RISE: ADAPTIVE IMAGINATION FOR WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World Action Models (WAMs) improve planning by incorporating future world evolution into action generation, yet existing methods allocate a fixed imagination budget to every scene. We propose RISE (**R**efining **I**magination through **SE**lective Rollout), a system-level adaptive imagination framework that makes sequential ROLL/STOP decisions according to the expected planning benefit of continued rollout. At each step, a Latent Evaluator estimates the risk revealed by the current prefix and how much planning could improve if imagination continues, while a Rollout Gate weighs this expected benefit against additional computation cost. Since factual driving logs expose only one realized future, we further construct **CounterDrive**, a counterfactual dataset with diverse outcomes and risk levels, to enrich future dynamics and provide localized risk supervision. Each retained sample undergoes expert verification and annotation of trajectory validity, incident onset, and causal category, providing a reusable resource for safety-critical world-modeling research. Experiments on NAVSIM and nuScenes show that RISE achieves the best overall planning performance while reducing unnecessary rollout, with additional transfer results supporting its plug-in generality across WAM architectures.

1 INTRODUCTION

End-to-end autonomous driving Li et al. (2025a); Shang et al. (2025) is moving beyond direct trajectory regression toward models that reason about the future before acting Chen et al. (2025); Maes et al. (2026). World Action Models (WAMs) Shen et al. (2026); Wang et al. (2026c) incorporate predicted world evolution into planning, allowing policies to evaluate actions through their consequences rather than the current observation alone. This capability is particularly relevant to driving, where similar visual contexts can lead to different outcomes as the ego vehicle and surrounding agents interact. Consequently, recent methods Wang et al. (2026b); Yang et al. (2026b) employ latent future prediction, action-conditioned dynamics, or world-model-based policy learning to improve planning under interaction and uncertainty.

However, future imagination adds inference cost because latent futures must be generated before an action can be produced. As illustrated in Fig. 1(a)–(c), existing WAMs Wang et al. (2026a); Yang et al. (2026c); Yuan et al. (2026); Lu et al. (2026) organize this computation differently. Shared-backbone models perform prediction and planning from a common representation through an *imagine-and-plan* design. Cascaded models complete a future rollout before passing it to the Planner through an *imagine-then-plan* design. Other approaches remove test-time imagination and plan directly from the observation. Despite these differences, their inference schedules are specified globally. They do not reassess after each partial rollout whether the current prefix is sufficient for planning or continued prediction is likely to improve the final decision.

We refer to this missing decision signal as *Future Planning Gain*. At rollout step, its profile measures the change in planning score associated with each valid continuation relative to stopping with the current prefix. By expressing continuation utility in planning score, it directly connects additional rollout to downstream planning quality. Because each predicted latent updates the current prefix and the available future evidence, the gain may change throughout rollout. Therefore, it should be re-estimated after each step and used as a sequential stopping signal rather than to select a final rollout depth in advance.

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.

This perspective captures the varying utility of imagination without dividing scenes into predefined easy and difficult categories. The observation or an early prefix may already support a stable trajectory, so continuing offers little gain and only increases latency. Alternatively, the current prefix may leave an interaction unresolved, and an additional predicted step may change the preferred plan. In either case, the decision is local: the model should ROLL when the expected improvement justifies its cost and STOP otherwise. Repeating this decision produces a horizon ranging from zero rollout to the dataset-specific maximum.

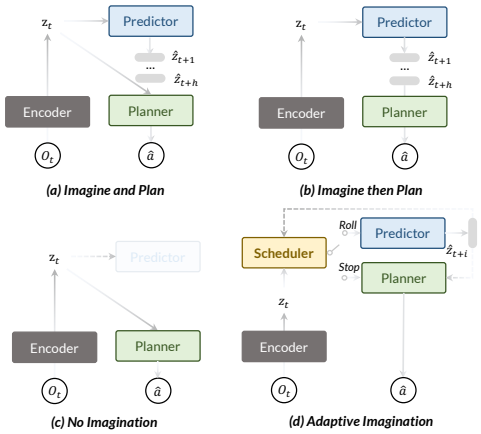

Figure 1: Comparison of various imagination strategies in WAMs.

Building on this principle, we introduce RISE (**R**efining **I**magination through **SE**lective Rollout), the plug-in adaptive imagination framework illustrated in Fig. 1(d). RISE augments an Encoder–Predictor–Planner WAM with a lightweight Scheduler composed of a Latent Evaluator and a Rollout Gate. At each step, the Evaluator predicts a Risk Profile summarizing the planning-relevant risk already revealed and a Future Planning Gain Profile describing the improvement available through continued rollout. The Gate weighs the predicted gain against computation cost and makes the current binary decision. Under ROLL, the Predictor appends one future latent and the Scheduler evaluates the updated prefix again. Under STOP, the selected prefix is passed once to the Planner. The Gate does not predict the final horizon directly. Therefore, the effective horizon emerges from repeated ROLL/STOP decisions.

The two Evaluator signals require different supervision. Planning scores obtained by stopping at every valid prefix provide direct Future Planning Gain targets: the score difference between a continued and current prefix measures the realized benefit of continuation. Risk learning faces a different limitation because factual logs record only the future that occurred and leave plausible safety-critical alternatives unseen. We construct COUNTERDRIVE to augment selected contexts with diverse counterfactual outcomes and risk levels. Accepted clips enrich future-latent prediction, while verified factual–counterfactual incident pairs provide temporally localized Risk Profile supervision. Consequently, all-horizon planning outcomes teach how planning changes under continuation, while CounterDrive improves sensitivity to risks revealed along that process.

Extensive experiments on NAVSIM and nuScenes validate the effectiveness of RISE. It achieves 91.5 PDMS and 90.8 EPDMS on NAVSIM v1 and v2, respectively, while obtaining state-of-the-art trajectory accuracy and collision performance on nuScenes. Further analyses show that CounterDrive improves hazard discrimination and that the Scheduler transfers to another architecture without modifying its Predictor or Planner, supporting both adaptive imagination and architectural generality. Our contributions are summarized as follows:

- We propose **RISE**, a plug-in adaptive imagination framework for WAMs that uses Future Planning Gain to make sequential ROLL/STOP decisions, yielding a scene-dependent rollout horizon while balancing planning quality and inference cost.
- We construct **CounterDrive**, a counterfactual driving dataset with diverse interaction outcomes and risk levels, supporting both RISE training and safety-critical world modeling research.
- Experiments on driving benchmarks show that RISE achieves state-of-the-art planning performance with lower rollout cost, while its transfer to other WAM architecture supports plug-in generality.

2 COUNTERDRIVE

Factual driving logs contain only realized futures and provide limited safety-critical alternatives. Therefore, we construct COUNTERDRIVE from selected NAVSIM and nuScenes scenes. Each re-

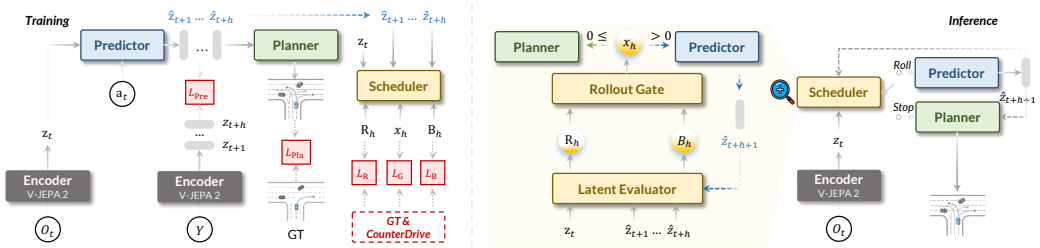

Figure 2: Overview of RISE. The WAM learns future prediction and variable-prefix planning. The Latent Evaluator predicts a Risk Profile $\mathbf{R}_h$ and a Future Planning Gain Profile $\mathbf{B}_h$, supervised by counterfactual risks and all-horizon planning outcomes. The Rollout Gate balances the predicted gain against computation cost, routing $x_h > 0$ to further rollout and $x_h \leq 0$ to planning.

tained counterfactual clip is associated with its factual source, forming pairs within the selected subset rather than covering the complete datasets.

Counterfactual Video Generation For each selected source key frame, the prompt combines a fixed instruction, a scene description, and an incident description specifying the event location and involved object. Prompts additionally constrain the camera viewpoint, road geometry, background, and initial traffic configuration so that the generated clip remains anchored to the source scene. Conditioned on the key frame and prompt, Wan 2.7 generates a 10-second video at 1080p resolution, which is sampled at 2 Hz into 20 frames.

Trajectory Annotation We apply OpenVO Nguyen et al. (2026) to recover the frame-wise ego poses:

$$\tilde{\tau}_{i,t} = (\tilde{p}^x_{i,t}, \tilde{p}^y_{i,t}, \tilde{\psi}_{i,t}). \tag{1}$$

The corresponding ego-motion actions are computed from adjacent poses:

$$\tilde{a}_{i,t} = \mathcal{A}(\tilde{\tau}_{i,t}, \tilde{\tau}_{i,t+1}). \tag{2}$$

Human Verification and Annotation Annotators verify ego-motion consistency, identify the first incident frame, mark generation distortions, and categorize each clip as normal, caused by non-ego behavior, or caused by ego behavior. A recovered trajectory is marked invalid when it disagrees with the visually observed ego motion. For ego-caused incidents, annotators also record recommended avoidance or stopping actions as clip-level metadata. Clips with severe distortion or unreliable motion are removed. Accepted clips supervise future prediction, while verified incident pairs provide temporally localized risk-ranking supervision. After filtering, CounterDrive contains 2,432/511 training/test clips from nuScenes and 5,013/1,000 from NAVSIM. Unpaired factual samples remain available for standard training objectives. Additional construction and annotation details are provided in the Appendix.

3 RISE

We propose RISE (**R**efining **I**magination through **SE**lective Rollout), a plug-in adaptive imagination framework for World Action Models. As shown in Fig. 2, RISE augments an Encoder–Predictor–Planner WAM with a lightweight Scheduler composed of a Latent Evaluator and a Rollout Gate. Evaluator estimates the planning risk revealed by the current future prefix and the potential gain of deeper rollout, while Gate determines whether additional imagination is worth its computation cost.

3.1 Problem Formulation

Let c denote the current driving context, $\tau_{1:P}$ the future trajectory over planning horizon P, and $z_{1:H}$ the future world representation over a maximum rollout horizon H. A standard World Action Model uses a fixed rollout depth for every scene:

$$p(z_{1:H}, \tau_{1:P} \mid c) = p(z_{1:H} \mid c)\, p(\tau_{1:P} \mid c, z_{1:H}). \tag{3}$$

RISE instead makes the effective rollout depth scene-dependent. Given an imagined prefix $\hat{Z}_h = \hat{z}_{1:h}$, where $\hat{Z}_0 = \emptyset$, the Scheduler determines whether the current prefix is sufficient for planning:

$$\begin{aligned} d_h &= \mathcal{S}(\hat{Z}_h; c, \lambda), \\ K(c; \lambda) &= \min\left(\{h \mid d_h = \textsc{Stop}\} \cup \{H\}\right), \end{aligned} \tag{4}$$

where λ controls the preference for computation cost.

Our RISE models

$$p(z_{1:K(c;\lambda)}, \tau_{1:P} \mid c) = p(z_{1:K(c;\lambda)} \mid c)\, p(\tau_{1:P} \mid c, z_{1:K(c;\lambda)}). \tag{5}$$

Here, $K(c; \lambda) \in \{0, \ldots, H\}$ is selected independently for each scene. When $K = 0$, the model plans directly from the observed context; when $K = H$, it recovers the full-rollout behavior of a standard WAM.

3.2 World Action Model Architecture

Encoder We use a frozen V-JEPA 2 Assran et al. (2025) image encoder with a ViT-L backbone to encode front-camera observations into latent tokens. Four input frames are resized to 256×512 and processed using 16×16 patches and non-overlapping two-frame tubelets. The Encoder produces two observed latent time steps, each containing 512 spatial tokens of dimension 1024.

Predictor We adopt a frame-causal ego motion conditioned Transformer with 12 layers, a hidden dimension of 384, and 12 attention heads. Starting from the observed latent tokens, the Predictor autoregressively generates up to $H_{\mathcal{D}}$ future latent steps, each corresponding to two raw video frames. Hereafter, $H_{\mathcal{D}}$ denotes the dataset-specific realization of the maximum horizon H. Ego-motion conditions are represented by relative pose changes computed between adjacent ego poses. The Predictor uses no route command, additional state input, or camera extrinsic.

Planner We employ a 12-layer diffusion Transformer with a hidden dimension of 384 and 12 attention heads. The Planner conditions on the observed tokens and the selected future prefix, distinguished by learned source embeddings, together with the observed ego-motion history and current ego kinematics. It generates six candidate trajectories using a variance-preserving diffusion process with 20 DPM-Solver++ Lu et al. (2025) sampling steps. Each candidate contains eight future poses at 0.5 s intervals, represented as $(x, y, \cos\psi, \sin\psi)$. Candidate selection and dataset-specific scoring details are provided in the Appendix.

3.3 Adaptive Rollout Scheduler

At rollout depth h, the Scheduler decides whether to stop with the currently available future prefix or generate one additional latent step. It is guided by the predicted Future Planning Gain of continuing rollout and computation cost.

3.3.1 Latent Evaluator

Given the observed latent z_t and imagined prefix $\hat{Z}_h$, the Latent Evaluator predicts a Risk Profile and a Future Planning Gain Profile:

$$\begin{aligned} (R_h, B_h) &= \mathrm{Eval}_{\theta_E}(z_t, \hat{Z}_h), \\ R_h &= [r_1, \ldots, r_h], \\ B_h &= [b_{h\to h+1}, \ldots, b_{h\to H_{\mathcal{D}}}], \end{aligned} \tag{6}$$

where r_k estimates the trajectory risk associated with the prefix ending at depth k, while $b_{h\to j}$ predicts the Future Planning Gain of continuing from depth h to j, measured relative to stopping with the current prefix. Thus, R_h summarizes the risk already exposed, whereas B_h summarizes the potential planning gains available from continued rollout.

3.3.2 Rollout Gate

For $0 \le h < H_{\mathcal{D}}$, we construct

$$\begin{aligned} e_h &= [\rho(z_t), \rho(\hat{Z}_h)], \\ \xi_h &= [\overline{R}_h, \overline{B}_h, h/H_{\mathcal{D}}, c_h, \Delta c_{h+1}], \end{aligned} \tag{7}$$

where $\rho(\cdot)$ denotes token pooling, $\rho(\emptyset)$ is a learned empty-prefix embedding, $\overline{R}_h$ and $\overline{B}_h$ are zero-padded to length $H_{\mathcal{D}}$, c_h is the cumulative cost, and $\Delta c_{h+1} = c_{h+1} - c_h$. Given computation preference λ, the Gate predicts

$$x_h = G_{\theta_G}(e_h, \xi_h, \lambda). \tag{8}$$

The sign of x_h indicates whether the predicted planning gain available from continued rollout justifies its additional cost. The Scheduler applies

$$d_h = \begin{cases} \textsc{Roll}, & x_h > 0 \text{ and } h < H_{\mathcal{D}}, \\ \textsc{Stop}, & \text{otherwise.} \end{cases} \tag{9}$$

For ROLL, the Predictor appends one future latent step and the Scheduler evaluates the extended prefix. For STOP, the selected prefix is risk-refined when $h > 0$ and passed once to the Planner. Refinement is skipped when $h = 0$, and rollout terminates at $H_{\mathcal{D}}$ by construction.

3.4 THREE-STAGE TRAINING

RISE is trained in three stages. Stage I trains the Predictor and an initial variable-prefix Planner. Stage II learns planning-oriented risk and Future Planning Gain. Stage III converts these signals into a cost-aware stopping policy.

3.4.1 STAGE I: PREDICTOR AND INITIAL PLANNER

The Predictor recursively generates the maximum dataset-specific prefix $\hat{Z}_{H_{\mathcal{D}}}$. It is trained on real and accepted CounterDrive sequences using teacher-encoded future latents:

$$\mathcal{L}_{\text{Pre}} = \sum_{k=1}^{H_{\mathcal{D}}} \ell_z \left(\hat{z}_{t+k}, \text{sg}(z_{t+k}) \right), \tag{10}$$

where z_{t+k} is the target latent, $\text{sg}(\cdot)$ denotes the stop-gradient. For CounterDrive sequences, the ego-motion conditions are computed from adjacent recovered poses. We then train an initial Planner Π_0 using real trajectory supervision:

$$\mathcal{L}^0_{\text{Pla}} = \ell_\tau \left(\Pi_0(z_t, \hat{Z}_h), \tau^* \right), h \sim \{0, \ldots, H_{\mathcal{D}}\}. \tag{11}$$

The Encoder, Predictor, and Π_0 are frozen when constructing the subsequent training targets.

3.4.2 STAGE II: LATENT EVALUATOR AND GUIDED PLANNER

For each real prefix, Π_0 produces candidate trajectories whose planning risk is computed by a fixed geometry-based evaluator:

$$\begin{aligned} r^*_k &= \mathcal{E}_{\text{risk}} \left(\Pi_0(z_t, \hat{Z}_k) \right), \\ \mathcal{L}_{\text{real}} &= \sum_{k=1}^{H_{\mathcal{D}}} \ell_{\text{Huber}} \left(r_k, r^*_k \right). \end{aligned} \tag{12}$$

The evaluator includes the common candidate-selection rule and geometry-based risk terms, whose definitions are provided in the Appendix.

CounterDrive contributes paired supervision only when a verified factual–counterfactual pair is available. Let $\mathcal{I}_{\text{pair}}$ denote the verified pairs in which the counterfactual future contains an annotated incident and is ranked as riskier than its factual source. For a pair i, the annotated incident frame is mapped to latent step k^{inc}_i, and we define

$$\begin{aligned} m_{i,k} &= \mathbb{I}\left[k \geq k^{\text{inc}}_i \right], \\ \mathcal{L}_{\text{rank}} &= \sum_{i \in \mathcal{I}_{\text{pair}}} \sum_{k=1}^{H_{\mathcal{D}}} m_{i,k} \left[\gamma + r^{\text{fac}}_{i,k} - r^{\text{cf}}_{i,k} \right]_+ . \end{aligned} \tag{13}$$

Real samples without a paired counterfactual clip are excluded only from this ranking term. The complete Risk Profile objective is

$$\mathcal{L}_R = \mathcal{L}_{\text{real}} + \beta_{\text{cf}} \mathcal{L}_{\text{rank}} + \beta_{\text{loc}} \mathcal{L}_{\text{loc}}, \tag{14}$$

Table 1: Performance on nuScenes.

Method	L2 (m)↓				Collision Rate↓			
	1s	2s	3s	Avg.	1s	2s	3s	Avg.
BEV-Planner	0.30	0.52	0.83	0.55	0.10	0.37	1.30	0.59
LAW	0.26	0.57	1.01	0.61	0.14	0.21	0.54	0.30
World4Drive	0.23	0.47	0.81	0.50	0.02	0.12	0.33	0.16
WorldRFT	0.21	0.44	0.76	0.47	0.10	0.11	0.23	0.15
DAWN	0.17	0.31	0.52	0.33	0.00	**0.10**	0.23	0.11
RISE	**0.16**	**0.29**	**0.49**	**0.31**	**0.00**	0.11	**0.20**	**0.10**

Table 2: Performance on NAVSIMv1.

Method	NC↑	DAC↑	EP↑	C↑	TTC↑	PDMS↑
DriveVLA-W0	98.4	95.3	80.9	100	95.2	87.2
PWM	98.6	95.9	81.8	100	95.4	88.1
DreamerAD	98.0	97.2	83.1	100	94.3	88.7
DriveLaW	99.0	97.1	81.3	100	96.7	89.1
Drive-JEPA	98.7	96.2	82.9	100	95.5	89.0
DAWN	98.7	95.9	84.3	100	96.0	89.1
EponaV2	98.6	97.9	84.8	100	95.7	90.4
DriveFuture	98.8	**99.1**	84.2	100	95.4	90.7
RISE	**99.1**	97.7	**87.4**	100	**98.6**	**91.5**

where $\mathcal{L}_{\text{loc}}$ uses the annotated incident onset for temporally localized risk calibration. Its detailed construction and loss weights are provided in the Appendix.

After learning the Risk Profile, we refine each non-empty prefix using a small norm-constrained update:

$$\tilde{Z}_h^{m+1} = \tilde{Z}_h^m - \eta \nabla_{\tilde{Z}_h^m} \sum_{k=1}^{h} r_k, \tilde{Z}_h^0 = \hat{Z}_h. \tag{15}$$

For $h = 0$, we define $\tilde{Z}_0 = \emptyset$ and skip refinement. The final Planner Π_1 is trained at all valid rollout depths:

$$\mathcal{L}_{\text{Pla}}^1 = \ell_\tau \left(\Pi_1(z_t, \tilde{Z}_h), \tau^* \right), h \sim \{0, \ldots, H_{\mathcal{D}}\}. \tag{16}$$

We evaluate Π_1 at every rollout depth and denote its dataset-specific planning score by

$$q_h = \mathcal{E}_{\text{plan}} \left(\Pi_1(z_t, \tilde{Z}_h), \tau^* \right). \tag{17}$$

The Future Planning Gain target is

$$\begin{aligned} B_h^* &= [q_{h+1} - q_h, \ldots, q_{H_{\mathcal{D}}} - q_h], \\ \mathcal{L}_B &= \sum_{h=0}^{H_{\mathcal{D}}-1} \ell_B \left(B_h, B_h^* \right). \end{aligned} \tag{18}$$

The loss details are in the Appendix.

3.4.3 Stage III: Rollout Gate

Finally, we freeze other modules and train the Rollout Gate as a cost-aware stopping policy. For each real sample, we enumerate all valid rollout depths and record q_h and cumulative cost c_h. For a computation preference λ, the best remaining cost-adjusted gain is

$$g_h^* = \max_{j \in \{h+1, \ldots, H_{\mathcal{D}}\}} \left[(q_j - q_h) - \lambda (c_j - c_h) \right]. \tag{19}$$

The continuation target and Gate objective are

$$\begin{aligned} y_h^* &= \mathbb{I} \left[g_h^* > 0 \right], \\ \mathcal{L}_G &= \sum_{h=0}^{H_{\mathcal{D}}-1} \ell_{\text{BCE}} \left(\sigma(x_h), y_h^* \right). \end{aligned} \tag{20}$$

At the maximum depth, rollout stops by construction. Full-horizon enumeration is used only to construct training targets; at inference, the Gate observes only the currently available prefix and the final Planner is invoked once.

4 Experiments

4.1 Implementation Details

All experiments are implemented in PyTorch using bfloat16 precision. The training is conducted on 8 × NVIDIA A100 GPUs with a per-GPU batch size of 4. Videos are sampled at 2 Hz and resized

Table 3: Performance on NAVSIMv2.

Method	NC↑	DAC↑	DDC↑	TL↑	EP↑	TTC↑	LK↑	HC↑	EC↑	EPDMS↑
DAWN	97.3	92.0	99.1	99.7	87.4	96.6	96	98.3	85.5	83.2
DreamerAD	98.0	97.2	99.5	99.8	87.8	97.4	97.5	98.3	72.4	85.1
DriveLaW	98.7	96.9	99.6	99.8	87.5	98.3	97.6	98.4	77.4	88.6
EponaV2	98.5	97.4	99.5	99.9	**87.9**	98.1	97.7	98.2	77.4	88.9
Latent-WAM	98.1	97.3	99.6	99.8	87.7	97.3	97.6	98.1	87.3	89.3
DriveFuture	98.8	99.1	99.6	99.9	86.6	98.4	96.4	98.3	74.8	89.9
RISE	**99.1**	**97.7**	**99.7**	**99.9**	87.8	**98.7**	**98.0**	**98.4**	**87.4**	**90.8**

to 256×512. Each clip contains four observed frames and a dataset-specific future horizon. With two-frame tubelets, we set $H_{\text{NAVSIM}} = 4$ and $H_{\text{nuScenes}} = 3$. We train Π_0 for 20 epochs with a learning rate of 2×10^{-5} and Π_1 for 50 epochs with 5×10^{-5} using AdamW. The scheduler is trained for 50 epochs with a learning rate of 10^{-3}. Latent guidance performs two gradient steps with a step size of 0.05 and a maximum update norm of 0.25. We use normalized rollout costs $c_h = h$ and evaluate $\lambda \in \{0, 0.001, 0.005, 0.01, 0.05\}$, with 0.005 as the default. We utilize NAVSIM Dauner et al. (2024) and nuScenes Caesar et al. (2020) to train and evaluate.

4.2 Main Results

We compare RISE with representative driving WAMs in the perception-free setting, including DrivingGPT Chen et al. (2024), LAW Li et al. (2024), World4Drive Zheng et al. (2025), Epona Zhang et al. (2025), DriveVLA-W0 Li et al. (2025c), PWM Zhao et al. (2025b), DreamerAD Yang et al. (2026c), DriveLaW Xia et al. (2026), Drive-JEPA Wang et al. (2026b), DAWN Lu et al. (2026), EponaV2 Xu et al. (2026), DriveFuture Hong et al. (2026), and Latent-WAM Wang et al. (2026a). As shown in Tables 1, 2 and 3, RISE achieves the best overall planning performance on both NAVSIM V1 and V2, reaching 91.5 PDMS and 90.8 EPDMS, respectively. It surpasses the strongest baselines by 0.8 points on V1 and 0.9 points on V2. On NAVSIM V1, RISE improves the previous best EP and TTC by 2.6 and 1.9 points, respectively. On NAVSIM V2, it ranks first or ties for first on seven of the nine component metrics, demonstrating strong performance across safety, compliance, and planning quality. RISE also establishes state-of-the-art results on nuScenes, achieving the lowest average L2 error of 0.31 m and collision rate of 0.10. These results demonstrate that adaptive rollout improves planning across evaluation settings.

4.3 Ablation Studies

Ablation on Key Components Table 4 isolates the Scheduler and CounterDrive under the same backbone and training settings. CounterDrive alone raises EPDMS/PDMS from 88.9/89.7 to 89.8/90.5, indicating that counterfactual futures provide useful supervision for future representation learning. The Scheduler alone reaches 90.4/91.2, confirming that scene-dependent rollout allocation independently improves planning. Combining both yields the best results of 90.8/91.5, showing their complementary roles: CounterDrive enriches future supervision, while the Scheduler allocates additional imagination where it remains useful.

Table 4: Ablation of the Scheduler and CounterDrive on NAVSIM. Both components improve planning performance.

Scheduler	CounterDrive	EPDMS↑	PDMS↑
		88.9	89.7
	✓	89.8	90.5
✓		90.4	91.2
✓	✓	**90.8**	**91.5**

Ablation on Adaptive Rollout Method Table 5 compares the Scheduler with Random Stop and Latent Margin. Random Stop assigns each sample a reproducible horizon from $\{0, \ldots, 4\}$, whereas Latent Margin stops when consecutive latents converge. Random Stop provides the lowest latency of 264.075 ms but reaches only 89.5 EPDMS. Latent Margin averages 2.98 rollouts and incurs 308.532 ms latency, yet obtains only 89.7 EPDMS. In contrast, the Scheduler achieves 90.8 EPDMS with 2.40 rollouts and 287.429 ms latency. It shows that effective stopping should reflect the planning gain rather than random allocation or latent convergence alone.

Table 5: Ablations on adaptive rollout.

Method	# Avg. Rollout	Latency (ms)	EPDMS↑
Random Stop	2.03	**264.075**	89.5
Latent Margin	2.98	308.532	89.7
Scheduler	2.40	287.429	**90.8**

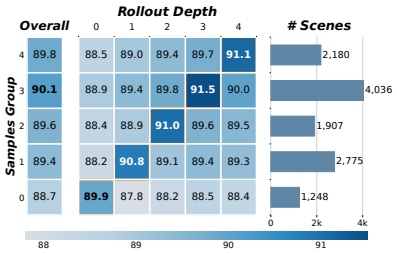

Figure 3: Performance under different rollout depths.



Figure 4: Representative scenes grouped by their preferred rollout depths.

4.4 Further Analysis

Is Adaptive Rollout Necessary? To examine whether a single rollout depth can serve all scenes, we evaluate every sample at fixed depths $h \in \{0, 1, 2, 3, 4\}$. As shown in Fig. 3, the overall EPDMS increases from 88.7 at $h = 0$ to a peak of 90.1 at $h = 3$, but decreases to 89.8 at $h = 4$, showing that deeper imagination is not uniformly beneficial. In contrast, RISE achieves 90.8 EPDMS with adaptive rollout, outperforming every fixed-depth configuration, including the best fixed depth $h = 3$ by 0.7 points. We further group samples by the depth h^* yielding their highest planning score to analyze scene-level variation. Substantial groups appear at every depth: 1,248 scenes favor no rollout, whereas 4,036 and 2,180 favor depths 3 and 4, respectively. For the $h^* = 0$ group, increasing the depth from 0 to 4 reduces EPDMS from 89.9 to 88.4, while the $h^* = 4$ group improves from 88.5 to 91.1. The examples in Fig. 4 provide an intuitive view: open roads often require little imagination, moderate interactions benefit from short rollout, and dense pedestrian or crossing-vehicle scenes benefit from deeper prediction. These results demonstrate that the utility of imagination varies across scenes and motivate allocating rollout adaptively rather than using a globally fixed depth.

Does CounterDrive Improve Safety-Critical Evaluation? To assess whether CounterDrive improves safety-critical evaluation, we evaluate identical models with and without CounterDrive on the held-out counterfactual test set. As shown in Table 6, the model without CounterDrive remains close to random risk ranking, with AUC scores of 0.49–0.52 and an accident-recognition accuracy of 0.51. CounterDrive increases the four horizon-specific AUC scores to 0.93–0.96 and accuracy to 0.96, demonstrating substantially improved risk discrimination and accident awareness.

Table 6: Effect of CounterDrive on safety-critical evaluation. AUC@h measures hazard–safe risk ranking at horizon h. Acc measures accident recognition.

CounterDrive	AUC↑				Acc↑
	@1s	@2s	@3s	@4s	
	0.49	0.52	0.51	0.50	0.51
✓	**0.96**	**0.95**	**0.96**	**0.93**	**0.96**

Can RISE Benefit Other Paradigm? To evaluate whether the proposed Scheduler generalizes beyond our base WAM, we reproduce DAWN Lu et al. (2026), a representative World–Action Interactive Model, and integrate the Scheduler without modifying its Predictor or Planner. As shown in Table 7, the resulting model improves PDMS from 89.1 to 90.3. Consistent gains are also observed in NC, DAC, EP, and TTC, with particularly clear improvements of 2.7 points in EP and 2.3 points in TTC, while retaining a perfect collision score. These results suggest that the Scheduler can serve as a plug-in adaptive computation module for different world–action modeling paradigms, rather than being tied to a specific WAM.

Table 7: Transfer of the RISE Scheduler to DAWN. Adding the Scheduler consistently improves planning performance without modifying the underlying WAIM.

Method	NC↑	DAC↑	EP↑	C↑	TTC↑	PDMS↑
DAWN	98.7	95.9	84.3	100	96.0	89.1
+ Scheduler	**99.9**	**96.8**	**87.0**	100	**98.3**	**90.3**

Can RISE Generate Plausible and Safe Trajectories? Fig. 5 compares RISE with Drive-JEPA across representative scenes. In the intersection and turning cases of Fig. 5(b)–(d), Drive-JEPA exhibits lateral or directional deviations from the local lane geometry, whereas RISE remains within feasible lane corridors and closely matches the human trajectory. In the curved-road cases of Fig. 5(a) and (e), RISE likewise follows the reference closely, including at $h = 0$, while Drive-JEPA

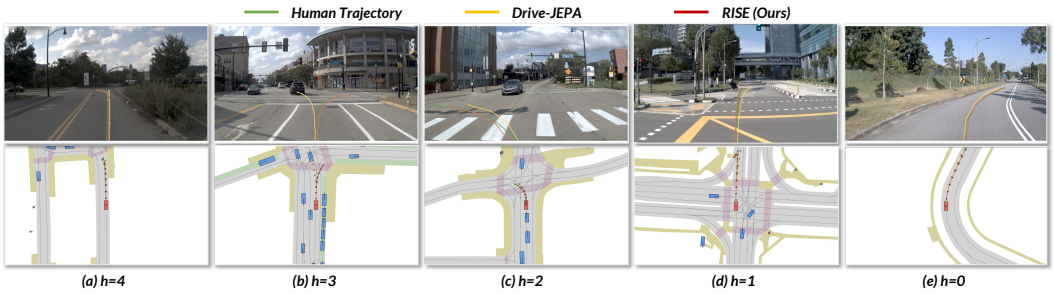

Figure 5: Qualitative trajectory comparison on NAVSIM at depths $h = 4$ to $h = 0$. The rows show front-camera views and corresponding bird's-eye-view scenes. Human, Drive-JEPA, and RISE trajectories are shown in green, yellow, and red, respectively. RISE remains aligned with human trajectories and feasible lane geometry across scene-dependent rollout depths.

shows greater lateral deviation. These examples show that RISE produces plausible, lane-consistent trajectories across varied road structures and selected rollout depths.

5 Related Works

5.1 World Action Models

World action models (WAMs) extend predictive world modeling Ha & Schmidhuber (2018); Zhou et al. (2024) with executable action generation. Rather than directly mapping observations to actions, they use future prediction to learn physical dynamics and state–action correspondences. Unified Video Action Model Li et al. (2025b) learns a shared video–action latent space for forward prediction, inverse dynamics, and real-time control. Cosmos Policy Kim et al. (2026) adapts pretrained video diffusion models to jointly predict future states, actions, and values. LingBot-VA Li et al. (2026), DreamZero Ye et al. (2026), MotuBrain Team et al. (2026), and WLA-0 Yang et al. (2026d) further scale WAMs toward language-conditioned, long-horizon, and cross-embodiment control. However, most existing WAMs treat test-time imagination as a fixed architectural choice.

5.2 Autonomous Driving World Models

Driving world models predict scene evolution from historical observations and ego actions Feng et al. (2025). ReSim Yang et al. (2026a) further studies reliable simulation for autonomous driving. GAIA-1 Hu et al. (2023) and DriveDreamer Wang et al. (2023) generate controllable futures from video, text, and action conditions. OccWorld Zheng et al. (2024) models future 3D occupancy, while HERMES Zhou et al. (2025) and UniFuture Liang et al. (2025) couple generation with scene understanding or geometric perception. DriveDreamer4D Zhao et al. (2025a) uses generated futures to improve 4D scene representations. More recent methods emphasize planning-oriented latent dynamics: Epona Zhang et al. (2025) integrates autoregressive diffusion with trajectory planning, whereas Latent-WAM Wang et al. (2026a), DriveFuture Hong et al. (2026), and DreamerAD Yang et al. (2026c) learn compact future states for end-to-end planning or reinforcement learning. Despite their different representations, these methods generally use a fixed imagination strategy, whereas RISE allocates future reasoning according to scene-dependent planning utility.

6 Conclusion

We presented RISE, a plug-in adaptive imagination framework that replaces fixed-depth WAM rollout with scene-dependent computation. Its Latent Evaluator estimates current risk and Future Planning Gain, allowing the Scheduler to make cost-aware sequential Roll/Stop decisions. We also introduced CounterDrive for counterfactual future learning and risk evaluation. Experiments on nuScenes and NAVSIM demonstrate competitive planning performance while balancing planning quality and inference cost.

REFERENCES

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.

Delong Chen, Mustafa Shukor, Theo Moutakanni, Willy Chung, Jade Yu, Tejaswi Kasarla, Yejin Bang, Allen Bolourchi, Yann LeCun, and Pascale Fung. Vl-jepa: Joint embedding predictive architecture for vision-language. *arXiv preprint arXiv:2512.10942*, 2025.

Yuntao Chen, Yuqi Wang, and Zhaoxiang Zhang. Drivinggpt: Unifying driving world modeling and planning with multi-modal autoregressive transformers, 2024. URL https://arxiv.org/abs/2412.18607.

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. *Advances in Neural Information Processing Systems*, 37:28706–28719, 2024.

Tuo Feng, Wenguan Wang, and Yi Yang. A survey of world models for autonomous driving, 2025. URL https://arxiv.org/abs/2501.11260.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

Yufeng Hong, Xiaotian Zhou, Yingyan Li, Xiangpo Zhou, Lin Liu, Yadan Luo, Shaoqing Xu, Lei Yang, and Ziying Song. Drivefuture: Future-aware latent world models for autonomous driving, 2026. URL https://arxiv.org/abs/2605.09701.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning, 2026. URL https://arxiv.org/abs/2601.16163.

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.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control, 2026. URL https://arxiv.org/abs/2601.21998.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model, 2025b. URL https://arxiv.org/abs/2503.00200.

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*, 2024.

Yingyan Li, Shuyao Shang, Weisong Liu, Bing Zhan, Haochen Wang, Yuqi Wang, Yuntao Chen, Xiaoman Wang, Yasong An, Chufeng Tang, Lu Hou, Lue Fan, and Zhaoxiang Zhang. Drivevla-w0: World models amplify data scaling law in autonomous driving, 2025c. URL https://arxiv.org/abs/2510.12796.

Dingkang Liang, Dingyuan Zhang, Xin Zhou, Sifan Tu, Tianrui Feng, Xiaofan Li, Yumeng Zhang, Mingyang Du, Xiao Tan, and Xiang Bai. Seeing the future, perceiving the future: A unified driving world model for future generation and perception. *arXiv preprint arXiv:2503.13587*, 2025.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22 (4):730–751, 2025.

Hongbo Lu, Liang Yao, Chenghao He, Haoyu Wang, Xiang Gu, Xianfei Li, Wenlong Liao, Tao He, and Pai Peng. The dawn of world-action interactive models, 2026. URL https://arxiv.org/abs/2605.11550.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Phuc Nguyen, Anh N Nhu, and Ming C Lin. Openvo: Open-world visual odometry with temporal dynamics awareness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14208–14218, 2026.

Shuyao Shang, Yuntao Chen, Yuqi Wang, Yingyan Li, and Zhaoxiang Zhang. Drivedpo: Policy learning via safety dpo for end-to-end autonomous driving. *arXiv preprint arXiv:2509.17940*, 2025.

Qiuhong Shen, Shihua Zhang, Yue Liao, Qi Li, Zhenxiong Tan, Shizun Wang, Shuicheng Yan, and Xinchao Wang. World action models: A survey, 2026. URL https://arxiv.org/abs/2606.20781.

MotuBrain Team, Chendong Xiang, Fan Bao, Haitian Liu, Hengkai Tan, Hongzhe Bi, James Li, Jiabao Liu, Jingrui Pang, Kiro Jing, Louis Liu, Mengchen Cai, Rongxu Cui, Ruowen Zhao, Runqing Wang, Shuhe Huang, Yao Feng, Yinze Rong, Zeyuan Wang, and Jun Zhu. Motubrain: An advanced world action model for robot control, 2026. URL https://arxiv.org/abs/2604.27792.

Linbo Wang, Yupeng Zheng, Qiang Chen, Shiwei Li, Yichen Zhang, Zebin Xing, Qichao Zhang, Xiang Li, Deheng Qian, Pengxuan Yang, Yihang Dong, Ce Hao, Xiaoqing Ye, Junyu han, Yifeng Pan, and Dongbin Zhao. Latent-wam: Latent world action modeling for end-to-end autonomous driving, 2026a. URL https://arxiv.org/abs/2603.24581.

Linhan Wang, Zichong Yang, Chen Bai, Guoxiang Zhang, Xiaotong Liu, Xiaoyin Zheng, Xiao-Xiao Long, Chang-Tien Lu, and Cheng Lu. Drive-jepa: Video jepa meets multimodal trajectory distillation for end-to-end driving. *arXiv preprint arXiv:2601.22032*, 2026b.

Siyin Wang, Junhao Shi, Zhaoyang Fu, Xinzhe He, Feihong Liu, Chenchen Yang, Yikang Zhou, Zhaoye Fei, Jingjing Gong, Jinlan Fu, Mike Zheng Shou, Xuanjing Huang, Xipeng Qiu, and Yu-Gang Jiang. World action models: The next frontier in embodied ai, 2026c. URL https://arxiv.org/abs/2605.12090.

Xiaofeng Wang, Zheng Zhu, Guan Huang, Xinze Chen, Jiagang Zhu, and Jiwen Lu. Drivedreamer: Towards real-world-driven world models for autonomous driving, 2023. URL https://arxiv.org/abs/2309.09777.

Tianze Xia, Yongkang Li, Lijun Zhou, Jingfeng Yao, Kaixin Xiong, Haiyang Sun, Bing Wang, Kun Ma, Guang Chen, Hangjun Ye, Wenyu Liu, and Xinggang Wang. Drivelaw:unifying planning and video generation in a latent driving world, 2026. URL https://arxiv.org/abs/2512.23421.

Jiawei Xu, Zhizhou Zhong, Zhijian Shu, Mingkai Jia, Mingxiao Li, Jia-Wang Bian, Qian Zhang, Kaicheng Zhang, Jin Xie, Jian Yang, and Wei Yin. Eponav2: Driving world model with comprehensive future reasoning, 2026. URL https://arxiv.org/abs/2605.14696.

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.

Pengxuan Yang, Yupeng Zheng, Deheng Qian, Zebin Xing, Qichao Zhang, Linbo Wang, Yichen Zhang, Shaoyu Guo, Zhongpu Xia, Qiang Chen, Junyu Han, Lingyun Xu, Yifeng Pan, and Dongbin Zhao. Dreamerad: Efficient reinforcement learning via latent world model for autonomous driving, 2026c. URL https://arxiv.org/abs/2603.24587.

Yi Yang, Zhihong Liu, Siqi Kou, Yiyang Chen, Yanzhe Hu, Jianbo Zhou, Boyuan Zhao, Zhijie Wei, Xiao Xia, Xueqi Li, Pengfei Liu, and Zhijie Deng. World-language-action model for unified world modeling, language reasoning, and action synthesis, 2026d. URL https://arxiv.org/abs/2606.05979.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 27220–27230, 2025.

Guosheng Zhao, Chaojun Ni, Xiaofeng Wang, Zheng Zhu, Xueyang Zhang, Yida Wang, Guan Huang, Xinze Chen, Boyuan Wang, Youyi Zhang, et al. Drivedreamer4d: World models are effective data machines for 4d driving scene representation. In *Proceedings of the computer vision and pattern recognition conference*, pp. 12015–12026, 2025a.

Zhida Zhao, Talas Fu, Yifan Wang, Lijun Wang, and Huchuan Lu. From forecasting to planning: Policy world model for collaborative state-action prediction. *arXiv preprint arXiv:2510.19654*, 2025b.

Wenzhao Zheng, Weiliang Chen, Yuanhui Huang, Borui Zhang, Yueqi Duan, and Jiwen Lu. Occworld: Learning a 3d occupancy world model for autonomous driving. In *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.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

Xin Zhou, Dingkang Liang, Sifan Tu, Xiwu Chen, Yikang Ding, Dingyuan Zhang, Feiyang Tan, Hengshuang Zhao, and Xiang Bai. Hermes: A unified self-driving world model for simultaneous 3d scene understanding and generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 27817–27827, 2025.

A AI USE STATEMENT

Generative AI tools aided this work by providing code-writing support, enhancing manuscript readability, facilitating information searches, and identifying relevant literature. All AI-assisted contributions, including code and literature, were thoroughly reviewed, verified, and manually inspected for correctness. We assume full responsibility for the final content, including all AI-generated text, claims, code, and other artifacts.

B LIMITATIONS

Although adaptive rollout is designed as a general principle for world action models, our experiments currently focus on autonomous driving, and its applicability to other domains remains to be explored. Furthermore, due to the cost of generating and filtering counterfactual samples, CounterDrive does not yet provide one-to-one coverage of the NAVSIM training set. Nevertheless, the primary goal of this work is to establish the necessity of scene-adaptive rollout, which is consistently supported by our experiments. Future work will extend the evaluation to broader domains and further scale the counterfactual dataset.

C IMPLEMENTATION DETAILS

Compute and data. We implement RISE in PyTorch. The WAM and diffusion planners are trained with bfloat16 precision on eight NVIDIA A100 GPUs using a per-GPU batch size of 4, giving an effective batch size of 32. The lightweight Evaluator and Rollout Gate are trained offline in FP32 on one A100. We use four data-loading workers per process with pinned and persistent workers. Front-camera videos are sampled at 2 Hz and resized to 256×512. Each clip contains four observed frames. With a spatial patch size of 16 and a temporal tubelet size of 2, the four observations correspond to two observed latent steps. We set the maximum rollout horizons to $H_{\mathrm{NAVSIM}} = 4$ and $H_{\mathrm{nuScenes}} = 3$. Horizontal flipping, motion shifting, GridMask, AutoAugment, and random erasing are disabled.

World Action Model. The visual encoder is a Drive-JEPA ViT-L/16 with causal attention. The action-conditioned Predictor contains 12 Transformer blocks with a hidden dimension of 384, 12 attention heads, rotary positional embeddings, and activation checkpointing. Each rollout step is conditioned on a three-dimensional ego-action vector. Ego-state conditioning, route-command conditioning, camera extrinsics, and parallel future prediction are disabled. A frozen Token-AE represents each latent frame using 128 tokens arranged on an 8×16 latent grid. It uses a four-layer encoder, a two-layer decoder, and a two-layer causal temporal module with 16 attention heads and zero dropout. During Planner, Evaluator, and Scheduler training, the visual encoder and Predictor remain frozen.

Dynamic planners. Both Π_0 and Π_1 use the same diffusion Transformer architecture with 12 layers, a hidden dimension of 384, 12 attention heads, an MLP ratio of 4, and zero dropout. The variance-preserving diffusion process uses $\beta_{\min} = 0.1$ and $\beta_{\max} = 20$. At inference, the Planner performs 20 denoising steps and generates $K = 6$ trajectory modes. Each pose is represented by $(x, y, \cos\theta, \sin\theta)$ with a temporal interval of 0.5 s. The classification, regression, velocity, and yaw losses are weighted by 1.0, 1.0, 0.5, and 0.5, respectively. The confidence temperature is 1.5, with classification and ignore thresholds of 2.0 and 0.2. The adaptive winner-take-all temperature starts at 8.0, decays with a factor of 0.984, and is lower-bounded by 0.1. During dynamic-prefix training, the full prefix $h = H$ is sampled with probability 0.25, while the remaining probability is distributed uniformly over $h \in \{0, \ldots, H-1\}$.

Planner optimization. We optimize both planners using AdamW with $\boldsymbol{\beta} = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and a weight decay of 0.04. For Π_0, training lasts 20 epochs; the learning rate is linearly warmed up from 2×10^{-6} to 2×10^{-5} during the first two epochs and then cosine-decayed to zero. For Π_1, training lasts 50 epochs; the learning rate is warmed up from 10^{-5} to 5×10^{-5} during the first five epochs and then cosine-decayed to zero. We use deterministic execution and fixed diffusion noise for validation.

Evaluator and latent guidance. The Evaluator first mean-pools the spatial tokens of each latent frame, applies LayerNorm, and encodes the prefix using a one-layer causal GRU with a hidden dimension of 512 and zero dropout. Two linear heads produce its task-evaluation profiles. The initial warm-up is performed for one epoch with 256 updates, a batch size of 4, and a learning rate of 10^{-4}. Real samples use a Huber loss with $\delta = 1$, while counterfactual hazard and trajectory-quality ranking losses use unit weights and a margin of 1.0. Local calibration uses one epoch with 256 updates, a batch size of 1, and a learning rate of 5×10^{-5}. For each latent prefix, we sample four Gaussian perturbations with a scale of 0.05 and clip their per-token norm to 0.25; the local ordering loss has weight 1.0 and margin 0.1. The second profile head is subsequently calibrated for one epoch with a batch size of 1 and a learning rate of 5×10^{-5}, while the shared prefix encoder is frozen. All Evaluator stages use AdamW with the same $\boldsymbol{\beta}$, ϵ, and weight decay as the planners.

Rollout Gate. The Rollout Gate applies LayerNorm followed by two 128-dimensional fully connected layers with GELU activations. It is distilled from complete horizon-wise utility curves for 50 epochs using AdamW, a batch size of 1, a constant learning rate of 10^{-3}, and a weight decay of 0.04. Its classification loss uses a temperature of 0.05, and the Smooth-L1 utility-regression term is weighted by 0.5. We define normalized rollout costs as $c_h = h$ and train the conditional Gate with $\lambda \in \{0, 0.001, 0.005, 0.01, 0.05\}$, using $\lambda = 0.005$ by default. Latent guidance performs two gradient steps on the latest imagined prefix with a step size of 0.05 and a maximum update norm of 0.25. The guided latent is detached before being passed to the Planner. Unless otherwise specified, all experiments use random seed 239.

D More Training Details of RISE

Notation and conventions. We use H for the abstract maximum latent rollout horizon and $H_{\mathcal{D}}$ for its dataset-specific realization. In our experiments, $H_{\text{NAVSIM}} = 4$ and $H_{\text{nuScenes}} = 3$. We use P for the number of future poses in a planned trajectory and J for the number of Planner modes. The indices $h \in \{0, \dots, H_{\mathcal{D}}\}$, $p \in \{1, \dots, P\}$, and $s \in \{1, \dots, J\}$ denote rollout depth, trajectory pose, and Planner mode, respectively. Index i denotes a training sample or, when $i \in \mathcal{I}_{\text{pair}}$, a verified factual–counterfactual source pair; j denotes a valid continuation depth, and m denotes a risk-refinement iteration. The real and counterfactual datasets are denoted by $\mathcal{D}_{\text{r}}$ and $\mathcal{D}_{\text{c}}$.

The Predictor, initial Planner, final Planner, Latent Evaluator, Rollout Gate, risk evaluator, and planning-score evaluator are denoted by $\mathcal{P}$, Π_0, Π_1, Eval_{θ_E}, G_{θ_G}, $\mathcal{E}_{\text{risk}}$, and $\mathcal{E}_{\text{plan}}$, respectively. The relative-pose action mapping and token-pooling operator are denoted by $\mathcal{A}$ and ρ. We use $|\mathcal{S}|$ for the cardinality of a set $\mathcal{S}$, $\mathbb{I}[\cdot]$ for an indicator function, $[a]_+ = \max(a, 0)$, $\sigma(\cdot)$ for the logistic sigmoid, and $\text{sg}(\cdot)$ for stop-gradient.

All scalar regression losses use the Huber function

$$\ell_\kappa(a, b) = \begin{cases} \frac{1}{2}(a-b)^2, & |a-b| \le \kappa, \\ \kappa\left(|a-b| - \frac{\kappa}{2}\right), & \text{otherwise}, \end{cases} \tag{21}$$

where $\kappa > 0$ is the transition point. We use $\kappa = 1$ in all experiments.

D.1 Stage I: Predictor and Initial Planner

Future-latent prediction. For each sample, the frozen target Encoder produces the future latent target z_{t+h}. Starting from the observed latent, the Predictor recursively generates the maximum future prefix $\hat{Z}_{H_{\mathcal{D}}}$. The per-step latent loss is

$$\ell_z(\hat{z}, z) = \frac{\|\text{LN}(\hat{z}) - \text{sg}(\text{LN}(z))\|_1}{N_{\text{tok}} D_z}, \tag{22}$$

where LN denotes token-wise LayerNorm, N_{tok} is the number of tokens in one latent step, and D_z is their embedding dimension. The Predictor objective is

$$\mathcal{L}_{\text{Pre}} = \frac{1}{H_{\mathcal{D}}} \sum_{h=1}^{H_{\mathcal{D}}} \ell_z\left(\hat{z}_{t+h}, z_{t+h}\right). \tag{23}$$

This objective directly supervises the autoregressive rollout and is averaged over the minibatch. Real and accepted CounterDrive samples use the same loss. For CounterDrive, the generated future video is encoded into z_{t+h}, while relative pose changes between adjacent recovered ego poses provide the ego-motion conditioning. At inference, the same representation is computed only from adjacent poses in the observed ego-motion history; no future ground-truth pose is used. No route command is used.

Diffusion Planner objective. At training time, we sample a diffusion time $u \sim \mathcal{U}(0,1)$ and Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. Under the variance-preserving schedule, the noisy trajectory representation is

$$\tau_i^{(u)} = \alpha_u \tau_i^* + \sigma_u \boldsymbol{\epsilon}, \qquad \alpha_u^2 + \sigma_u^2 = 1. \tag{24}$$

Conditioned on $\tau_i^{(u)}$, u, the observed latent, and the available future prefix, the Planner predicts clean trajectory candidates and their confidence logits. The losses below are evaluated on these denoised predictions, and the expectation over u and $\boldsymbol{\epsilon}$ is approximated by minibatch sampling. The Planner additionally conditions on the observed ego-motion history and current ego kinematics; for compactness, these two inputs are suppressed in all subsequent $\Pi_0(\cdot)$ and $\Pi_1(\cdot)$ notation.

The Planner produces J candidate trajectories. Let $\hat{\tau}_{i,s}$ denote candidate s for sample i, and let τ_i^* denote the corresponding ground-truth trajectory. Each trajectory contains P poses represented by $(x, y, \cos\theta, \sin\theta)$.

For a Planner minibatch of size B_{pla}, the displacement error of mode s is

$$d_{i,s} = \frac{1}{P} \sum_{p=1}^{P} \left\| \hat{\mathbf{p}}_{i,s,p} - \mathbf{p}_{i,p}^* \right\|_2, \tag{25}$$

where $\hat{\mathbf{p}}_{i,s,p} = (\hat{x}_{i,s,p}, \hat{y}_{i,s,p})$ and $\mathbf{p}_{i,p}^* = (x_{i,p}^*, y_{i,p}^*)$.

The annealed winner-take-all weight is

$$\pi_{i,s}^{\text{aw}} = \text{sg}\left(\frac{\exp(-d_{i,s}/T_{\text{aw}})}{\sum_{s'=1}^{J} \exp(-d_{i,s'}/T_{\text{aw}})} \right), \tag{26}$$

where $T_{\text{aw}} > 0$ is the annealed mode temperature. The position regression loss is

$$\mathcal{L}_{\text{xy}} = \frac{1}{B_{\text{pla}} P} \sum_{i=1}^{B_{\text{pla}}} \sum_{s=1}^{J} \sum_{p=1}^{P} \pi_{i,s}^{\text{aw}} \cdot \left[\ell_\kappa(\hat{x}_{i,s,p}, x_{i,p}^*) + \ell_\kappa(\hat{y}_{i,s,p}, y_{i,p}^*) \right]. \tag{27}$$

Let

$$\begin{aligned} \hat{\mathbf{y}}_{i,s,p} &= (\widehat{\cos\theta}_{i,s,p}, \widehat{\sin\theta}_{i,s,p}), \\ \mathbf{y}_{i,p}^* &= (\cos\theta_{i,p}^*, \sin\theta_{i,p}^*). \end{aligned} \tag{28}$$

The yaw consistency loss is

$$\mathcal{L}_{\text{yaw}} = \frac{1}{2 B_{\text{pla}} P} \sum_{i=1}^{B_{\text{pla}}} \sum_{s=1}^{J} \sum_{p=1}^{P} \pi_{i,s}^{\text{aw}} \cdot \left(1 - \frac{\langle \hat{\mathbf{y}}_{i,s,p}, \mathbf{y}_{i,p}^* \rangle}{\|\hat{\mathbf{y}}_{i,s,p}\|_2 \|\mathbf{y}_{i,p}^*\|_2} \right). \tag{29}$$

Let $g_{i,s}$ be the predicted confidence logit of mode s. Its soft target is

$$\pi_{i,s}^{\text{mode}} = \text{sg}\left(\frac{\exp(-d_{i,s}/T_{\text{conf}})}{\sum_{s'=1}^{J} \exp(-d_{i,s'}/T_{\text{conf}})} \right), \tag{30}$$

where $T_{\text{conf}} > 0$ is the confidence temperature. Define

$$\begin{aligned} d_i^\star &= \min_s d_{i,s}, \\ s_i^\star &= \arg\min_s d_{i,s}, \\ \mathcal{I}_{\text{mode}} &= \{ i : d_i^\star < \xi_{\text{cls}} \}, \\ \mathcal{J}_i &= \{ s : d_{i,s} - d_i^\star > \xi_{\text{ign}} \} \cup \{ s_i^\star \}, \end{aligned} \tag{31}$$

where ξ_{cls} removes samples for which no mode is sufficiently close to the target, and ξ_{ign} removes ambiguous non-winning modes. The mode-confidence loss is

$$\mathcal{L}_{\text{mode}} = -\frac{1}{|\mathcal{I}_{\text{mode}}|} \sum_{i \in \mathcal{I}_{\text{mode}}} \sum_{s \in \mathcal{J}_i} \pi_{i,s}^{\text{mode}} \log \frac{\exp(g_{i,s})}{\sum_{s'=1}^{J} \exp(g_{i,s'})}. \tag{32}$$

If $\mathcal{I}_{\text{mode}}$ is empty, this term is set to zero. The complete trajectory objective is

$$\ell_\tau = \beta_{\text{xy}} \mathcal{L}_{\text{xy}} + \beta_{\text{yaw}} \mathcal{L}_{\text{yaw}} + \beta_{\text{mode}} \mathcal{L}_{\text{mode}}, \tag{33}$$

where the expectation over sampled diffusion time and noise is implemented by the corruption process in Eq. equation 24. Since our trajectory representation contains no velocity channels, no velocity loss is used.

Initial variable-prefix Planner. The Encoder and Predictor are frozen before Planner training. For each real sample, the Predictor produces all prefixes $\{\hat{Z}_h\}_{h=0}^{H_{\mathcal{D}}}$, where $\hat{Z}_h = \hat{z}_{t+1:t+h}$ and $\hat{Z}_0 = \emptyset$. We sample h uniformly and optimize

$$\mathcal{L}_{\text{Pla}}^0 = \ell_\tau \left(\Pi_0(z_t, \hat{Z}_h), \tau^* \right), \qquad h \sim \mathcal{U}\{0, \ldots, H_{\mathcal{D}}\}. \tag{34}$$

Only real samples provide Planner trajectory supervision.

D.2 Stage II: Latent Evaluator and Guided Planner

Geometry-based risk and planning scores. For each Planner output, we select the candidate with the highest predicted confidence. On real data, the fixed risk evaluator returns

$$\begin{aligned} r &= \mathcal{E}_{\text{risk}}(\tau, \tau^*) \\ &= \frac{4R_{\text{col}} + R_{\text{near}} + R_{\text{traj}} + 0.2R_{\text{comf}}}{6.2}, \end{aligned} \tag{35}$$

where R_{col}, R_{near}, R_{traj}, and R_{comf} are the collision, near-miss, normalized trajectory-error, and comfort risks. Every component and the resulting risk lie in $[0, 1]$, with larger values indicating less desirable trajectories. The corresponding higher-is-better planning score is

$$q = \mathcal{E}_{\text{plan}}(\tau, \tau^*) = 1 - \mathcal{E}_{\text{risk}}(\tau, \tau^*). \tag{36}$$

The trajectory-error normalization and geometric checks follow the native planning horizon and coordinate system of each dataset. Both evaluators are fixed, used only to construct targets, and never differentiated through.

Latent Evaluator. Given the observed latent z_t and prefix $\hat{Z}_h$, the Latent Evaluator predicts

$$\begin{aligned} (\mathbf{R}_h, \mathbf{B}_h) &= \text{Eval}_{\theta_E}(z_t, \hat{Z}_h), \\ \mathbf{R}_h &= [r_1, \ldots, r_h], \\ \mathbf{B}_h &= [b_{h \to h+1}, \ldots, b_{h \to H_{\mathcal{D}}}], \end{aligned} \tag{37}$$

where $\mathbf{R}_h$ is the Risk Profile and $\mathbf{B}_h$ is the Future Planning Gain Profile. Each r_k estimates the trajectory risk associated with the prefix ending at depth k, while $b_{h \to j}$ estimates the planning-score change from continuing to depth j relative to stopping with the current prefix at depth h. For $h = 0$, $\mathbf{R}_0$ is empty and $\mathbf{B}_0$ is predicted from the observed latent together with the learned empty-prefix embedding.

The Evaluator spatially averages the tokens of each latent step, applies LayerNorm, and processes the resulting sequence with a one-layer causal GRU. A risk projection produces the scalar estimate associated with each available prefix. A gain projection produces an $H_{\mathcal{D}}$-dimensional vector from the current recurrent state; entries for invalid continuation depths are masked. Consequently, all predictions depend only on the prefix available at the current rollout step.

Real Risk Profile regression. For real sample i and prefix ending at depth k, the target is

$$r_{i,k}^* = \mathcal{E}_{\text{risk}} \left(\Pi_0(z_{i,t}, \hat{Z}_{i,k}), \tau_i^* \right). \tag{38}$$

For a real-data minibatch of size B_{r}, we optimize

$$\mathcal{L}_{\text{real}} = \frac{1}{B_{\text{r}} H_{\mathcal{D}}} \sum_{i=1}^{B_{\text{r}}} \sum_{k=1}^{H_{\mathcal{D}}} \ell_\kappa \left(r_{i,k}, r_{i,k}^* \right). \tag{39}$$

Verified counterfactual risk ranking. CounterDrive does not provide reliable future-agent geometry and therefore does not receive a scalar target from $\mathcal{E}_{\text{risk}}$. Let $\mathcal{I}_{\text{pair}}$ denote the verified factual–counterfactual source pairs for which the counterfactual future contains an annotated incident within the modeled horizon and is ranked as riskier than its factual source. Pairing is defined only within the selected source subset and does not imply one-to-one coverage of either original dataset.

For pair i, the annotated incident frame is temporally aligned with latent step k_i^{inc}. We define

$$m_{i,k} = \mathbb{I}\left[k \geq k_i^{\text{inc}}\right]. \tag{40}$$

Only risk estimates at and after the incident onset contribute to the ranking objective:

$$\mathcal{L}_{\text{rank}} = \frac{1}{|\mathcal{I}_{\text{pair}}|} \sum_{i \in \mathcal{I}_{\text{pair}}} \frac{1}{\sum_{k=1}^{H_{\mathcal{D}}} m_{i,k}} \sum_{k=1}^{H_{\mathcal{D}}} m_{i,k} \left[\gamma + r_{i,k}^{\text{fac}} - r_{i,k}^{\text{cf}}\right]_+. \tag{41}$$

Thus, a hazardous counterfactual is assigned higher predicted risk than its factual source after the verified incident becomes observable. Real samples without a verified counterfactual partner are excluded only from this ranking term.

Temporally localized risk calibration. The onset mask separates supervision before and after the incident becomes observable. While $\mathcal{L}_{\text{rank}}$ enforces higher counterfactual risk at and after the onset, the local term aligns the paired risks beforehand:

$$\mathcal{L}_{\text{loc}} = \frac{1}{|\mathcal{I}_{\text{pair}}|} \sum_{i \in \mathcal{I}_{\text{pair}}} \frac{\sum_{k=1}^{H_{\mathcal{D}}} (1 - m_{i,k}) \ell_\kappa \left(r_{i,k}^{\text{fac}}, r_{i,k}^{\text{cf}}\right)}{\max\left(1, \sum_{k=1}^{H_{\mathcal{D}}} (1 - m_{i,k})\right)}. \tag{42}$$

This prevents the counterfactual prefix from being labeled as riskier before the annotated incident is revealed and does not require geometry targets for generated CounterDrive clips. The complete Risk Profile objective is

$$\mathcal{L}_R = \mathcal{L}_{\text{real}} + \beta_{\text{cf}} \mathcal{L}_{\text{rank}} + \beta_{\text{loc}} \mathcal{L}_{\text{loc}}. \tag{43}$$

Risk-guided latent refinement. After Risk Profile training, the risk-prediction branch is frozen. For a non-empty prefix, let $\boldsymbol{\epsilon}_h^m$ be the refinement residual at iteration m, initialized by $\boldsymbol{\epsilon}_h^0 = 0$. Token-wise norm clipping is

$$\text{clip}_{\varepsilon_{\text{r}}}(\mathbf{a}) = \begin{cases} \mathbf{0}, & \|\mathbf{a}\|_2 = 0, \\ \mathbf{a} \min\left(1, \frac{\varepsilon_{\text{r}}}{\|\mathbf{a}\|_2}\right), & \text{otherwise}, \end{cases} \tag{44}$$

where ε_{r} is the refinement trust radius. We reduce the accumulated prefix risk through

$$\boldsymbol{\epsilon}_h^{m+1} = \text{clip}_{\varepsilon_{\text{r}}}\left(\boldsymbol{\epsilon}_h^m - \eta \nabla_{\boldsymbol{\epsilon}_h^m} \sum_{k=1}^{h} r_k\left(z_t, \hat{Z}_h + \boldsymbol{\epsilon}_h^m\right)\right). \tag{45}$$

After M_{r} iterations, the risk-refined prefix is

$$\tilde{Z}_h = \hat{Z}_h + \boldsymbol{\epsilon}_h^{M_{\text{r}}}. \tag{46}$$

Refinement is skipped at $h = 0$, for which $\tilde{Z}_0 = \hat{Z}_0 = \emptyset$.

Final Planner training. We initialize Π_1 from Π_0 and train it on risk-refined real prefixes:

$$\mathcal{L}_{\text{Pla}}^1 = \ell_\tau\left(\Pi_1(z_t, \tilde{Z}_h), \tau^*\right), \qquad h \sim \mathcal{U}\{0, \ldots, H_{\mathcal{D}}\}. \tag{47}$$

The refined prefix is detached before entering Π_1. Therefore, $\mathcal{L}_{\text{Pla}}^1$ updates only Π_1.

Future Planning Gain supervision. For every real sample, we evaluate the final Planner at each valid rollout depth:

$$q_{i,h} = \mathcal{E}_{\text{plan}}\left(\Pi_1(z_{i,t}, \tilde{Z}_{i,h}), \tau_i^*\right). \tag{48}$$

The realized Future Planning Gain Profile at depth h is

$$\mathbf{B}^*_{i,h} = [q_{i,h+1} - q_{i,h}, \ldots, q_{i,H_{\mathcal{D}}} - q_{i,h}]. \tag{49}$$

The gain branch is trained only on valid continuation entries:

$$\mathcal{L}_B = \frac{1}{B_{\text{r}}} \sum_{i=1}^{B_{\text{r}}} \sum_{h=0}^{H_{\mathcal{D}}-1} \frac{1}{H_{\mathcal{D}} - h} \sum_{j=h+1}^{H_{\mathcal{D}}} \ell_\kappa \left(b_{i,h\to j}, q_{i,j} - q_{i,h}\right). \tag{50}$$

These targets compare every valid continuation with the planning score already available from the current prefix; they do not supervise a final rollout depth directly.

D.3 Stage III: Rollout Gate

Cost-adjusted continuation target. Gate supervision uses real samples and their all-horizon planning scores. Given cumulative rollout cost c_h and computation preference $\lambda \geq 0$, the best remaining cost-adjusted gain from depth h is

$$g^*_{i,h} = \max_{j \in \{h+1,\ldots,H_{\mathcal{D}}\}} \left[(q_{i,j} - q_{i,h}) - \lambda(c_j - c_h)\right]. \tag{51}$$

The binary continuation target is

$$y^*_{i,h} = \mathbb{I}\left[g^*_{i,h} > 0\right], \qquad 0 \leq h < H_{\mathcal{D}}. \tag{52}$$

A tie is assigned to Stop, and no Gate target is constructed at $h = H_{\mathcal{D}}$ because rollout terminates by construction.

Online Gate input. For one sample and $0 \leq h < H_{\mathcal{D}}$, the pooled latent feature and rollout-state feature are

$$\begin{aligned} e_h &= [\rho(z_t), \rho(\hat{Z}_h)], \\ \xi_h &= [\overline{\mathbf{R}}_h, \overline{\mathbf{B}}_h, h/H_{\mathcal{D}}, c_h, \Delta c_{h+1}], \end{aligned} \tag{53}$$

where ρ denotes token pooling, $\rho(\emptyset)$ is a learned empty-prefix embedding, $\overline{\mathbf{R}}_h$ and $\overline{\mathbf{B}}_h$ are zero-padded to $H_{\mathcal{D}}$ entries, and $\Delta c_{h+1} = c_{h+1} - c_h$. Given λ, the Gate predicts

$$x_h = G_{\theta_G}(e_h, \xi_h, \lambda). \tag{54}$$

The sign of x_h indicates whether the predicted planning gain available from continued rollout justifies the additional cost.

Gate objective. Let Λ be the set of computation preferences used for Gate training and define

$$\Omega_G = \{(i, h, \lambda) : 1 \leq i \leq B_G,\ 0 \leq h < H_{\mathcal{D}},\ \lambda \in \Lambda\}. \tag{55}$$

With $N_G = |\Omega_G|$, the Gate objective is

$$\mathcal{L}_G = \frac{1}{N_G} \sum_{(i,h,\lambda)\in\Omega_G} \ell_{\text{BCE}}\left(\sigma(x_{i,h}), y^*_{i,h}\right). \tag{56}$$

Full-horizon enumeration is used only to construct training targets. At inference, the Gate observes only the currently available prefix, appends exactly one latent after each Roll decision, and reevaluates the extended prefix.

Hyperparameters and optimization. We use $P = 8$ trajectory poses, $J = 6$ Planner modes, and $M_{\mathrm{r}} = 2$ risk-refinement iterations. The Planner weights are $\beta_{\mathrm{xy}} = 1$, $\beta_{\mathrm{yaw}} = 0.5$, and $\beta_{\mathrm{mode}} = 1$. We use $T_{\mathrm{conf}} = 1.5$, $\xi_{\mathrm{cls}} = 2.0$, and $\xi_{\mathrm{ign}} = 0.2$. The aWTA temperature starts from 8.0, is multiplied by 0.984 after each epoch, and is lower-bounded by 0.1.

For Risk Profile training, we use $\gamma = 1$, $\beta_{\mathrm{cf}} = 1$, and $\beta_{\mathrm{loc}} = 1$. Risk refinement uses $\eta = 0.05$ and $\varepsilon_{\mathrm{r}} = 0.25$. For Gate training, $c_h = h$ and

$$\Lambda = \{0, 0.001, 0.005, 0.01, 0.05\}. \tag{57}$$

The default inference preference is $\lambda = 0.005$.

All experiments use a per-GPU batch size of 4. The Predictor is trained for 80 epochs with learning rate 2×10^{-4}. The initial Planner Π_0 is trained for 20 epochs with learning rate 2×10^{-5}, and Π_1 is trained for 50 epochs with learning rate 5×10^{-5}. The Latent Evaluator and Rollout Gate are trained for 50 epochs with learning rate 10^{-3}. All stages use AdamW, weight decay 0.04, and bfloat16 precision.

CounterDrive participates in Predictor training and, only for verified factual–counterfactual source pairs, in $\mathcal{L}_{\mathrm{rank}}$ and $\mathcal{L}_{\mathrm{loc}}$. Unpaired factual samples remain available for the standard real-data objectives. Final Planner, Future Planning Gain, and Rollout Gate supervision use real planning outcomes.

E More Experimental Results

E.1 Is Adaptive Rollout Necessary? (Detail Results)

Table 8 provides a detailed comparison between globally fixed rollout depths and the adaptive policy of RISE. For fixed rollout, EPDMS increases from 88.7 at $h = 0$ to a peak of 90.1 at $h = 3$, but drops to 89.8 at $h = 4$, while latency grows monotonically from 93.975 ms to 426.483 ms. This shows that increasing rollout depth does not yield monotonic planning improvements and incurs substantial additional computation. In contrast, RISE achieves 90.8 EPDMS with 287.429 ms latency, outperforming the best fixed-depth configuration ($h = 3$) by 0.7 EPDMS while reducing latency by 9.8%. Compared with full rollout at $h = 4$, RISE further improves EPDMS by 1.0 point while reducing latency by 32.6%. The preferred-depth distribution is also spread across all rollout depths, with no single depth dominating all scenes, further illustrating the heterogeneity of rollout utility across driving contexts. These results show that adaptive rollout provides a more favorable planning–computation trade-off than using a globally fixed imagination budget.

Table 8: **Detailed analysis of fixed rollout depths and scene-dependent preferences on NAVSIM.** Rows are indexed by rollout depth h. *Overall EPDMS* reports the dataset-wide performance when all scenes use the same fixed depth h. *Ratio* and *# Scenes* describe the subset whose preferred rollout depth is $h^* = h$, while the last five columns report the EPDMS of each subset evaluated at fixed depths 0–4. Bold values in these columns indicate the preferred depth of each subset.

h	Overall EPDMS ↑	Latency (ms) ↓	Ratio	# Scenes	EPDMS at Fixed Rollout Depth				
					0	1	2	3	4
0	88.7	93.975	0.1027	1,248	**89.9**	87.8	88.2	88.5	88.4
1	89.4	167.682	0.2285	2,775	88.2	**90.8**	89.1	89.4	89.3
2	89.6	227.653	0.1570	1,907	88.4	88.9	**91.0**	89.6	89.5
3	90.1	318.554	0.3323	4,036	88.9	89.4	89.8	**91.5**	90.0
4	89.8	426.483	0.1795	2,180	88.5	89.0	89.4	89.7	**91.1**
RISE	**90.8**	287.429	-	-	-	-	-	-	-

E.2 How Does Rollout Depth Affect Different Scenes?

Figure 6 compares trajectories generated at different fixed rollout depths. In the first two simple lane-following scenes, predictions at all depths nearly overlap, indicating that additional rollout has little influence when the current observation already supports a stable plan. In contrast, the turning and intersection cases exhibit clear differences in trajectory curvature and direction across rollout depths. This comparison shows that sensitivity to rollout depth is scene-dependent: simple scenes remain stable across depths, whereas complex scenes require an appropriate rollout budget.

E.3 Latent Evaluator and Rollout Gate

Table 9: Ablation of the Latent Evaluator and Rollout Gate on NAVSIM with all other components fixed.

Latent Evaluator	**Gate**	**EPDMS↑**	**PDMS↑**
		89.8	90.5
	✓	90.6	91.2
✓		90.3	90.9
✓	✓	**90.8**	**91.5**

Table 9 separates the contributions of the two Scheduler components. Adding the Latent Evaluator alone improves EPDMS/PDMS from 89.8/90.5 to 90.3/90.9, while the Gate alone raises them to 90.6/91.2. Combining both components gives the best result of 90.8/91.5, showing that evaluating predicted futures and adaptively allocating rollout provide complementary improvements.

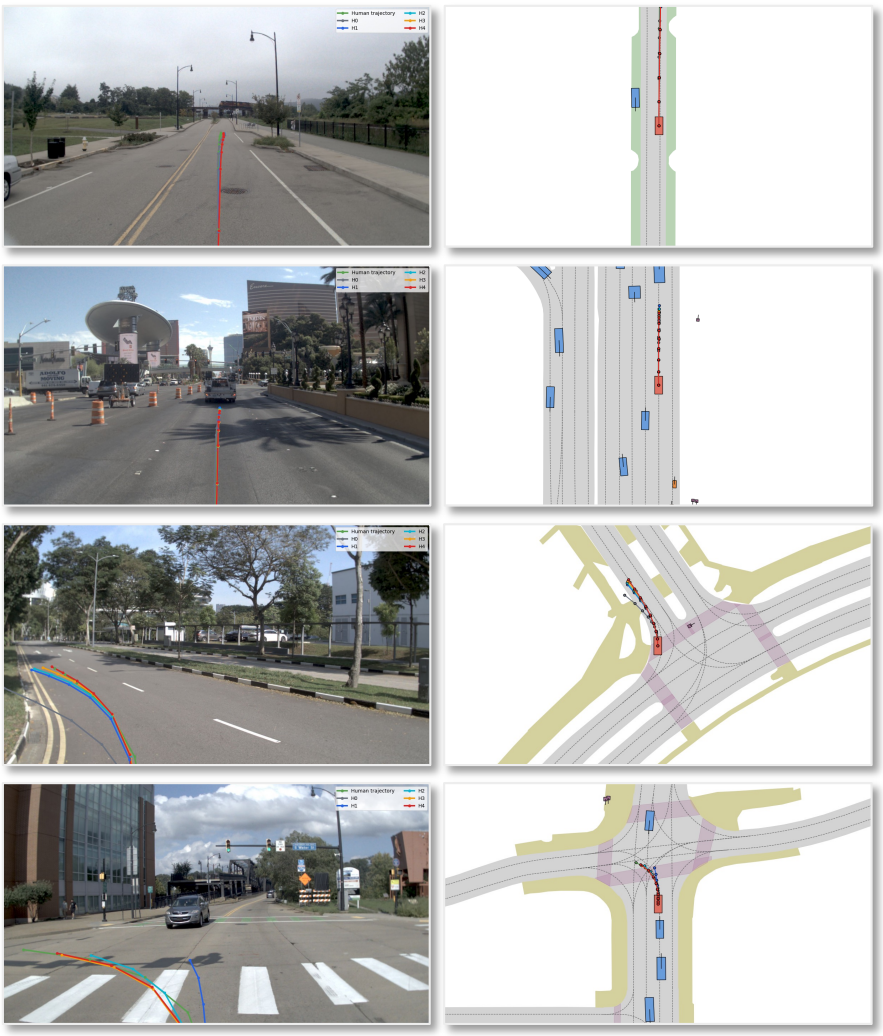

Figure 6: Trajectory predictions at different fixed rollout depths. Predictions remain nearly identical in simple scenes but vary noticeably in turning and intersection cases.

E.4 Variable-Prefix Planner Training Distribution

Table 10: Ablation of prefix-depth sampling for variable-prefix Planner training on NAVSIM. The distributions correspond to $h = 0, \ldots, 4$.

Strategy	**Training Distribution**	**EPDMS↑**
Uniform	$[0.20, 0.20, 0.20, 0.20, 0.20]$	**89.8**
Extremes	$[0.50, 0, 0, 0, 0.50]$	87.4
Short-heavy	$[0.225, 0.225, 0.225, 0.225, 0.10]$	89.6
No-full	$[0.25, 0.25, 0.25, 0.25, 0]$	89.2

Table 10 examines how the sampled prefix distribution affects variable-prefix Planner training. Uniform sampling performs best, while training only on the two extreme depths causes a substantial drop to 87.4 EPDMS. Short-heavy sampling remains competitive, but excluding the full prefix also degrades performance, indicating that balanced coverage of all valid depths is important.

E.5 Number of Risk-Refinement Steps

Table 11: Effect of the number of risk-refinement steps on NAVSIM.

Steps m	**PDMS**↑	**EPDMS**↑
0	90.5	89.8
1	90.7	90.1
2	**90.9**	**90.2**
4	**90.9**	**90.2**
8	**90.9**	**90.2**

As shown in Table 11, one refinement step already improves both metrics, and two steps increase PDMS/EPDMS from 90.5/89.8 to 90.9/90.2. Increasing m to 4 or 8 provides no further gain, so we use two steps to avoid redundant optimization.

E.6 Effect of the Computation-Cost Weight

Table 12: Effect of the computation-cost weight λ on NAVSIM planning performance.

λ	**EPDMS**↑
0	90.0
0.001	90.7
0.005	**90.8**
0.01	88.6
0.05	88.4

Table 12 shows that a modest computation penalty improves EPDMS from 90.0 at $\lambda = 0$ to 90.8 at $\lambda = 0.005$. Larger values substantially reduce planning performance, consistent with over-penalizing continued rollout. We therefore use $\lambda = 0.005$ as the default setting.

F SAMPLES OF COUNTERDRIVE



Figure 7: Sample 1 of CounterDrive.



Figure 8: Sample 2 of CounterDrive.

G PROMPTS FOR COUNTERDRIVE GENERATION

CounterDrive constructs each video-generation prompt in two steps. First, a vision-language model (VLM) describes the near-field content of the source key frame and generates a one-sentence accident description. Second, these two variable components are concatenated with a manually written camera constraint prompt and supplied to Wan together with the source key frame. The operational prompts are written in Chinese; we provide faithful English translations below.

G.1 VLM-BASED PROMPT CONSTRUCTION

Table 13 presents the instructions used to generate the two variable prompt components. The key-frame description is restricted to nearby visible objects so that the subsequent accident description remains grounded in the source scene. The accident description explicitly specifies both the relative accident location and the involved object.

Table 13: VLM instructions for constructing the variable components of the Wan prompt.

Component	Prompt content and constraints
A. Near-field key-frame description	
VLM input	One front-facing dashcam key frame.
Role	You are a near-field object-description assistant for dashcam images. Given a front-facing dashcam image, describe only objects immediately surrounding the camera in exactly one sentence. Do not analyze accidents and do not describe the ego vehicle.
Spatial range	Only consider objects within approximately 30 m in front of the camera and 20 m on either side of the road.
Include	Include vehicles immediately ahead using their vehicle type only, such as "double-decker bus" or "SUV." Also include nearby roadside objects, such as trees, guardrails, building facades, streetlights, or traffic signs. Mention the nearby road surface only when necessary.
Exclude	Do not mention city names, street names, distant landmarks, distant buildings, weather, illumination, time of day, detailed traffic conditions, or inferred geographical locations.
Output format	Return exactly one plain sentence. Do not output JSON, Markdown, numbering, explanations, or additional analysis.
Example output	"An SUV is immediately ahead, and a sedan is traveling in the adjacent left lane."
B. Accident description	
VLM input	The source key frame and its near-field description generated in Part A.
Role	You are an accident-event description assistant for dashcam scenes. Based on the visible nearby objects, describe one physically plausible accident that occurs immediately after the source frame.
Grounding	Select an involved object that is visible in the key frame and present in the near-field description. Do not introduce a new vehicle, pedestrian, obstacle, or road structure.
Required content	Explicitly state the accident location relative to the ego vehicle, such as front, front-left, or front-right, and identify the involved object. Describe the event directly rather than analyzing its cause.
Temporal constraint	The accident must be a continuous and plausible evolution from the source key frame. Do not introduce camera cuts, viewpoint changes, or discontinuous object motion.
Output format	Return exactly one plain sentence. Do not provide reasoning, warnings, explanations, JSON, Markdown, or multiple candidate events.
Example output	"The ego vehicle directly rear-ends the SUV ahead."

G.2 WAN VIDEO-GENERATION PROMPT

The final Wan prompt contains three ordered components: a fixed camera prompt, the VLM-generated key-frame description, and the VLM-generated accident description. No additional language-model rewriting is applied after concatenation. Table 14 shows the template and an example.

Table 14: Construction of the prompt supplied to Wan for counterfactual video generation.

Component	Content
Wan input	The source key frame and the concatenated text prompt described below.
Fixed system prompt	"Dashcam perspective with fixed camera height and pitch. The camera height must remain unchanged, and no abrupt camera cuts may occur. The video must remain in a first-person driving view, and the ego vehicle must never appear in the image."
Key-frame description	A one-sentence near-field scene description generated by the VLM using Part A of Table 13.
Accident description	A one-sentence accident description generated by the VLM using Part B of Table 13. The sentence specifies the accident location and involved object.
Concatenation rule	`<FIXED CAMERA PROMPT> + <KEY-FRAME DESCRIPTION> + <ACCIDENT DESCRIPTION>`
Complete example	"Dashcam perspective with fixed camera height and pitch. The camera height must remain unchanged, and no abrupt camera cuts may occur. The video must remain in a first-person driving view, and the ego vehicle must never appear in the image. An SUV is immediately ahead, and a sedan is traveling in the adjacent left lane. The ego vehicle directly rear-ends the SUV ahead."
Wan output	A temporally continuous counterfactual driving video that preserves the source viewpoint and scene layout while realizing the specified accident.

H ALGORITHMS OF RISE

H.1 INFERENCE

Here, $\mathrm{RiskRefine}$ denotes the iterative update in Eq. equation 45 using the frozen Risk Profile branch of the Latent Evaluator; it is not a separate module.

Algorithm 1 Adaptive inference with RISE

Require: Observation O_t, observed ego-pose history $\tau^{\mathrm{obs}}_{t-L:t}$, current ego kinematics $\mathbf{s}^{\mathrm{ego}}_t$, maximum horizon $H_{\mathcal{D}}$, computation preference λ
Ensure: Planned trajectory $\hat{\tau}$

$z_t \leftarrow \mathrm{Encoder}(O_t)$
$a^{\mathrm{obs}} \leftarrow \{\mathcal{A}(\tau^{\mathrm{obs}}_u, \tau^{\mathrm{obs}}_{u+1})\}_{u=t-L}^{t-1}$
$\hat{Z}_0 \leftarrow \emptyset$, $h \leftarrow 0$
while $h < H_{\mathcal{D}}$ **do**
  $(\mathbf{R}_h, \mathbf{B}_h) \leftarrow \mathrm{Eval}_{\theta_E}(z_t, \hat{Z}_h)$
  $e_h \leftarrow [\rho(z_t), \rho(\hat{Z}_h)]$
  $\xi_h \leftarrow [\overline{\mathbf{R}}_h, \overline{\mathbf{B}}_h, h/H_{\mathcal{D}}, c_h, \Delta c_{h+1}]$
  $x_h \leftarrow G_{\theta_G}(e_h, \xi_h, \lambda)$
  if $x_h \leq 0$ **then**
    break
  end if
  $\hat{z}_{t+h+1} \leftarrow \mathcal{P}(z_t, \hat{Z}_h, a^{\mathrm{obs}})$
  $\hat{Z}_{h+1} \leftarrow \hat{Z}_h \cup \{\hat{z}_{t+h+1}\}$
  $h \leftarrow h + 1$
end while
if $h \geq 0$ **then**
  $\widetilde{Z}_h \leftarrow \mathrm{RiskRefine}(z_t, \hat{Z}_h; \mathrm{Eval}_{\theta_E})$
else
  $\widetilde{Z}_h \leftarrow \emptyset$
end if
$(\{\hat{\tau}_s\}, \{g_s\}) \leftarrow \Pi_1(z_t, \widetilde{Z}_h, a^{\mathrm{obs}}, \mathbf{s}^{\mathrm{ego}}_t)$
$\hat{\tau} \leftarrow \hat{\tau}_{\arg\max_s g_s}$
return $\hat{\tau}$

H.2 TRAINING

Algorithm 2 Three-stage training procedure of RISE

Require: Real data $\mathcal{D}_{\mathrm{r}}$, accepted CounterDrive data $\mathcal{D}_{\mathrm{c}}$, verified pair set $\mathcal{I}_{\mathrm{pair}}$, maximum horizon $H_{\mathcal{D}}$, and computation preferences Λ

Ensure: Predictor $\mathcal{P}$, Planners Π_0, Π_1, Latent Evaluator Eval_{θ_E}, and Rollout Gate G_{θ_G}

Stage I: Predictor and initial Planner

Train $\mathcal{P}$ on $\mathcal{D}_{\mathrm{r}} \cup \mathcal{D}_{\mathrm{c}}$ using $\mathcal{L}_{\mathrm{Pre}}$
Freeze the Encoder and $\mathcal{P}$
for each sample in $\mathcal{D}_{\mathrm{r}}$ **do**
    Generate $\{\hat{Z}_h\}_{h=0}^{H_{\mathcal{D}}}$ and sample $h \sim \mathcal{U}\{0, \ldots, H_{\mathcal{D}}\}$
    Update $\Pi_0(z_t, \hat{Z}_h)$ using $\mathcal{L}^0_{\mathrm{Pla}}$
end for
Freeze Π_0

Stage II: Latent Evaluator and final Planner

for each real-data batch **do**
    Compute $r^*_{i,k} = \mathcal{E}_{\mathrm{risk}}(\Pi_0(z_{i,t}, \hat{Z}_{i,k}), \tau^*_i)$
    Accumulate $\mathcal{L}_{\mathrm{real}}$
end for
for each verified pair in $\mathcal{I}_{\mathrm{pair}}$ **do**
    Map the incident onset to k^{inc}_i and construct $m_{i,k}$
    Accumulate $\mathcal{L}_{\mathrm{rank}}$ and $\mathcal{L}_{\mathrm{loc}}$
end for
Update the Risk Profile branch using $\mathcal{L}_R$ and then freeze it
Initialize $\Pi_1 \leftarrow \Pi_0$
for each real sample **do**
    Sample h and compute the risk-refined prefix $\widetilde{Z}_h$
    Update $\Pi_1(z_t, \widetilde{Z}_h)$ using $\mathcal{L}^1_{\mathrm{Pla}}$
end for
Freeze Π_1
for each real sample **do**
    Enumerate $h = 0, \ldots, H_{\mathcal{D}}$ and compute $q_h = \mathcal{E}_{\mathrm{plan}}(\Pi_1(z_t, \widetilde{Z}_h), \tau^*)$
    for $h = 0, \ldots, H_{\mathcal{D}} - 1$ **do**
        Construct $\mathbf{B}^*_h = [q_{h+1} - q_h, \ldots, q_{H_{\mathcal{D}}} - q_h]$
    end for
    Update the Future Planning Gain branch using $\mathcal{L}_B$
end for

Stage III: Rollout Gate

Freeze $\mathcal{P}$, Π_1, and Eval_{θ_E}
for each real sample **do**
    Reuse the all-horizon planning scores $\{q_h\}$
    for all $\lambda \in \Lambda$ and $h = 0, \ldots, H_{\mathcal{D}} - 1$ **do**
        $g^*_h \leftarrow \max_{j>h}[(q_j - q_h) - \lambda(c_j - c_h)]$
        $y^*_h \leftarrow \mathbb{I}[g^*_h > 0]$
        Construct (e_h, ξ_h) from the current prefix and predict $x_h = G_{\theta_G}(e_h, \xi_h, \lambda)$
        Update G_{θ_G} using $\ell_{\mathrm{BCE}}(\sigma(x_h), y^*_h)$
    end for
end for
return $\mathcal{P}, \Pi_0, \Pi_1, \mathrm{Eval}_{\theta_E}, G_{\theta_G}$