD-Score ↑				
	E	M	H	X	Avg.	E	M	H	X	Avg.
LTF (Chitta et al., 2023)	67.80	35.10	26.20	**40.50**	40.42	58.90	18.00	9.80	**25.90**	25.57
UniAD (Hu et al., 2023b)	78.40	60.50	33.60	17.80	47.77	64.90	45.80	20.60	6.60	34.50
VAD (Jiang et al., 2023)	51.30	31.10	25.30	26.50	32.44	36.30	9.50	8.00	11.50	14.56
GTRS-Dense (Li et al., 2025)	64.20	50.00	20.70	22.30	39.61	55.50	39.00	11.70	14.30	30.18
Latent-WAM (Wang et al., 2026b)	84.20	42.50	30.60	35.50	45.88	72.50	24.00	12.20	18.10	28.91
Epona (Base)	70.70	49.95	27.60	21.48	42.11	59.96	38.30	14.02	7.76	29.71
DreamerAD–Epona	77.94	49.84	33.02	21.33	44.56	68.22	38.25	19.45	7.74	32.40
Wan2.2 (Base)	89.54	75.49	47.81	26.40	60.38	74.59	62.28	34.02	13.01	46.68
DreamerAD–Wan2.2	**95.51**	**83.72**	**56.89**	28.22	**66.87**	**83.23**	**72.24**	**44.70**	16.21	**54.96**

Table 4: Ablation study of DreamerAD components. SF-WM denotes Shortcut Forcing World Model and AD-RM denotes Autoregressive Dense Reward Model. D-E and D-W denote DreamerAD–Epona and DreamerAD–Wan2.2, respectively. Final-only replaces AD-RM with a learned reward model that predicts only the full-trajectory reward at the final horizon. One-step world-model rollout is used by default during RL training.

ID	SF-WM	Reward	Sampling	EPDMS ↑	
				D-E	D-W
1		–	–	85.1	89.1
2	√	AD-RM	WorldRFT	86.6	89.5
3	√	AD-RM	Flow-GRPO	87.0	90.1
4		AD-RM	Vocab	86.4	90.5
5	√	Final-only	Vocab	87.0	90.8
6	√	AD-RM	Vocab	**87.8**	**91.4**

sampling-step compression preserves latent representations that remain effective for downstream reinforcement learning.

Table 5 evaluates SF-WM rollout efficiency during RL training. At 100 steps, per-frame latency reaches 2359.2 ms for Epona and 2027.1 ms for Wan2.2, making RL training prohibitively costly in this setting. Reducing the rollout from 16 steps to one lowers latency from 403.3 to 30.2 ms for Epona and from 320.0 to 20.0 ms for Wan2.2, corresponding to approximately $13.4\times$ and $16\times$ speedups, respectively. EPDMS remains at 87.8 for Epona and decreases only from 91.5 to 91.4 for Wan2.2, supporting one-step rollout as the default setting for RL training.

Table 5: Effect of SF-WM sampling steps during RL training. EPDMS is evaluated on policies trained with the corresponding SF-WM rollout step count. – indicates that RL training was not performed due to prohibitively high rollout latency. ∗Per-frame latency is averaged over Wan2.2's 8-frame prediction.

Steps	Rollout Latency / Frame (ms) ↓		EPDMS ↑	
	D-E	D-W*	D-E	D-W
100	2359.2	2027.1	–	–
16	403.3	320.0	87.8	91.5
4	102.0	80.0	**87.9**	**91.6**
1	**30.2**	**20.0**	87.8	91.4

Figure 4 qualitatively shows the effect of shortcut forcing under one-step inference. Direct one-step inference with Epona exhibits progressive error accumulation and blurring, whereas SF-WM preserves clearer scene structure throughout the autoregressive rollout.



(a) Epona: progressive blurring under one-step inference.



(b) SF-WM: visual structure is preserved under one-step inference.

Figure 4: One-step autoregressive prediction with Epona and SF-WM.

4.4.2 Effectiveness of Autoregressive Dense Reward Modeling

Comparing IDs 5 and 6 isolates the effect of dense temporal reward modeling while keeping SF-WM and vocabulary sampling unchanged. Replacing the final-only trajectory reward with AD-RM improves EPDMS from 87.0 to 87.8 (+0.8) with Epona and from 90.8 to 91.4 (+0.6) with Wan2.2. These results support the effectiveness of step-wise reward prediction over final-only reward prediction for downstream policy optimization.

4.4.3 Effectiveness of Vocab Sampling Method

IDs 2, 3, and 6 in Table 4 compare different trajectory sampling strategies under the same SF-WM, AD-RM, and RL optimization configuration. WorldRFT (Yang et al., 2026b) and Flow-GRPO (Liu et al., 2025a) perform on-policy trajectory exploration, whereas our method samples candidates from a predefined trajectory vocabulary guided by the current policy prediction. Vocabulary sampling achieves 87.8 EPDMS with Epona, outperforming WorldRFT and Flow-GRPO by 1.2 and 0.8 points, respectively. The same trend holds with Wan2.2, where it achieves 91.4 EPDMS, compared with 89.5 and 90.1. These results indicate that vocabulary-constrained sampling provides more effective exploration than the alternative sampling strategies, while preserving dynamically plausible trajectory candidates.

4.5 Qualitative Results

As shown in Figure 5, the SFT trajectories collide with preceding vehicles in rows 1, 2, and 4, while the RL trajectories avoid these collisions. In row 3, the RL policy adjusts its trajectory to avoid the obstacle encountered by the SFT trajectory. These examples illustrate safer planned trajectories in the selected scenarios, consistent with the quantitative improvements in NC, DAC, and TTC.

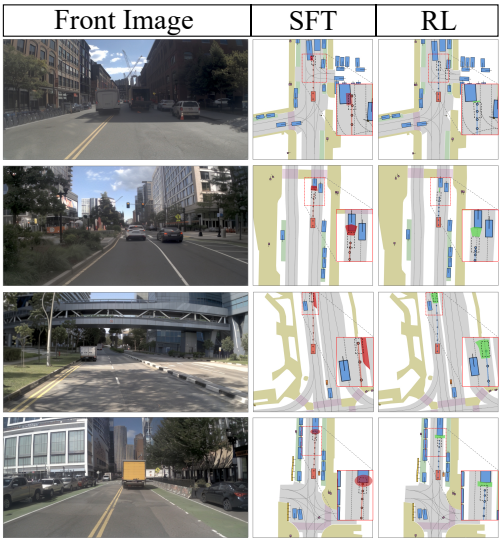

Figure 5: **Qualitative comparison of SFT and RL with the Epona backbone.** Columns show the front-view image and BEV trajectories from SFT (red) and RL (blue). Red and green highlights indicate collisions and safe passage, respectively.

5 Conclusion

We present DreamerAD, a framework for reinforcement learning through latent-space imagination that combines efficient world-model rollout, autoregressive dense reward modeling, and Gaussian vocabulary sampling. DreamerAD consistently improves Epona and Wan2.2, with DreamerAD–Wan2.2 achieving a state-of-the-art 91.4 EPDMS on NAVSIM v2 and 55.0 HD-Score on zero-shot HUGSIM. These results demonstrate that reinforcement learning through latent-space imagination generalizes across different world-model backbones and enhances closed-loop driving capability.

AI USE STATEMENT

We used generative AI tools to improve the language and readability of the manuscript and to assist with code organization. The authors take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Evaluation protocols and metric definitions are provided in Appendix A. Implementation details for the Epona- and Wan2.2-based models are provided in Appendices B and C, respectively. Additional experimental analyses are provided in Appendix D.

REFERENCES

Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*, 2017.

Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Pseudo-simulation for autonomous driving. In *Conference on Robot Learning (CoRL)*, 2025.

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, 2023. doi: 10.1109/TPAMI.2022.3200245.

Chenxu Dang, Sining Ang, Yongkang Li, Haochen Tian, Jie Wang, Guang Li, Hangjun Ye, Jie Ma, Long Chen, and Yan Wang. DriveFine: Refining-Augmented Masked Diffusion VLA for Precise and Robust Driving. *arXiv preprint arXiv:2602.14577*, 2026. URL https://arxiv.org/abs/2602.14577.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

H. Gao, S. Chen, B. Jiang, B. Liao, Y. Shi, X. Guo, Y. Pu, H. Yin, X. Li, X. Zhang, Y. Zhang, W. Liu, Q. Zhang, and X. Wang. Rad: Training an end-to-end driving policy via large-scale 3dgs-based reinforcement learning. *arXiv preprint arXiv:2502.13144*, 2025.

Yinfeng Gao, Qichao Zhang, Da-Wei Ding, and Dongbin Zhao. Dream to drive with predictive individual world model. *IEEE Transactions on Intelligent Vehicles*, 9(12):8224–8238, 2024. doi: 10.1109/TIV.2024.3408830.

Yinfeng Gao, Deqing Liu, Qichao Zhang, Yupeng Zheng, Haochen Tian, Guang Li, Hangjun Ye, Long Chen, Da-Wei Ding, and Dongbin Zhao. Learning from mistakes: Post-training for driving vla with takeover data. *arXiv preprint arXiv:2603.14972*, 2026a.

Yinfeng Gao, Qichao Zhang, Deqing Liu, Zhongpu Xia, Guang Li, Kun Ma, Guang Chen, Hangjun Ye, Long Chen, Da-Wei Ding, and Dongbin Zhao. Perlad: Towards enhanced closed-loop end-to-end autonomous driving with pseudo-simulation-based reinforcement learning. *IEEE Robotics and Automation Letters*, 11(5):5821–5828, 2026b. doi: 10.1109/LRA.2026.3675928.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

A. Hu, L. Russell, H. Yeo, Z. Murez, G. Fedoseev, A. Kendall, J. Shotton, and G. Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023a.

Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, L. Lu, X. Jia, Q. Liu, J. Dai, Y. Qiao, and H. Li. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17853–17862, 2023b.

Feiyang Jia, Lin Liu, Ziying Song, Caiyan Jia, Hangjun Ye, Xiaoshuai Hao, and Long Chen. DriveWorld-VLA: Unified Latent-Space World Modeling with Vision-Language-Action for Autonomous Driving. *arXiv preprint arXiv:2602.06521*, 2026. URL https://arxiv.org/abs/2602.06521.

B. Jiang, S. Chen, Q. Xu, Be. Liao, J. Chen, H. Zhou, Q. Zhang, W. Liu, C. Huang, and X. Wang. Vad: Vectorized scene representation for efficient autonomous driving. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Zhennan Jiang, Kai Liu, Yuxin Qin, Shuai Tian, Yupeng Zheng, Mingcai Zhou, Chao Yu, Haoran Li, and Dongbin Zhao. World4rl: Diffusion world models for policy refinement with reinforcement learning for robotic manipulation. *arXiv preprint arXiv:2509.19080*, 2025.

Zhennan Jiang, Shangqing Zhou, Yutong Jiang, Zefang Huang, Mingjie Wei, Yuhui Chen, Tianxing Zhou, Zhen Guo, Hao Lin, Quanlu Zhang, et al. Wovr: World models as reliable simulators for post-training vla policies with rl. *arXiv preprint arXiv:2602.13977*, 2026.

Qifeng Li, Xiaosong Jia, Shaobo Wang, and Junchi Yan. Think2drive: Efficient reinforcement learning by thinking with latent world model for autonomous driving (in carla-v2). In *European conference on computer vision*, pp. 142–158. Springer, 2024.

Yingyan Li, Shuyao Shang, Weisong Liu, Bing Zhan, Haochen Wang, Yuqi Wang, Yuntao Chen, Xiaoman Wang, Yasong An, Chufeng Tang, et al. Drivevla-w0: World models amplify data scaling law in autonomous driving. In *International Conference on Learning Representations*, volume 2026, pp. 7890–7911, 2026a.

Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, BING WANG, Kun Ma, Guang Chen, Hangjun Ye, Wenyu Liu, and Xinggang Wang. Recogdrive: A reinforced cognitive framework for end-to-end autonomous driving. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=JoXwhGbuMi.

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. *arXiv preprint arXiv:2506.06664*, 2025.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025a.

Mengmeng Liu, Diankun Zhang, Jiuming Liu, Jianfeng Cui, Hongwei Xie, Guang Chen, Hangjun Ye, Michael Ying Yang, Francesco Nex, and Hao Cheng. Driveva: Video action models are zero-shot drivers. *arXiv preprint arXiv:2604.04198*, 2026.

Xueyi Liu, Zuodong Zhong, Junli Wang, Yuxin Guo, Zhiguo Su, Qichao Zhang, Yun-Fu Liu, Yinfeng Gao, Yupeng Zheng, Qiao Lin, Huiyong Chen, and Dongbin Zhao. Reasonplan: Unified scene prediction and decision reasoning for closed-loop autonomous driving. In *9th Conference on Robot Learning*, 2025b.

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. *arXiv preprint arXiv:2603.29163*, 2026. URL https://arxiv.org/abs/2603.29163.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Junli Wang, Yinan Zheng, Xueyi Liu, Zebin Xing, Pengfei Li, Kun Ma, Hangjun Ye, Guang Chen, Guang Li, Long Chen, et al. Meanfuser: Fast one-step multi-modal trajectory generation and adaptive reconstruction via meanflow for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17884–17893, 2026a.

Linbo Wang, Yupeng Zheng, Qiang Chen, Shiwei Li, Yichen Zhang, Zebin Xing, Qichao Zhang, Xiang Li, Deheng Qian, Pengxuan Yang, et al. Latent-wam: Latent world action modeling for end-to-end autonomous driving. *arXiv preprint arXiv:2603.24581*, 2026b.

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, 2024.

Tianyi Yan, Tao Tang, Xingtai Gui, Yongkang Li, Jiasen Zheng, Weiyao Huang, Lingdong Kong, Wencheng Han, Xia Zhou, Xueyang Zhang, Yifei Zhan, Kun Zhan, Cheng-zhong Xu, and Jianbing Shen. Ad-r1: Closed-loop reinforcement learning for end-to-end autonomous driving with impartial world models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 1085–1095, June 2026.

Jiazhi Yang, Kashyap Chitta, Shenyuan Gao, Long Chen, Yuqian Shao, Xiaosong Jia, Hongyang Li, Andreas Geiger, Xiangyu Yue, and Li Chen. Resim: Reliable world simulation for autonomous driving. *Advances in Neural Information Processing Systems*, 38:167710–167741, 2026a.

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, 2026b.

Kaiwen Zhang, Zhenyu Tang, Xiaotao Hu, Xingang Pan, Xiaoyang Guo, Yuan Liu, Jingwei Huang, Li Yuan, Qian Zhang, Xiao-Xiao Long, et al. Epona: Autoregressive diffusion world model for autonomous driving. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 27220–27230. IEEE, 2025.

W. Zheng, W. Chen, Y. Huang, B. Zhang, Y. Duan, and J. Lu. Occworld: Learning a 3d occupancy world model for autonomous driving. In *Proceedings of the European Conference on Computer Vision*, pp. 55–72. Springer, 2024.

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.

Hongyu Zhou, Longzhong Lin, Jiabao Wang, Yichong Lu, Dongfeng Bai, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. Hugsim: A real-time, photo-realistic and closed-loop simulator for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

A Dataset and Evaluation Metrics

A.1 NAVSIM

We evaluate planning performance on NAVSIM v1 and v2 (Dauner et al., 2024; Cao et al., 2025). Built upon OpenScene, NAVSIM contains 1,192 training sequences and 136 test sequences with more than 100,000 keyframes. The planner predicts 8 future waypoints at 2 Hz over a 4-second horizon.

NAVSIM v1 adopts the PDM Score (PDMS), which evaluates *no at-fault collision* (NC), *drivable area compliance* (DAC), *ego progress* (EP), *time-to-collision* (TTC), and *comfort* (Comf.):

$$\mathrm{PDMS} = \mathrm{NC} \cdot \mathrm{DAC} \cdot \frac{5\,\mathrm{EP} + 5\,\mathrm{TTC} + 2\,\mathrm{Comf.}}{12}. \tag{13}$$

NC and DAC are multiplicative safety factors; EP, TTC, and Comf. measure progress, collision risk, and motion smoothness.

NAVSIM v2 extends the evaluation with *driving direction compliance* (DDC), *traffic light compliance* (TLC), *lane keeping* (LK), *history comfort* (HC), and *extended comfort* (EC). The resulting Extended PDM Score (EPDMS) is

$$\mathrm{EPDMS} = \mathrm{NC} \cdot \mathrm{DAC} \cdot \mathrm{DDC} \cdot \mathrm{TLC} \cdot \frac{5\,\mathrm{EP} + 5\,\mathrm{TTC} + 2(\mathrm{LK} + \mathrm{HC} + \mathrm{EC})}{16}. \tag{14}$$

EPDMS additionally captures traffic-rule compliance, lane keeping, and temporal driving consistency.

A.2 HUGSIM

We use the 436-scenario HUGSIM benchmark (Zhou et al., 2025) for zero-shot closed-loop evaluation of the baseline and DreamerAD policies, without HUGSIM-specific training or fine-tuning. HUGSIM executes ego motion and updates subsequent observations in a photo-realistic simulator, enabling interactive closed-loop evaluation.

HUGSIM evaluates trajectory quality using *no collision* (NC), *drivable area compliance* (DAC), *time-to-collision* (TTC), and *comfort* (C). The frame-level PDM-style score is

$$s_t^{\mathrm{PDM}} = \mathrm{NC}_t \cdot \mathrm{DAC}_t \cdot \frac{5\,\mathrm{TTC}_t + 2\,\mathrm{C}_t}{7}, \tag{15}$$

which is averaged over the evaluated keyframes to obtain the scenario PDMS. HUGSIM further measures *route completion* (RC), and combines trajectory quality with closed-loop progress as

$$\text{HD-Score} = \mathrm{PDMS} \cdot \mathrm{RC}. \tag{16}$$

The equations use normalized scores; reported HUGSIM metrics are scaled to 0–100. HD-Score is computed per scenario before aggregation. Difficulty-wise analysis is provided in Appendix D.2.

B Epona Implementation Details

B.1 Input Representation and Backbone Architecture

Input and encoding. The Epona-based implementation takes four observed RGB frames, including the current frame, together with the corresponding ego-motion history. Images are sampled at 2 Hz and resized to 512×1024. The DCAE encoder compresses each image into a 16×32 latent feature map with 32 channels. Flattening its spatial dimensions gives $Z \in \mathbb{R}^{B \times P \times L \times C}$, where B is the batch size, $P = 4$, $L = 512$, and $C = 32$. Historical ego motion is represented by relative planar displacements and heading changes. The three motion components are encoded separately using Fourier features followed by MLP projectors, producing three motion tokens per observed frame. A visual projector maps the image latents to the same $D = 2048$ dimensional feature space.

Spatial and temporal fusion. For each observed frame, the 512 projected visual tokens are concatenated with the three motion tokens. Learned temporal embeddings identify their positions in the observation sequence. The resulting representation $E \in \mathbb{R}^{B\times P\times(L+3)\times D}$ is processed by a spatial-temporal Transformer (STT). Each STT block applies causal attention across time and spatial attention among the tokens within each frame, allowing visual observations and ego motion to be integrated over the observed history. The output tokens at the latest observed time form the context $F \in \mathbb{R}^{B\times(L+3)\times D}$ for subsequent generation.

Visual and trajectory generation. The fused context conditions two flow-based diffusion transformers: a visual generation branch (VisDiT) and a trajectory generation branch (TrajDiT). VisDiT additionally receives an embedding of the ego motion for the next transition and transforms Gaussian latent noise into the next-frame prediction $\hat{z}_1 \in \mathbb{R}^{B\times L\times C}$. This motion condition allows the same observed scene to produce different imagined outcomes under different candidate actions. TrajDiT generates a future ego trajectory $\hat{\tau} \in \mathbb{R}^{B\times T\times 3}$, where each waypoint contains planar position and heading. We use $T = 8$ waypoints at 0.5-second intervals, covering a four-second planning horizon. The predicted visual latent can be passed directly to subsequent latent-space computation; RGB decoding is used when visualizing the imagined outcome.

B.2 Trajectory-Conditioned Latent Rollout

Given a candidate trajectory, DreamerAD evaluates its imagined consequences by recursively predicting future visual latents. All candidates for a scene start from the same observed latent history and ego-motion context. At future step j, the motion associated with the corresponding trajectory transition is encoded and supplied to VisDiT, together with the context produced by STT. The candidate trajectory remains fixed throughout its evaluation, so each rollout follows the motion specified by that candidate.

VisDiT predicts the next visual latent $\hat{z}_j$. This prediction is appended to the visual context, and the oldest latent is removed to maintain a four-frame sliding window. The ego-motion history is updated with the corresponding candidate motion. STT then processes the updated context for the next prediction. Repeating this procedure produces the imagined latent sequence $\{\hat{z}_1, \ldots, \hat{z}_T\}$. After the first prediction, the context therefore progressively incorporates imagined states.

Shortcut forcing accelerates the visual prediction within each rollout step. The visual generator is conditioned on both flow time and integration step size, as described in Section 3.2. We use one sampling step per predicted latent by default during reward-model and RL training. The trajectory generation branch retains its separate 20-step sampling procedure. The imagined latents and candidate trajectory information are passed to AD-RM for step-wise reward prediction, without decoding the latents into RGB images.

B.3 Dense Reward Network and Feature Fusion

Input features and compression. At future step j, AD-RM receives the observed and imagined visual latents available up to that step, the candidate trajectory prefix $\tau_{1:j}$, and the step index. An MLP projector maps the 32-dimensional visual latents into a reward feature space of dimension $D_r = 1024$. For each observed or imagined frame, 32 learned queries attend to its 512 projected visual tokens, producing a compact frame representation. These representations are retained as the rollout progresses, and each newly imagined frame contributes another 32 tokens. The accumulated visual tokens provide context for reward prediction.

Trajectory-conditioned reward queries. AD-RM maintains 8 learned queries, one for each driving metric. At step j, the trajectory prefix is zero-padded to the full planning length and encoded by an MLP. Its embedding and a learned step embedding are added to each metric query. The resulting queries therefore identify both the metric to predict and the trajectory prefix being evaluated. Only the prefix through step j is supplied to the trajectory encoder; subsequent waypoints are excluded from this input.

Feature fusion and reward outputs. A three-layer Transformer decoder with 8 attention heads fuses the reward queries with the feature memory. Each layer updates the queries through self-

attention among the metric queries and cross-attention to the memory. An independent MLP head maps each resulting query to a scalar reward logit, which is converted into a score using sigmoid. Each rollout step produces 8 scores, and repeating this process over the 8 future steps yields a $B \times 8 \times 8$ reward tensor, with the last two dimensions indexing prediction steps and driving metrics. The scores provide the step-wise signals used for trajectory reward aggregation and policy optimization.

B.4 Training Procedure and Configuration

We initialize DreamerAD–Epona from the pretrained Epona model (Zhang et al., 2025) and first adapt it to NAVSIM for 2 Hz prediction. We then train SF-WM, AD-RM, and the policy in sequence. Shortcut training updates STT and both the visual and trajectory generation branches. Reward-model training updates only AD-RM, while RL updates only the trajectory generation branch. Table 6 summarizes the training configurations and approximate wall-clock costs. NAVSIM adaptation and SF-WM training use the AdamW settings listed in the table.

Table 6: **Training configuration and computational cost of the Epona-based implementation.** Training time denotes approximate wall-clock time.

Stage	Batch Size	Learning Rate	Epochs	Training Time
NAVSIM Adaptation	128	3×10^{-5}	5	$\sim$1 day
Shortcut Forcing	128	3×10^{-5}	12	$\sim$3 days
Reward Model	320	3×10^{-4}	12	$\sim$1 week
Reinforcement Learning	196	1×10^{-6}	2	$\sim$8 hours

Implementation settings	
Training hardware	32 NVIDIA H20 GPUs
Image resolution	512×1024
Future waypoints T / interval Δt	8 / 0.5 s
RL sampling vocabulary	Spatially filtered subset
Annotation candidates per scene M	256
Endpoint thresholds (x, y, θ)	$(10\,\mathrm{m}, 5\,\mathrm{m}, 20^\circ)$
Compressed visual tokens per frame l	32
RL candidates per group G	8
Adaptation / SF-WM optimizer	AdamW
Adaptation / SF-WM weight decay	5×10^{-2}
SF-WM maximum multi-scale sampling steps	64
SF-WM teacher EMA decay	0.999
SF-WM steps during RM / RL training	1
Trajectory generation steps	20
Latency measurement hardware	1 NVIDIA H20 GPU

The world-model latency reported for Epona measures rollout computation on a single NVIDIA H20 GPU, rather than end-to-end policy inference.

C Wan2.2 Implementation Details

C.1 Backbone and Initialization

We build the Wan2.2-based implementation on DriveVA (Liu et al., 2026), which adapts Wan2.2-TI2V-5B for joint video and trajectory prediction. The backbone contains a video VAE, a UMT5 text encoder, and a shared DiT with separate video and trajectory output heads. The DiT uses 30 transformer blocks with a hidden dimension of 3,072 and 24 attention heads.

Video representation and tokenization. The video VAE has a temporal compression factor of 4, a spatial compression factor of 16, and 48 latent channels. It encodes the first frame separately and then advances in groups of four frames. Thus, a sequence of five observed frames and 8 future frames is represented by four latent time positions: two for the observed history and two for the future. The future positions correspond to frames 1–4 and 5–8, respectively; these are temporal

compression groups, not independently encoded clips. At an image resolution of 480×832, the full video latent has shape $B \times 48 \times 4 \times 30 \times 52$, where B is the batch size. The DiT uses $1 \times 2 \times 2$ patches in latent time, height, and width, respectively. Patch embedding therefore preserves the four latent time positions and produces $15 \times 26 = 390$ spatial tokens at each position, for a total of 1,560 video tokens. Both future latent positions are denoised jointly. VAE decoding is used only when RGB predictions are required for visualization.

Trajectory representation. The trajectory contains 8 waypoints at 0.5-second intervals, $\tau \in \mathbb{R}^{B\times 8\times 3}$, covering a four-second planning horizon. Each waypoint contains (x, y, ψ): planar position in meters and heading in radians, expressed in the current ego rear-axle coordinate frame. We retain DriveVA's affine normalization:

$$\widetilde{x} = 2\frac{x + 1.57}{66.74} - 1, \qquad \widetilde{y} = 2\frac{y + 19.68}{42.0} - 1, \qquad \widetilde{\psi} = 2\frac{\psi + 1.67}{3.53} - 1. \tag{17}$$

The inverse transformation recovers physical coordinates for trajectory output. A two-layer MLP with a 256-dimensional hidden layer maps each normalized waypoint to the 3,072-dimensional DiT token space.

Ego-motion and text conditioning. The planar ego velocity (v_x, v_y), in meters per second, is projected by a separate MLP into one conditioning token. The DreamerAD interface uses this velocity token without additional historical-trajectory tokens. Text conditioning is obtained by encoding a templated driving prompt with the UMT5 encoder. The prompt specifies the driving command and current speed and asks for a four-second driving continuation from the observed history. Its embeddings enter the DiT through text cross-attention.

Shared sequence and prediction interfaces. The shared DiT sequence is ordered as video tokens, one velocity token, and 8 future-waypoint tokens, giving 1,569 tokens per example. The two output heads predict video-latent and trajectory flow velocities; the velocity prefix is a condition and is excluded from the trajectory update. For joint policy prediction, observed video latents remain clean and fixed, while future video latents and normalized trajectory variables are updated. For candidate evaluation, the normalized candidate waypoints instead enter as fixed clean trajectory tokens. Only the future video latents are updated; the predicted trajectory velocity is not applied. This preserves the same candidate throughout its imagined rollout and reward evaluation.

C.2 NAVSIM Adaptation and Policy Training

We adapt the DriveVA-initialized model on the NAVSIM trainval split. Each training sequence contains five observed frames, including the current frame, followed by 8 future frames sampled at 2 Hz. Images are resized to 480×832. The observed frames provide clean visual conditioning, while the future frames and the corresponding ego trajectory provide supervision for video and trajectory prediction.

During training, noise is added to the future video latents and trajectory variables, while the observed history is retained as clean conditioning. The shared DiT processes both modalities, and the video and trajectory heads predict their respective flow velocities. The training objective consists of video and trajectory flow-matching losses, each measuring the mean-squared error between the predicted velocity and its corresponding target. Video supervision is applied only to future latents. Their combined objective is

$$\mathcal{L}_{\text{adapt}} = \mathcal{L}^{\text{policy}}_{\text{video}} + \mathcal{L}^{\text{policy}}_{\text{traj}} \tag{18}$$

These video and trajectory losses define the supervised NAVSIM adaptation stage. We retain the joint prediction interface when integrating the backbone into DreamerAD; the subsequent shortcut and reward-model training objectives are defined in Appendix C.3.

C.3 Shortcut Forcing and Reward Modeling

Shortcut forcing. We augment the shared DiT with integration-step conditioning to support video prediction with different sampling budgets. The observed history remains clean throughout training. For shortcut supervision, an EMA teacher performs two consecutive local updates, and the student

Table 7: Configuration of the Wan2.2-based implementation.

Configuration	Value
Backbone and inputs	
Backbone	Wan2.2-TI2V-5B (Wan et al., 2025)
Initialization	DriveVA (Liu et al., 2026)
Image resolution	480×832
Observed / future frames	5 / 8
Frame rate	2 Hz
Planning horizon	4 s, 8 waypoints
Shortcut forcing and reward modeling	
Initial shortcut training	3 epochs
Reward-model training	2 epochs
RM / RL reward evaluation steps	1
Final evaluation trajectory sampling steps	4
Reward feature dimension	1,024
Spatial queries per latent time position	32
Reward prediction horizons	8
Reward metrics per horizon	8
Reinforcement learning	
Training GPUs	16
Candidates per observation	8
Ratio clipping interval	$[0.8, 1.2]$
Behavioral cloning coefficient	1
Learning rate	3×10^{-7}
Global batch size	32
Training epochs	2
EMA decay	0.999

learns to reproduce their combined video-latent displacement with a single update. Under Wan2.2's shifted noise schedule, the two local updates can span unequal noise-level intervals. We weight each teacher velocity by its share of the total noise-level change when forming the target. This interval-weighted average reduces to the equal-weight average in Section 3.2 when the two intervals are equal. Gradients are stopped through the teacher target. We retain ground-truth trajectory flow-matching supervision during shortcut training, rather than distilling the trajectory output. We first train the shortcut-conditioned world model for 3 epochs.

Dense reward modeling. For each candidate trajectory, we keep the trajectory fixed as a clean condition and apply one shortcut update to the temporally compressed future video latents, following the default setting in Table 7. Visual features are projected into a 1,024-dimensional reward feature space and compressed using 32 learned queries at each of the four latent time positions, yielding 128 visual context tokens while retaining their temporal organization. The compressor uses one transformer decoder layer. Reward prediction uses three transformer decoder layers with 8 attention heads and separate MLP output heads for the 8 metrics. Metric and future-step embeddings define $8 \times 8 = 64$ reward queries, which are additionally conditioned on the candidate trajectory embedding. These horizon-specific queries produce step-wise rewards at the 8 future waypoints. We apply a block-wise cross-attention mask to the visual context: queries for future steps 1–4 attend to the observed-history blocks and the first future latent block, while queries for steps 5–8 attend to the observed-history blocks and both future latent blocks. This mask restricts direct attention at the latent-block level, reflecting the temporal compression of the video backbone. At each future trajectory step, the reward model predicts the 8 driving metrics: NC, DAC, TTC, EP, DDC, LK, TLC, and HC. For each candidate, the output consists of 8 logits at each of 8 future steps. Applying a sigmoid produces the step-wise scores $\mathbf{p}^j \in [0, 1]^8$ used in Section 3.4. Starting from the shortcut-trained model, we train the reward model for the duration listed in Table 7.

Reward-model optimization. The reward model is supervised using a weighted binary cross-entropy loss over trajectory candidates, future steps, and driving metrics:

$$\mathcal{L}_{\mathrm{RM}} = \frac{\sum_{n=1}^{B_c} \sum_{j=1}^{8} \sum_{k=1}^{8} \gamma_j \omega_k \, \mathrm{BCEWithLogits}\left(s_{n,j,k}, y_{n,j,k}\right)}{B_c \left(\sum_{j=1}^{8} \gamma_j\right) \left(\sum_{k=1}^{8} \omega_k\right)}, \tag{19}$$

where B_c is the number of candidate instances in the batch, $s_{n,j,k}$ is a predicted logit, and $y_{n,j,k}$ is its target. The temporal and metric weights γ_j and ω_k follow the notation of Section 3.3. The denominator normalizes the weighted loss over the candidate batch, future steps, and metrics. We use temporal weights $(1, 1, 1, 1, 1.2, 1.5, 2, 3)$ and metric weights $(3, 5, 4, 3, 1, 2, 1, 1)$ in the order (NC, DAC, TTC, EP, DDC, LK, TLC, HC).

C.4 Reinforcement Learning

Trajectory sampling. Given the observed driving context, the policy predicts a continuous trajectory. Each vocabulary trajectory contains 8 (x, y, ψ) waypoints in the same ego coordinate frame. As in the Epona implementation, we first apply the spatial constraints in Section 3.3 and then use Gaussian vocabulary sampling over the retained subset, following Section 3.4. Sampling complete vocabulary trajectories provides structured exploration around the current policy.

Online imagination and reward evaluation. Each candidate trajectory is supplied as a clean action condition to the shortcut-trained world model, which predicts future video latents in one sampling step conditioned on the observed history. The Wan2.2 implementation of AD-RM trained in Appendix C.3 evaluates the candidate-conditioned latent features to produce step-wise driving scores. We apply the log-domain score fusion at each future step and then take the temporally weighted sum specified in Section 3.4. Candidate rewards are therefore obtained online through latent-space imagination, without decoding the predicted latents into RGB images.

Policy optimization. We compute group-relative advantages from the candidate rewards and optimize the policy using a clipped Gaussian surrogate objective. The predicted trajectory defines the Gaussian mean, and the likelihoods of each candidate under the current and old policies form the surrogate ratio. This ratio approximates the policy update rather than providing an exact importance correction for discrete vocabulary sampling. Behavioral cloning on ground-truth trajectories regularizes the optimization. At final evaluation, we use four sampling steps to generate the predicted trajectory.

D Additional Experimental Analysis

This appendix examines reward-model data requirements, closed-loop behavior across scenario difficulties, trajectory exploration, and reward prediction quality.

D.1 Reward Model Training Data Scale

Table 8: Effect of reward model training data scale on NAVSIM v2 EPDMS.

RM training data	DreamerAD–Epona	DreamerAD–Wan2.2
20%	87.5	91.2
40%	87.5	91.1
100%	**87.8**	**91.4**

Table 8 reports downstream planning performance when AD-RM is trained on different fractions of the training scenes. EPDMS varies by at most 0.3 points across the tested fractions for either backbone. Using the full dataset gives the best performance, improving over the 20% setting by 0.3 points for Epona and 0.2 points for Wan2.2. These results suggest limited sensitivity to RM training data scale within the evaluated range.

Table 9: Difficulty-wise HD-Score on HUGSIM (0–100). Each difficulty averages the four datasets equally; Avg. uses the official difficulty counts of 80/157/96/103 across 436 indexed scenarios.

Method	Easy	Medium	Hard	Extreme	Avg.
Epona	59.96	38.30	14.02	7.76	29.71
DreamerAD–Epona	68.22	38.25	19.45	7.74	32.40
Wan2.2	74.59	62.28	34.02	13.01	46.68
DreamerAD–Wan2.2	**83.23**	**72.24**	**44.70**	**16.21**	**54.96**

D.2 Difficulty-wise Analysis

Table 9 compares each DreamerAD variant with its corresponding baseline across HUGSIM difficulties. DreamerAD–Wan2.2 improves the average HD-Score by 8.28 points. The gains are 8.64, 9.96, and 10.68 points on Easy, Medium, and Hard scenarios, respectively, with the largest improvement on Hard. The Extreme score increases by 3.20 points to 16.21, although this subset remains substantially more challenging than the other difficulties. DreamerAD–Epona improves the average by 2.69 points, with gains concentrated on Easy (+8.26) and Hard (+5.43); Medium (-0.05) and Extreme (-0.02) remain nearly unchanged. These results show that the aggregate gains are driven mainly by Easy–Hard scenarios, while Extreme scenarios remain challenging for both implementations.

D.3 CLOSED-LOOP QUALITATIVE COMPARISON

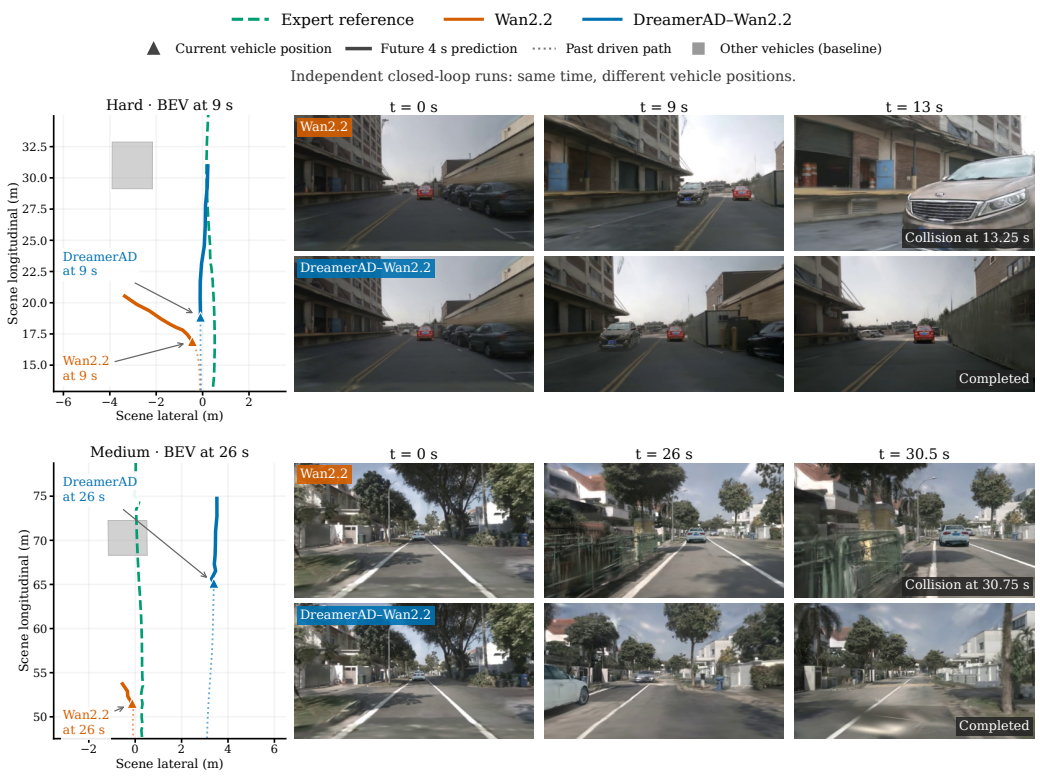

Figure 6: Closed-loop comparison of Wan2.2 and DreamerAD–Wan2.2 on HUGSIM. Dashed green lines show the expert reference route; triangles mark ego positions, solid lines show predicted motion over 4 s, and dotted lines show driven paths. BEV panels share scene coordinates, and camera frames are synchronized by simulation time. Each model follows its own closed-loop observations and vehicle state. Gray boxes represent other vehicles in the baseline observation. Outcome labels describe the full rollouts.

Figure 6 illustrates two cases in which DreamerAD–Wan2.2 completes the route while its baseline collides. In the Hard case, the baseline prediction deviates toward an oncoming vehicle, whereas DreamerAD follows the expert reference and passes it. In the Medium case, DreamerAD passes a slower vehicle and continues along the route. The baseline collisions occur at 13.25 s and 30.75 s in the two cases, respectively.

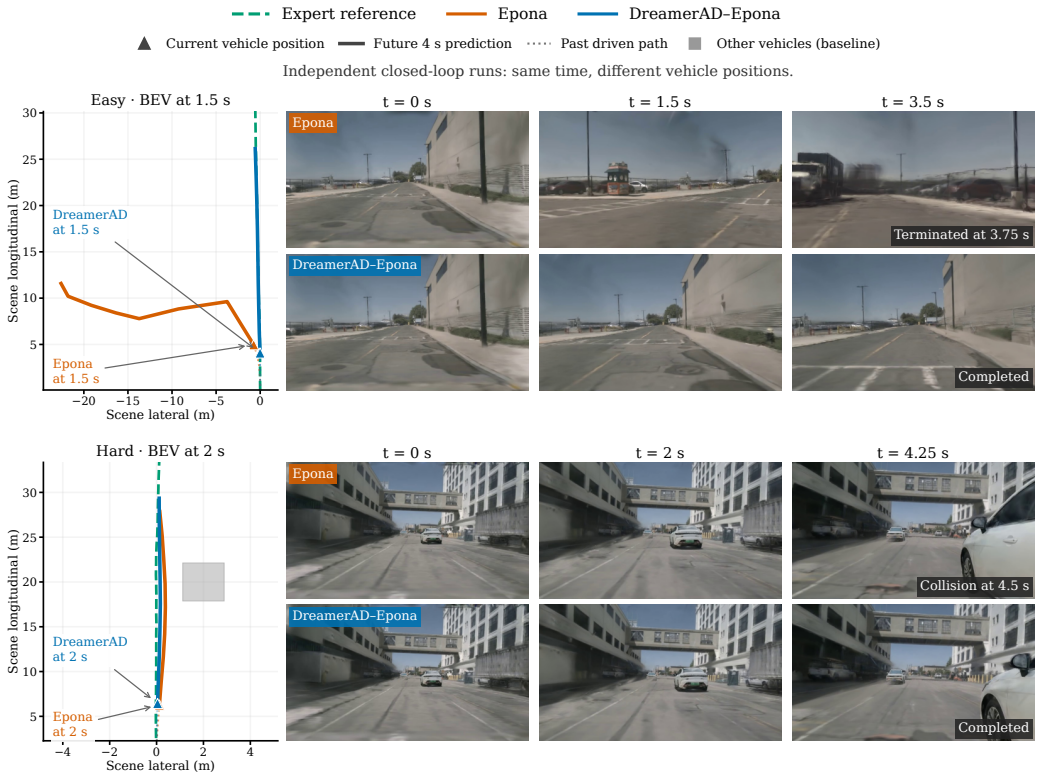

Figure 7: Closed-loop comparison of Epona and DreamerAD–Epona on HUGSIM, using the BEV conventions and synchronized-camera layout of Figure 6. Frames cover the common interval before baseline termination; outcome labels describe the full rollouts.

Figure 7 shows corresponding comparisons for Epona. In the Easy case, the baseline predicts a pronounced lateral deviation and terminates at 3.75 s with 8.99% route completion. In the Hard case, it collides with a vehicle near the planned path at 4.5 s. DreamerAD–Epona completes both routes. Together, these selected cases illustrate the within-backbone improvements summarized in Table 9.

D.4 CLOSED-LOOP FAILURE CASES

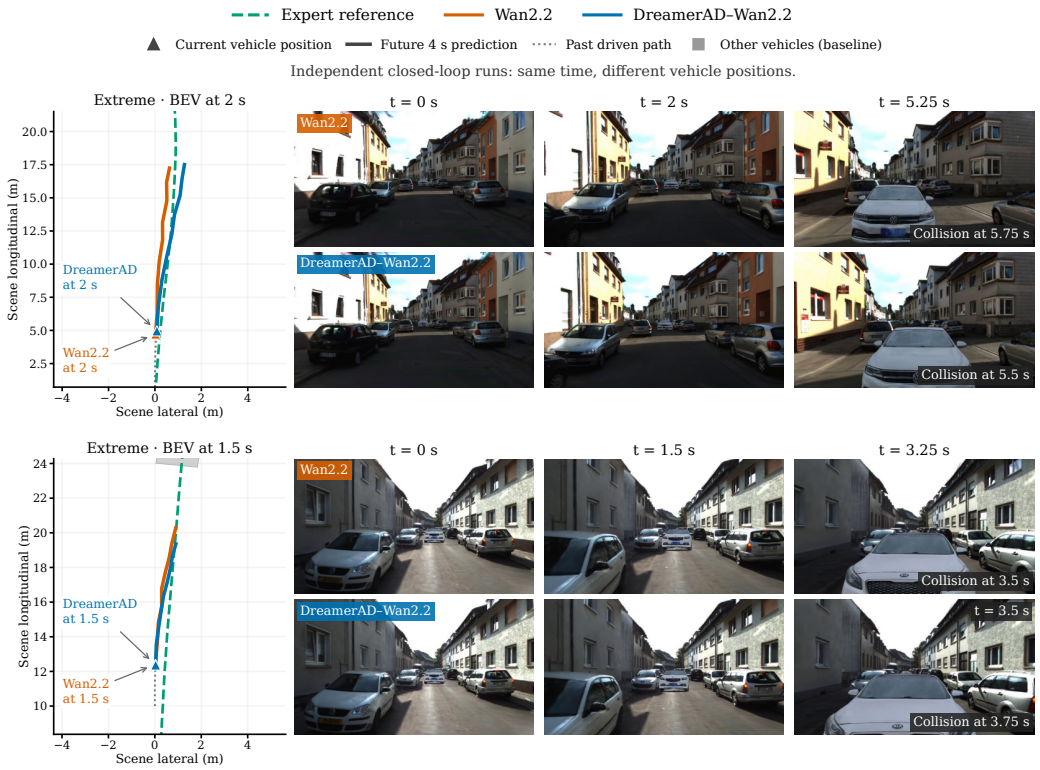

Figure 8: Closed-loop failures of Wan2.2 and DreamerAD–Wan2.2 in two Extreme HUGSIM scenarios. BEV panels show the expert reference and predicted trajectories; camera sequences show the encounters with oncoming vehicles. Final views use the last available frame at or before the requested time, with differing timestamps labeled.

Figure 8 shows two narrow-street encounters with parked cars and oncoming vehicles in which both planners collide. DreamerAD–Wan2.2 collides at 5.5 s and 3.75 s, completing 7.99% and 10.18% of the routes; the baseline collides at 5.75 s and 3.5 s. These failures show that improved average performance does not resolve all difficult interactions, consistent with the low Extreme HD-Scores in Table 9.

D.5 Exploration Geometry and Simulator Scores

We compare 8 candidates per sampling strategy around one policy prediction in a scene selected from 8 low-score scenes. All candidates use the same preceding-frame prediction for EC.

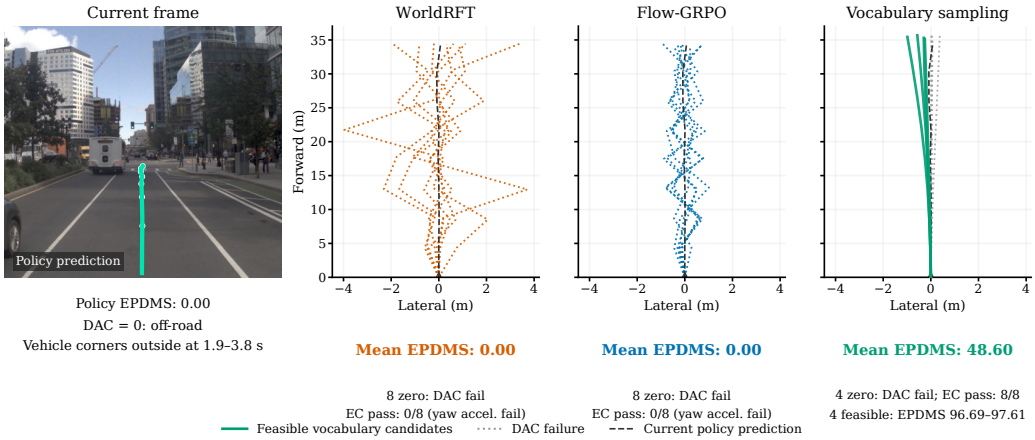

Figure 9: Candidate trajectories around a zero-score policy prediction (black dashed BEV line). Each group contains 8 candidates. Solid green candidates are feasible; dotted candidates fail DAC.

As illustrated in Figure 9, Gaussian-weighted vocabulary sampling produces smooth candidate trajectories that all satisfy EC in this example. The candidates also exhibit distinct simulator scores, providing relative evaluation signals to guide policy optimization while preserving cross-frame motion consistency.

D.6 Reward Model Ranking Accuracy

We evaluate both reward models over the full NAVSIM trainval split. For each scene, we randomly sample 8 distinct non-expert vocabulary trajectories and evaluate each at 8 future steps (0.5–4.0 s). Both implementations use the same scenes and candidates, with simulator-generated metric labels providing the reference scores.

For each trajectory, we combine the 8 metric scores at each future step using the log-domain fusion in Section 3.4, with weight 5 for TTC and weight 1 for each remaining metric. Metric values are floored at 0.01 before fusion, and the 8 step-wise scores are averaged to obtain a trajectory score. The same aggregation is applied to RM predictions and simulator labels. We compare all $\binom{8}{2} = 28$ unordered candidate pairs within each scene. Pairs with equal simulator-derived scores are excluded. A matching predicted ordering receives one credit, a reversed ordering receives zero, and a predicted tie receives half a credit. Ranking accuracy is the total credit divided by the total number of eligible pairs across all scenes, reported as a percentage.

Table 10: Within-scene trajectory ranking accuracy on the full NAVSIM trainval split.

Reward model	Ranking accuracy (%) ↑
DreamerAD–Epona	90.1
DreamerAD–Wan2.2	95.3

Table 10 reports ranking accuracies of 90.1% for DreamerAD–Epona and 95.3% for DreamerAD–Wan2.2. This evaluation measures whether AD-RM identifies relatively better trajectories within a candidate group, directly assessing the ordering information needed for group-relative policy optimization. The agreement with simulator-derived rankings supports the use of AD-RM to provide relative evaluation signals during imagination-based RL.