SCENE-ADAPTIVE READOUT AND DECODING IN DIFFUSION DRIVING VLAS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion driving vision-language-action models repeatedly execute the backbone at every denoising stage, incurring substantial inference cost, while it remains unclear whether all scenes require the same decoding computation. To distinguish when a trajectory becomes readable from how much further refinement it requires, we analyze trajectory readout across network depth and decoding progress. We identify a depth-decoding dissociation in which late-layer representations already support useful trajectory readout after the first forward sweep, whereas early-layer representations benefit more from continued decoding. We further find that both the preferred readout layer and decoding depth vary across scenes. Based on these observations, we propose a scene-adaptive readout and decoding framework. A scene-conditioned readout MoE adaptively weights and combines trajectory predictions from layer-specific late-layer representations after a single backbone sweep, while an adaptive decoding policy applies additional native refinement only when required by the scene. We evaluate our framework on two diffusion driving VLAs across the Waymo E2E and nuScenes benchmarks. Our method improves validation RFS by up to 0.56 while also improving test performance, and achieves an end-to-end speedup of up to $6.0\times$ on Waymo E2E, while consistently reducing trajectory error on nuScenes. These results demonstrate that scene-level readout and decoding allocation improves planning quality while avoiding unnecessary decoding.

1 INTRODUCTION

Driving vision-language-action (VLA) models integrate scene understanding, driving decisions, and trajectory planning within unified frameworks (Renz et al., 2025; Yang et al., 2026; Wang et al., 2026). Recent diffusion-based driving VLAs (Ma et al., 2026; Li et al., 2026b; Xu et al., 2026b; Zhang et al., 2026) generate driving outputs through an iterative denoising process rather than sequential autoregressive decoding. Unlike autoregressive decoding, diffusion allows intermediate predictions to be repeatedly revised over multiple refinement steps. This iterative refinement allows high-level driving reasoning and low-level trajectory planning to mutually inform one another throughout inference, promoting consistency between driving decisions and predicted trajectories.

However, this iterative denoising (decoding) comes at a substantial computational cost (Wu et al., 2026; Christopher et al., 2025; Li et al., 2026a; Gao et al., 2025; Sun et al., 2026), as each denoising step requires repeated evaluations of the backbone network. These repeated evaluations introduce a critical inference-latency bottleneck, motivating methods that reduce decoding computation while preserving planning quality. A direct approach to reduce this cost is to terminate denoising before completing the full decoding schedule (Xin et al., 2020; Gao et al., 2023; Schuster et al., 2022; Raposo et al., 2024; Belrose et al., 2023). For a driving VLA, early termination requires extracting a reliable trajectory from intermediate representations before the trajectory tokens are fully decoded. This leads to two key choices during inference: (1) which layer to use for trajectory readout, and (2) at which decoding step to return the prediction.

Driving scenes vary in their visual complexity, interaction patterns, and planning ambiguity. Intuitively, this variation affects both where useful trajectory information emerges in the network and how quickly the prediction stabilizes across denoising steps. For example, straightforward lane following may admit a reliable trajectory from relatively early representations, whereas complex turns

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

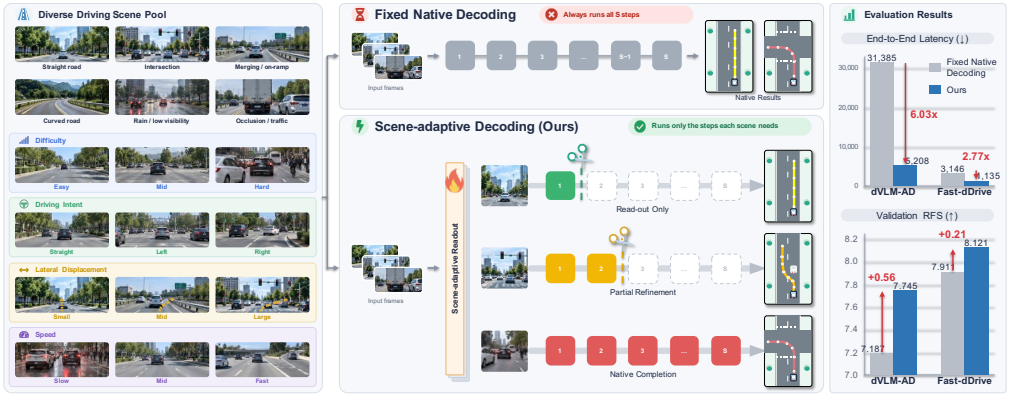


Figure 1: **Overview of scene-adaptive readout and decoding.**

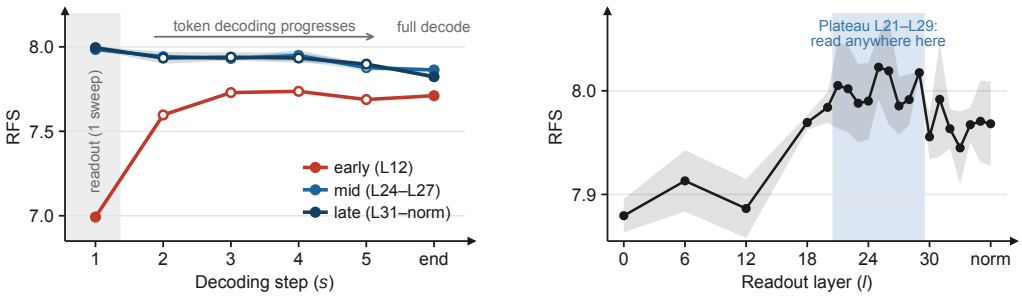
or interactions with other agents may require deeper semantic processing and additional refinement. As illustrated in Figure 1, different scenes can therefore require substantially different amounts of decoding to obtain reliable trajectories. A fixed readout layer and stopping step therefore risk over-computing simple scenes while under-computing difficult ones (Xin et al., 2020; Schuster et al., 2022; Raposo et al., 2024). This motivates a scene-adaptive strategy that selects an appropriate layer for trajectory readout and determines when further denoising is no longer beneficial.

To establish the empirical basis for this design, we use lightweight trajectory probes to examine readout quality across network layers and decoding stages in two architecturally distinct diffusion driving VLAs. We make two key observations. First, we identify a *depth-decoding dissociation*: late-layer representations at trajectory positions already support strong trajectory readout after the first forward sweep, whereas earlier-layer representations benefit substantially more from additional denoising. Second, beyond this aggregate trend, the optimal readout layer and decoding stage remain scene-dependent. Although a band of late layers consistently provides strong early readout, the best-performing layer within this band varies across scenes. Similarly, additional denoising yields only modest gains on average, but provides substantial improvements for particular scenes, whose optimal stopping points range from intermediate stages to native completion.

Based on these observations, we propose **scene-adaptive readout and decoding**, a framework that jointly adapts representation selection and decoding computation. After the first backbone sweep, a lightweight mixture-of-experts readout produces layer-specific trajectory predictions from a late-layer band and combines them using scene-dependent weights, enabling direct trajectory prediction before trajectory-token completion. An adaptive decoding policy then decides whether to terminate early, perform further refinement, or fall back to native completion. Together, these components adapt both which representations are used for trajectory prediction and how much decoding computation is allocated to each scene.

We evaluate our approach on two masked-diffusion driving VLAs (Ma et al., 2026; Zhang et al., 2026) and multiple autonomous-driving datasets (Xu et al., 2026a; Caesar et al., 2020). On the Waymo E2E benchmark, our method improves validation RFS by up to 0.56 and achieves up to a $6.0\times$ end-to-end inference speedup relative to the corresponding released native pipelines, while reducing trajectory errors at both the 3- and 5-second ADE on the test set. Notably, the adaptive policy invokes additional decoding only for a small subset of scenes, demonstrating that scene-dependent computation can preserve or improve planning quality while substantially reducing inference cost. Our contributions are summarized as follows:

- We systematically analyze trajectory readout across network depth and decoding progress, revealing a **depth-decoding dissociation** as well as scene-wise variation in the optimal readout layer and decoding stage.
- We introduce a **scene-adaptive readout MoE** that adaptively weights and combines trajectory predictions from layer-specific representations according to each scene.
- We propose **scene-adaptive decoding** that adaptively allocates the denoising budget across scenes, performing additional refinement when predicted to be beneficial.



(a) Readout quality across decoding stages.

(b) Readout quality across layers.

Figure 2: **Depth-decoding dissociation.** Late-layer representations support strong readout after first sweep and form a stable band, whereas early layers benefit more from continued decoding.

2 Related Work

Diffusion language models and driving VLAs. Diffusion language models replace autoregressive generation with masked denoising, enabling bidirectional refinement (Sahoo et al., 2024; Nie et al., 2026; Ye et al., 2025). Recent work studies reasoning and efficient decoding through confidence-based, parallel, speculative, and cached methods (Zhao et al., 2026; Hersche et al., 2026; Wu et al., 2026; Christopher et al., 2025; Li et al., 2026a; Gao et al., 2025; Sun et al., 2026). Recent work also finds that answers in diffusion language models can become identifiable before decoding is complete (Li et al., 2025). In autonomous driving, diffusion models have been extended to VLAs that jointly generate scene descriptions, decisions, and future trajectories (Ma et al., 2026; Zhang et al., 2026). Driving-specific methods reduce sampling cost through adaptive flow-matching steps (Trivedi et al., 2026) or KV-cache reuse, speculative drafting, and adaptive caching (Li et al., 2026c). While these methods focus on decoding efficiency, we study when trajectories become readable from intermediate representations and adapt readout depth and native decoding per scene.

Adaptive computation and representation readout. Adaptive computation methods use early exits, confidence-based halting, and conditional routing to allocate computation according to the input (Xin et al., 2020; Gao et al., 2023; Schuster et al., 2022; Raposo et al., 2024). Intermediate-state probes and tuned lenses further show that useful predictions can often be recovered before the final output is produced (Belrose et al., 2023), while mixture-of-experts models provide input-dependent expert selection (Zoph et al., 2022). We apply these ideas to diffusion driving VLAs by identifying a stable late-layer readout band and scene-wise variation in both the preferred readout layer and decoding depth. Our method therefore combines scene-adaptive trajectory readout with selective native decoding, rather than uniformly accelerating or shortening a fixed decoding procedure.

3 Trajectory Representation Analysis in Diffusion VLAs

We analyze where along network depth and when during iterative decoding trajectory information becomes available in diffusion VLAs, and whether a fixed readout layer and decoding depth are suitable for all scenes. We use Rater Feedback Score (RFS) (Xu et al., 2026a) as the primary metric. RFS is a preference-based planning metric in WOD-E2E that measures agreement between predicted trajectories and human-rater preferences, with higher values indicating better planning quality.

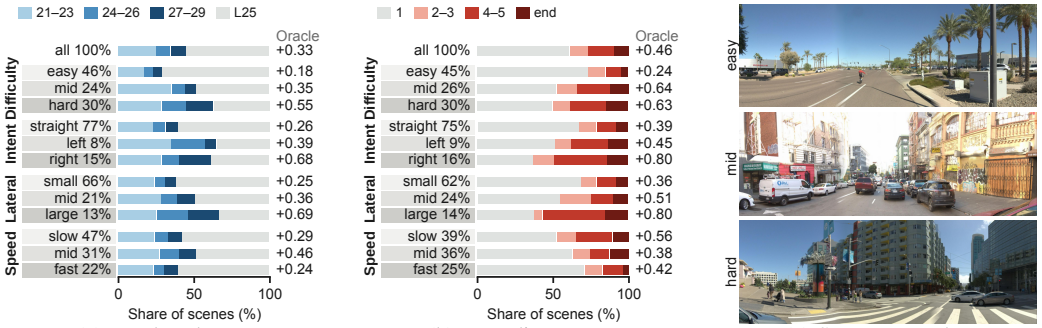
Let x denote an input scene, ℓ a network layer, and s a decoding stage. The first stage, $s = 1$, corresponds to the initial forward sweep before any iterative denoising step. For each pair (ℓ, s), we extract the trajectory-token states as:

$$z_\ell^{(s)}(x) = h_\ell^{(s)}(x)[\mathcal{I}_\tau], \tag{1}$$

where $\mathcal{I}_\tau$ indexes the token positions corresponding to the future waypoints.

We then apply a lightweight trainable trajectory probe $r_\ell^{(s)}$ to recover a continuous trajectory:

$$\hat{\tau}_\ell^{(s)}(x) = r_\ell^{(s)}\left(z_\ell^{(s)}(x)\right), \tag{2}$$



(a) Readout layer. (b) Decoding stage. (c) Scene examples.

Figure 3: **Scene-wise heterogeneity.** Preferred readout layers and decoding stages vary across groups. Percentages show scene shares, oracle gains appear on the right, and (c) shows representative scenes for each difficulty level.

where $\hat{\tau}_\ell^{(s)}(x)$ denotes the trajectory read from layer ℓ at decoding stage s. Each probe is trained with trajectory supervision while the diffusion backbone remains frozen, allowing us to measure the trajectory information available at different depths and decoding stages.

3.1 Pre-decoding Readout and a Stable Readout Band

We first examine whether trajectory information can be recovered before response tokens are fully decoded. Figure 2a compares readout quality across decoding stages and network depths. Intermediate-to-late representations (L24–L27 and L31–norm) already support strong trajectory prediction after the first sweep, with little benefit from further decoding. In contrast, the early-layer representation (L12) improves more substantially as decoding proceeds, yet remains weaker overall.

This depth-dependent behavior reveals a *depth-decoding dissociation*. Early-layer readout benefits more from continued decoding, whereas the marginal gain from additional decoding is limited in later layers. Thus, the benefit of iterative decoding for trajectory readout varies across network depth. In intermediate-to-late layers, useful trajectory predictions can already be recovered through lightweight readout before token completion.

We next fix the decoding stage at the first forward sweep ($s = 1$) and vary the readout layer. As shown in Figure 2b, readout quality forms a broad plateau rather than peaking at a single layer. Performance remains stable across a broad range of intermediate-to-late layers, approximately L21–L29. We therefore use this range as the operational readout band:

$$\mathcal{B} = \{21, \ldots, 29\}. \tag{3}$$

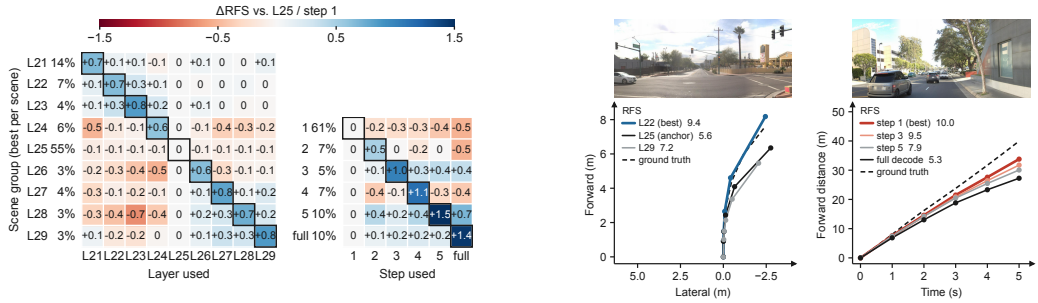
This band provides a compact set of layer representations for the scene-adaptive readout MoE.

3.2 Scene-wise Heterogeneity in Readout and Decoding

The existence of a stable late-layer readout band does not imply that the same layer is optimal for every scene. For each scene, we evaluate L21–L29 and define the layer with the highest RFS as the *oracle layer*. The oracle readout gain is the RFS difference between this scene-specific oracle and the native diffusion VLA prediction. Since the oracle choice is unavailable at inference time, it provides an upper bound on the benefit of scene-adaptive layer selection. To examine layer preference across scenes, we stratify by difficulty, driving intent, lateral displacement, and speed. These attributes are used only for analysis and are not provided to the policy. Representative difficulty examples are shown in Figure 3c, with additional examples in Appendix.

Figure 3a shows that approximately 55% of scenes select L25 as the oracle layer, the most frequent choice. However, the remaining 45% prefer other layers within the late-layer band, indicating scene-wise variation in the optimal readout depth. For these scenes, selecting a suitable layer can yield additional RFS gains over the native diffusion VLA. Thus, scene-adaptive readout is intended not to change the layer for every scene, but to identify when L25, the common oracle layer, is suboptimal.

A similar scene-wise variation appears in decoding. In Figure 3b, approximately 61% of scenes achieve their highest RFS after the first forward sweep ($s = 1$). For most scenes, it is therefore



(a) Per-scene gain. (b) Scene examples.

Figure 4: **Scene-adaptive gains and examples.** Heatmaps quantify scene-wise gains from adaptive readout and decoding, with representative trajectory comparisons.

preferable to read out the representation after the first sweep rather than denoise the trajectory tokens to completion. This observation is consistent with Figure 2a, where intermediate-to-late representations show only limited gains from further decoding on average. Additional decoding does not improve planning quality uniformly and can sometimes reduce the quality of an already strong readout prediction. In such cases, the trajectory recovered at later stages deviates from the first-sweep readout and yields lower RFS, as illustrated in Figure 4. Conversely, some scenes improve at intermediate or later stages, indicating that the benefit of additional decoding is scene-dependent.

Figure 4a quantifies the RFS gain from selecting scene-specific readout layers and decoding stages relative to the fixed anchor and first-sweep references. The examples in Figure 4 show that a readout from a layer other than L25 can better match the ground-truth trajectory in some scenes, whereas the first-stage trajectory can be more accurate than a trajectory produced after prolonged decoding in others. These examples show that the benefit of scene-adaptive inference comes not from uniformly increasing computation, but from selecting a representation and stopping point suited to each scene.

Overall, L25 and the first forward sweep provide global defaults, but neither is optimal for all scenes. Scene-adaptive inference is not a binary choice between readout and native decoding. It jointly determines *where* to read the trajectory representation and *how far* to continue decoding for each scene. In the next section, we implement this principle with a scene-adaptive readout MoE over the late-layer band and a scene-adaptive decoding policy that allocates computation only when needed.

4 Methodology

Based on the trajectory formation analysis in Section 3, we propose a scene-adaptive framework that jointly selects trajectory representations and decoding depth. Given an input scene x, the frozen diffusion VLA performs a single forward sweep, after which layer-specific experts convert representations from the readout band $\ell \in \mathcal{B}$ into candidate trajectories, and a scene-conditioned MoE combines them (Zoph et al., 2022; Fedus et al., 2022; Shazeer et al., 2017). The resulting readout draft is either returned directly or refined through adaptive fallback decoding when needed. The proposed framework, as illustrated in Figure 5, adapts inference along two axes: how trajectory predictions from layer-specific representations are combined, and how far iterative decoding should proceed. Instead of applying the same readout configuration and decoding schedule to every scene, it allocates inference computation according to each scene.

4.1 Early Trajectory Readout

For scene x, let $h_\ell^{(s)}(x)$ denote the hidden state at layer ℓ and decoding stage s. Following the analysis in Section 3, we use the trajectory-token representation from the first forward sweep:

$$z_\ell^{(1)}(x) = h_\ell^{(1)}(x)[\mathcal{I}_\tau], \tag{4}$$

where $\mathcal{I}_\tau$ indexes the token positions reserved for trajectory prediction, with one position corresponding to each future waypoint.

At the same time, we obtain a one-pass draft $a(x)$ by selecting, at each trajectory-token position, the vocabulary token with the largest logit during the first forward sweep. The layer-specific readout

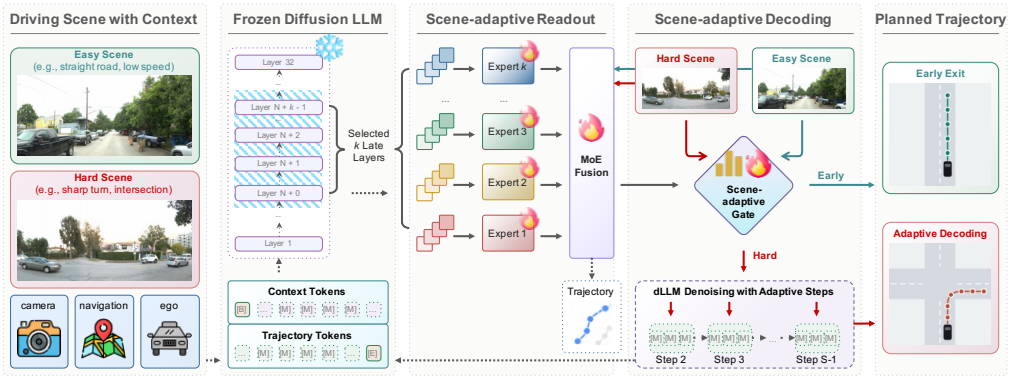


Figure 5: **Scene-adaptive readout and decoding pipeline.**

head, $\mathcal{R}_{\theta_\ell}$, takes the trajectory-token representation and this draft as input to produce a trajectory prediction:

$$\hat{\tau}_\ell(x) = \mathcal{R}_{\theta_\ell}\left(z_\ell^{(1)}(x), a(x)\right). \tag{5}$$

Each readout head is trained with GT trajectory supervision while the backbone remains frozen. The readout path maps the first-sweep representation to continuous waypoints without generating additional reasoning text or trajectory tokens, thereby skipping the remaining native decoding stages.

4.2 Scene-Adaptive Readout

For each layer in the readout band identified in Section 3, $\mathcal{B} = \{21, \ldots, 29\}$, we construct a trajectory expert with a readout head and a latent editor. Since all experts operate on representations extracted from the same first forward sweep, the expert bank requires no additional backbone forward passes. For layer $\ell \in \mathcal{B}$, the editor is initialized with the corresponding first-sweep representation:

$$z_\ell^{(r)}(x) = z_\ell^{(1)}(x), \tag{6}$$

where $z_\ell^{(r)}(x)$ denotes the representation after the r-th latent refinement step. The editor applies residual updates to provide a potential correction to the representation:

$$z_\ell^{(r+1)}(x) = z_\ell^{(r)}(x) + \Delta_{\phi_\ell}^{(r)}\left(z_\ell^{(r)}(x)\right), \qquad r = 0, \ldots, R-1, \tag{7}$$

where R is the number of latent refinement steps, and $\Delta_{\phi_\ell}^{(r)}$ denotes the residual update applied at step r for layer ℓ, parameterized by learnable parameters ϕ_ℓ. These refinements are intended to compensate for geometric inconsistencies that may arise from direct readout, such as lateral offsets or local curvature. The editor does not directly modify the native decoder output. Instead, its refined representation is converted into a trajectory by the corresponding layer-specific readout head.

After R latent refinement steps, the layer-specific readout head uses the refined representation and the one-pass draft from the first forward sweep to produce a candidate trajectory,

$$\hat{\tau}_\ell(x) = \mathcal{R}_{\theta_\ell}\left(z_\ell^{(R)}(x), a(x)\right). \tag{8}$$

The latent editor refines only the hidden representation, while $a(x)$ remains fixed. Each layer-specific expert therefore provides a trajectory candidate based on a different layer representation, while all candidates are computed from a single backbone forward sweep.

The candidate trajectories are combined by a scene-conditioned gate. We construct its input by mean-pooling the trajectory-slot and vision-token hidden states, and concatenating these statistics with the ego history, navigation command, one-pass draft, layer-specific candidate trajectories, and their deviations from the candidate mean. The resulting scene representation is denoted by $u(x)$. The gate produces layer-wise mixture weights:

$$w(x) = \mathrm{softmax}\left(g_\psi(u(x))\right), \quad \hat{\tau}_{\mathrm{moe}}(x) = \sum_{\ell \in \mathcal{B}} w_\ell(x)\hat{\tau}_\ell(x), \tag{9}$$

g_ψ outputs one score for each layer in $\mathcal{B}$. The softmax converts scores into normalized mixture weights $w_\ell(x)$, which sum to one over the candidate layers. Details are provided in the Appendix.

The gate is trained with trajectory supervision to optimize planning quality of the fused prediction:

$$\min_{\psi} \mathcal{L}_{\text{traj}} \left(\hat{\tau}_{\text{moe}}(x), \tau^{\text{GT}}(x) \right), \tag{10}$$

where $\tau^{\text{GT}}(x)$ denotes the ground-truth waypoint trajectory. Thus, the gate can emphasize a single layer or combine multiple layer-specific predictions according to the scene representation.

To reduce the variance of the MoE prediction, we retain an auxiliary readout branch based on a fixed reference layer $\ell^\star$. The reference layer is selected during training, and we use $\ell^\star = \text{L27}$ in our experiments. This branch applies a sparse mixture over the trajectory-token representation at $\ell^\star$, followed by a latent editor, to produce an independent trajectory prediction. It is not used to route experts across scenes, but instead serves as a reference branch that stabilizes the MoE output.

The final readout draft is obtained by combining the MoE and auxiliary predictions:

$$\hat{\tau}_{\text{read}}(x) = (1-\beta)\hat{\tau}_{\text{moe}}(x) + \beta\hat{\tau}_{\text{aux}}(x). \tag{11}$$

This trajectory serves as the input to scene-adaptive decoding and the output of the early-exit path.

4.3 Scene-Adaptive Decoding

The reliability of the readout draft varies across scenes, making it undesirable to apply native decoding uniformly. Our method therefore uses readout as the default path and invokes native decoding as a fallback only for scenes whose learned decoding gate predicts a benefit from additional refinement. Scenes for which the readout is predicted to be sufficient terminate immediately, while native computation is reserved for scenes predicted to benefit from further refinement.

Specifically, the decoding gate forms a scene representation $v(x)$ by mean-pooling the pre-decoding trajectory-slot and vision-token states and concatenating these features with the readout confidence, expert disagreement, and navigation context. The complete feature construction is provided in the Appendix. The gate then produces a score representing the benefit of additional decoding,

$$q_{\text{dec}}(x) = g_\omega(v(x)), \tag{12}$$

where g_ω is a trainable scene-conditioned gate. It is trained on the training split using a binary target indicating whether native refinement reduces the ground-truth trajectory error relative to the readout draft. At inference time, the gate uses only $v(x)$ and does not access ground-truth trajectories or evaluator outcomes. If the gate score exceeds a threshold λ, the scene is sent to the fallback decoding path; otherwise, the readout trajectory is returned directly,

$$G(x) = \mathbb{I}\left[q_{\text{dec}}(x) \geq \lambda\right]. \tag{13}$$

Fallback decoding evaluates the native decoder incrementally rather than executing the entire schedule at once. Let S denote the total number of decoding stages, and let $\ell^\star$ denote the reference layer used for stage-wise trajectory readout. At each intermediate stage $s = 1, \ldots, S-1$, we extract the trajectory-slot representation $h^{(s)}_{\ell^\star}(x)[\mathcal{I}_\tau]$ and apply a lightweight stage-specific readout probe,

$$P_s(x) = \mathcal{R}^{(s)}\left(h^{(s)}_{\ell^\star}(x)[\mathcal{I}_\tau]\right), \qquad s = 1, \ldots, S-1. \tag{14}$$

Each probe $\mathcal{R}^{(s)}$ is trained with trajectory supervision using only the hidden state from its decoding stage. The readout draft is withheld from the probe input, so $P_s(x)$ estimates the trajectory recoverable from the native decoder representation at stage s rather than copying the initial readout prediction.

During fallback decoding, each intermediate trajectory is compared with the readout draft at the prediction horizon. Let $\hat{\tau}_{\text{read}}(x)$ denote the trajectory produced by the scene-adaptive readout, and let $P_s(x)_{T_\tau}$ denote the final future waypoint predicted by the stage-s probe. We select the earliest stage at which the two predictions agree within a threshold ε,

$$s^\star(x) = \min\left\{s < S : \|P_s(x)_{T_\tau} - \hat{\tau}_{\text{read}}(x)_{T_\tau}\|_2 < \varepsilon\right\}, \tag{15}$$

Table 1: **Planning performance and inference efficiency on WOD-E2E.** † dVLM-AD results are reproduced using the released checkpoint under our evaluation setting, as the original reported setting is not fully recoverable from the released configuration.

Model	Backbone	Validation			Test			Latency
		RFS ↑	ADE@3s ↓	ADE@5s ↓	RFS ↑	ADE@3s ↓	ADE@5s ↓	ms/scene ↓
Autoregressive VLA baselines								
OpenEMMA (Xing et al., 2025)	AR	6.698	2.725	4.957	5.158	6.684	12.476	22,291
LightEMMA (Qiao et al., 2025)	AR	6.670	2.595	4.951	6.517	1.705	3.740	16,414
AutoVLA (Zhou et al., 2025)	AR	–	–	–	7.557	1.351	2.958	7,855
Poutine (Rowe et al., 2025)	AR	8.120	–	–	7.909	1.272	2.941	–
IRL-VLA (Jiang et al., 2025)	AR	–	–	–	7.837	1.193	2.821	–
Qwen-Drive-1.0 (Zhou et al., 2026)	AR	7.950	0.990	2.330	7.760	1.200	2.660	1,679
DrivePI-vision (Liu et al., 2026)	AR	–	–	–	7.520	1.250	2.910	–
MindVLA-U1 (Huang et al., 2026)	AR	7.920	0.890	2.110	7.770	1.160	2.670	–
Diffusion VLA baselines								
dVLM-AD† (Ma et al., 2026)	Diffusion	7.187	1.119	3.024	7.486	1.337	3.241	31,385
+Ours	Diffusion	**7.745** +0.558	**1.118** −0.001	**2.615** −0.409	**7.557** +0.071	**1.294** −0.043	**3.130** −0.111	**5.208** 6.03×
Fast-dDrive (Zhang et al., 2026)	Block Diff.	7.911	**0.937**	**2.173**	7.823	1.254	2.907	3,146
+Ours	Block Diff.	**8.121** +0.210	0.957 +0.020	2.186 +0.013	**7.881** +0.058	**1.196** −0.058	**2.681** −0.226	**1.135** 2.77×

where ε defines the agreement criterion. Agreement at the prediction horizon indicates that the trajectory recovered at the current stage is consistent with the initial readout, so further decoding is not performed. The final trajectory is obtained by averaging the readout and stage-$s^\star$ predictions:

$$\hat{\tau}(x) = \frac{1}{2}\left(\hat{\tau}_{\mathrm{read}}(x) + P_{s^\star}(x)\right). \tag{16}$$

If no intermediate stage agrees with the readout, decoding continues to the native endpoint. Since no probe is used at this endpoint, its native trajectory is combined directly with the readout:

$$\hat{\tau}(x) = \frac{1}{2}\left(\hat{\tau}_{\mathrm{read}}(x) + \hat{\tau}_{\mathrm{native}}(x)\right). \tag{17}$$

Consequently, scenes follow three routes: (1) early termination after readout, (2) termination after a subset of fallback stages, or (3) completion of native decoding. The learned entry gate determines which scenes enter fallback, while the agreement test determines the earliest stopping stage. Together, they allocate a scene-dependent decoding budget without applying the schedule uniformly.

5 Experiment

5.1 Experimental Setup

Datasets and Metrics. We evaluate on the Waymo End-to-End (WOD-E2E) and nuScenes (Xu et al., 2026a; Caesar et al., 2020). We report Average Displacement Error at the 3 s and 5 s horizons, along with the Rater Feedback Score (RFS), a human-aligned score measuring agreement with human-rated trajectories. On nuScenes, we report L2 displacement error at the 1 s, 2 s, and 3 s horizons. Latency is measured end-to-end in $ms/scene$ on a single NVIDIA A6000 with batch size one.

Implementation Details. The diffusion backbones remain frozen, while the readout heads, latent editors, gates, and stage-wise probes are trained on the corresponding training splits using ground-truth waypoints. Readout heads use the first-sweep argmax draft, latent editors run for $R = 12$ refinement rounds, and fallback decoding uses the agreement threshold ε. Further architectural and training details are provided in the Appendix.

5.2 Main Results

We first evaluate the proposed scene-adaptive inference policy on the Waymo E2E benchmark. Table 1 reports planning quality and end-to-end latency for two diffusion driving VLAs.

On dVLM-AD, our method improves RFS from 7.187 to 7.745 ($+0.558$) on the validation set and from 7.486 to 7.557 ($+0.071$) on the test set. On the official test split, it also reduces ADE from

Table 2: **Trajectory error on the nuScenes validation set.**

Method	1s	2s	3s	Avg.
UniAD (Hu et al., 2023)	0.20	0.42	0.75	0.46
VAD-Base (Jiang et al., 2023)	0.17	0.34	0.60	0.37
BEV-Planner (Li et al., 2024)	0.16	0.32	0.57	0.35
OpenEMMA (Xing et al., 2025)	1.45	3.21	3.76	2.81
DriveVLM (Tian et al., 2024)	0.18	0.34	0.68	0.40
AutoVLA (Zhou et al., 2025)	0.25	0.46	0.73	0.48
Diffusion VLAs				
dVLM-AD (Ma et al., 2026)	0.15	0.40	0.68	0.41
+ Ours	**0.14**	**0.38**	**0.62**	**0.39**
Fast-dDrive (Zhang et al., 2026)	0.12	0.33	0.50	0.32
+ Ours	**0.11**	**0.30**	**0.45**	**0.29**

Table 3: **Ablation of scene-adaptive readout and decoding on Waymo validation.**

Backbone	Variant	RFS	ADE@3s	ADE@5s	Latency
dVLM-AD	Base	7.187	1.119	3.024	31,385
	+ Readout	7.594	1.291	3.056	2,273
	+ Scene-Adaptive MoE	7.632	1.223	2.896	2,490
	+ Scene-Adaptive Decoding	**7.745**	**1.118**	**2.615**	5,208
Fast-dDrive	Base	7.911	**0.937**	**2.173**	3,146
	+ Readout	8.025	0.969	2.209	946
	+ Scene-Adaptive MoE	8.116	0.960	2.195	1,058
	+ Scene-Adaptive Decoding	**8.121**	0.957	2.186	1,135

1.337 to 1.294 at 3 seconds and from 3.241 to 3.130 at 5 seconds. Latency decreases from $31{,}385$ to $5{,}208$ ms per scene, corresponding to a $6.0\times$ speedup over the native pipeline.

On Fast-dDrive, RFS improves from 7.911 to 8.121 ($+0.210$) on validation and from 7.823 to 7.881 ($+0.058$) on the test set. Test ADE decreases from 1.254 to 1.196 at 3 seconds and from 2.907 to 2.681 at 5 seconds. Latency is reduced from $3{,}146$ to $1{,}135$ ms per scene, yielding a $2.8\times$ speedup.

We further evaluate the method on the nuScenes validation set in Table 2. Although the method is developed and analyzed primarily on Waymo, it also improves the average L2 error for both diffusion backbones. The average error decreases from 0.41 to 0.39 for dVLM-AD and from 0.32 to 0.29 for Fast-dDrive. These results suggest that the benefit of early readout and scene-adaptive computation is not limited to a single driving dataset.

The consistent improvements across both backbones suggest that the proposed benefit is not specific to a single diffusion architecture. Early readout avoids additional native decoding for scenes where a useful trajectory can already be recovered from the first-sweep representation, while adaptive decoding preserves further refinement for scenes that benefit from it. Scene-level allocation of these two computation paths therefore improves planning quality and inference efficiency simultaneously.

5.3 Ablation Study

Table 3 shows how each component affects driving performance and latency. Early readout alone improves RFS from 7.187 to 7.594 for dVLM-AD and from 7.911 to 8.025 for Fast-dDrive. This is consistent with our analysis in Section 3, which shows that the first forward sweep yields the highest readout RFS for most analyzed scenes. At the same time, omitting the remaining native decoding stages substantially reduces latency. Adding the scene-adaptive MoE further improves driving performance with only modest latency overhead. Scene-adaptive decoding provides additional gains, although its latency overhead depends on the backbone. In particular, for dVLM-AD, latency increases from $2{,}490$ to $5{,}208$ ms/scene when adaptive decoding is enabled, while remaining substantially lower than the $31{,}385$ ms/scene of native decoding. For Fast-dDrive, the corresponding increase is much smaller, from $1{,}058$ to $1{,}135$ ms/scene. Notably, the readout with scene-adaptive MoE already achieves 7.632 and 8.116 RFS for dVLM-AD and Fast-dDrive, respectively, providing an efficient operating point without additional native decoding.

6 Conclusion

We studied whether iterative denoising is uniformly necessary for trajectory prediction in diffusion driving VLAs. By analyzing trajectory readout across network depth and decoding progress, we identified a depth-decoding dissociation. Late-layer representations often support useful trajectory readout after the first forward sweep, while the benefit of further decoding varies across scenes. These findings motivate scene-conditioned readout and selective decoding as a way to exploit variation across scenes.

Based on these observations, we introduce a scene-adaptive inference framework combining a scene-conditioned readout MoE with adaptive fallback decoding, enabling early termination when possible and additional refinement when needed. On Waymo E2E, our method improves RFS by up to 0.56 and reduces end-to-end latency by up to $6.0\times$, while nuScenes results further support the generality of the approach. Overall, scene-level allocation of readout and decoding computation improves planning quality and inference efficiency.

AI USE STATEMENT

In this work, we did not use generative AI tools to develop theoretical models or conceptual frameworks, formulate mathematical claims, design research methodology or experiments, or interpret experimental results. Other required-disclosure tasks are not applicable to this work.

Additionally, we used generative AI tools for brainstorming, assisting with software code, and editing the manuscript to improve readability and language. All AI-assisted code was reviewed and tested by the authors, and all AI-assisted edits to the manuscript were manually reviewed to ensure technical accuracy and consistency with the authors' intended meaning. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or the collection of private data. We use publicly available datasets in accordance with their respective licenses. We are not aware of any significant ethical concerns specific to this work.

REPRODUCIBILITY STATEMENT

We provide a detailed description of the proposed method in Section 4 and the experimental setup, datasets, evaluation metrics, and main implementation details in Section 5. Additional training, hyperparameter, and implementation details required for reproducing our experiments are provided in the Appendix.

REFERENCES

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

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 *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11618–11628. IEEE, 2020.

Jacob K Christopher, Brian R Bartoldson, Tal Ben-Nun, Michael Cardei, Bhavya Kailkhura, and Ferdinando Fioretto. Speculative diffusion decoding: Accelerating language generation through diffusion. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 12042–12059, 2025.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Xiangxiang Gao, Wei Zhu, Jiasheng Gao, and Congrui Yin. F-pabee: Flexible-patience-based early exiting for single-label and multi-label text classification tasks. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

Yifeng Gao, Ziang Ji, Yuxuan Wang, Biqing Qi, Hanlin Xu, and Linfeng Zhang. Self speculative decoding for diffusion large language models. *arXiv preprint arXiv:2510.04147*, 2025.

Michael Hersche, Samuel Moor-Smith, Thomas Hofmann, and Abbas Rahimi. Soft-masked diffusion language models. In *International Conference on Learning Representations*, volume 2026, pp. 105685–105716, 2026.

Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17853–17862. IEEE, 2023.

Yuzhou Huang, Benjin Zhu, Hengtong Lu, Victor Shea-Jay Huang, Haiming Zhang, Wei Chen, Jifeng Dai, Yan Xie, and Hongsheng Li. Mindvla-u1: Vla beats va with unified streaming architecture for autonomous driving. *arXiv preprint arXiv:2605.12624*, 2026.

Anqing Jiang, Yu Gao, Yiru Wang, Zhigang Sun, Shuo Wang, Yuwen Heng, Hao Sun, Shichen Tang, Lijuan Zhu, Jinhao Chai, et al. Irl-vla: Training an vision-language-action policy via reward world model. *arXiv preprint arXiv:2508.06571*, 2025.

Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8306–8316. IEEE, 2023.

Guanghao Li, Zhihui Fu, Min Fang, Qibin Zhao, Ming Tang, Chun Yuan, and Jun Wang. Diffuspec: Unlocking diffusion language models for speculative decoding. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 20896–20910, 2026a.

Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Soroush Vosoughi, and Shiwei Liu. Diffusion language models know the answer before decoding. *arXiv preprint arXiv:2508.19982*, 2025.

Pengxiang Li, Yinan Zheng, Yue Wang, Huimin Wang, Hang Zhao, Jingjing Liu, Xianyuan Zhan, Kun Zhan, and Xianpeng Lang. Discrete diffusion for reflective vision-language-action models in autonomous driving. In *International Conference on Learning Representations*, volume 2026, pp. 14647–14666, 2026b.

Zekai Li, Yihao Liang, Hongfei Zhang, Jian Chen, Yesheng Liang, and Zhijian Liu. Flashdrive: Flash vision-language-action inference for autonomous driving. *arXiv preprint arXiv:2608.12932*, 2026c.

Zhiqi Li, Zhiding Yu, Shiyi Lan, Jiahan Li, Jan Kautz, Tong Lu, and Jose M Alvarez. Is ego status all you need for open-loop end-to-end autonomous driving? In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14864–14873. IEEE, 2024.

Zhe Liu, Runhui Huang, Rui Yang, Siming Yan, Zining Wang, Lu Hou, Di Lin, Xiang Bai, and Hengshuang Zhao. Drivepi: Spatial-aware 4d mllm for unified autonomous driving understanding, perception, prediction and planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3688–3698, 2026.

Yingzi Ma, Yulong Cao, Wenhao Ding, Shuibai Zhang, Yan Wang, Boris Ivanovic, Ming Jiang, Marco Pavone, and Chaowei Xiao. dvlm-ad: Enhance diffusion vision-language-model for driving via controllable reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1050–1061, 2026.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *Advances in Neural Information Processing Systems*, 38:50608–50646, 2026.

Zhijie Qiao, Haowei Li, Zhong Cao, and Henry X Liu. Lightemma: Lightweight end-to-end multimodal model for autonomous driving. *arXiv preprint arXiv:2505.00284*, 2025.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Katrin Renz, Long Chen, Elahe Arani, and Oleg Sinavski. Simlingo: Vision-only closed-loop autonomous driving with language-action alignment. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11993–12003. IEEE, 2025.

Luke Rowe, Rodrigue de Schaetzen, Roger Girgis, Christopher Pal, and Liam Paull. Poutine: Vision-language-trajectory pre-training and reinforcement learning post-training enable robust end-to-end autonomous driving. *arXiv preprint arXiv:2506.11234*, 2025.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Shengyin Sun, Yiming Li, Renxi Liu, Xinqi Li, Hui-Ling Zhen, Weizhe Lin, Chen Chen, Xianzhi Yu, Mingxuan Yuan, and Chen Ma. Psd: Pushing the pareto frontier of diffusion llms via parallel speculative decoding. *arXiv preprint arXiv:2605.15609*, 2026.

Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models. *arXiv preprint arXiv:2402.12289*, 2024.

Ananya Trivedi, Anjian Li, Mohamed Elnoor, Yusuf Umut Ciftci, Avinash Singh, Jovin D'sa, Sangjae Bae, David Isele, Taşkın Padır, and Faizan M Tariq. Adaptive time step flow matching for autonomous driving motion planning. In *2026 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1503–1510. IEEE, 2026.

Xinyang Wang, Qian Liu, Wenjie Ding, Zhao Yang, Wei Li, Chang Liu, Bailin Li, Kun Zhan, Xianpeng Lang, and Wei Chen. Unifying language-action understanding and generation for autonomous driving. *arXiv preprint arXiv:2603.01441*, 2026.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. In *International Conference on Learning Representations*, volume 2026, pp. 57027–57051, 2026.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 2246–2251, 2020.

Shuo Xing, Chengyuan Qian, Yuping Wang, Hongyuan Hua, Kexin Tian, Yang Zhou, and Zhengzhong Tu. Openemma: Open-source multimodal model for end-to-end autonomous driving. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, pp. 911–919. IEEE, 2025.

Runsheng Xu, Hubert Lin, Wonseok Jeon, Hao Feng, Yuliang Zou, Liting Sun, John Gorman, Kate Tolstaya, Sarah Tang, Brandyn White, et al. Wod-e2e: Waymo open dataset for end-to-end driving in challenging long-tail scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3709–3718, 2026a.

Yifang Xu, Jiahao Cui, Zhihao Zhu, Hanlin Shang, Shan Luan, Mingwang Xu, Feipeng Cai, Neng Zhang, Yaoyi Li, Jia Cai, et al. Wam-flow: Parallel coarse-to-fine motion planning via discrete flow matching for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24918–24928, 2026b.

Zhenjie Yang, Yilin Chai, Xiaosong Jia, Qifeng Li, Yuqian Shao, Xuekai Zhu, Haisheng Su, and Junchi Yan. Drivemoe: Mixture-of-experts for vision-language-action model in end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10678–10688, 2026.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025. *URL https://arxiv. org/abs/2508.15487*, 2025.

Kewei Zhang, Jin Wang, Sensen Gao, Chengyue Wu, Yulong Cao, Songyang Han, Boris Ivanovic, Langechuan Liu, Marco Pavone, Song Han, et al. Fast-ddrive: Efficient block-diffusion vlm for autonomous driving. *arXiv preprint arXiv:2605.23163*, 2026.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *Advances in Neural Information Processing Systems*, 38:56729–56762, 2026.

Xin Zhou, Zongchuang Zhao, Zhibo Yang, Mingsheng Li, Humen Zhong, Shuai Bai, Du Chu, Ruizhe Chen, Zhaohai Li, Jun Tang, et al. Qwen-drive-1.0: An initial step towards a vision-language foundation model for autonomous driving. *arXiv preprint arXiv:2609.00111*, 2026.

Zewei Zhou, Tianhui Cai, Seth Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. Autovla: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. *Advances in Neural Information Processing Systems*, 38: 27920–27956, 2025.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

APPENDIX

CONTENTS

A Implementation Details **13**

A.1 Hardware and software. 13

A.2 Training protocol and split usage. 14

A.3 Hyperparameters 14

B Performance–Latency Trade-off **14**

C Scene Stratification and Category Examples **15**

D Scene-Adaptive Readout and Decoding Details **15**

D.1 Scene-Adaptive Readout Gate 15

D.2 Scene-Adaptive Decoding Gate 16

E Scene-Adaptive Routing and Decoding Statistics **17**

F Limitations **18**

A IMPLEMENTATION DETAILS

A.1 HARDWARE AND SOFTWARE.

All training and all latency measurements use one NVIDIA RTX A6000 (48 GB, driver 535.274) on a host with an AMD EPYC 7413 CPU (24 cores, 96 threads) and 503 GB RAM, running Ubuntu 22.04 (kernel 6.8). Fast-dDrive experiments use Python 3.9, PyTorch 2.8.0 with CUDA 12.8 and cuDNN 9.1, and Transformers 4.57; dVLM-AD experiments use Python 3.10, PyTorch 2.4.0 with CUDA 12.1 and cuDNN 9.1, and Transformers 4.43, as required by the released code.

A.2 TRAINING PROTOCOL AND SPLIT USAGE.

Our framework is trained entirely on the training split. The readout heads, latent editors, scene-adaptive fusion module, decoding gate, and stage-wise probes are all optimized using training examples only. The validation and test sets are never used for learning model parameters.

The validation set is used only for the diagnostic analysis in Sec. 3 to characterize where trajectory information becomes readable in the frozen backbones. The validation analysis only determines the global candidate readout band; all adaptive components and the auxiliary reference layer are learned or selected using the training split. In particular, the auxiliary branch uses $\ell^\star = \mathrm{L27}$, selected during training. Per-scene oracle layer and decoding-stage assignments from the validation analysis are used solely for diagnosis and never as supervision for the routing policies.

The test split remains fully held out throughout model development and is used only for final evaluation.

A.3 HYPERPARAMETERS

Table A1 summarizes the implementation and training hyperparameters used for the proposed modules. The diffusion backbones are kept frozen throughout training, and only the readout heads, latent editors, scene-adaptive readout gate, and decoding components are optimized. Unless otherwise specified, architectural choices are shared across the two backbones, while batch sizes and gate optimization settings are adjusted to account for differences in model size and memory usage. We use $R = 12$ latent refinement rounds for the readout experts and an agreement radius of $\varepsilon = 1.5\,\mathrm{m}$ for adaptive fallback decoding.

Table A1: Implementation hyperparameters. Shared values apply to both backbones.

Configuration	**Fast-dDrive**	**dVLM-AD**
Backbone and features		
Language model / hidden size d	Qwen2.5-VL-3B / 2,048	LLaDA-V-8B / 4,096
Native decoding	Block diffusion; $S = 6$ stages	Semi-autoregressive; block size 512
Trajectory slots per scene	≤ 88	100
Readout heads		
Training epochs / batch size	60 / 256	60 / 64
Width / blocks / attention heads	256 / 4 / 4	
Training target	Ground-truth trajectory	
Optimizer	AdamW; learning rate 3×10^{-4}; weight decay 10^{-4}; gradient clipping 1.0	
Latent editors		
Applied rounds R	12	
Step scale (fraction of $\|z\|$)	0.06	
Optimizer / epochs / batch size	AdamW; learning rate 2×10^{-4}; weight decay 10^{-4} / 6 / 48	
Scene-adaptive readout fusion		
Gate architecture	Two-layer MLP; width 256; dropout 0.1	
Gate optimizer / epochs / batch size	AdamW; lr 5×10^{-4}; wd 10^{-3} / 12 / 256	AdamW; lr 3×10^{-4} / 30 / 64
Gate training seeds	3	
Auxiliary branch weight β	0.15	
Scene-adaptive decoding		
Agreement radius ε	1.5 m	

B PERFORMANCE–LATENCY TRADE-OFF

Figure A1 visualizes the validation RFS and end-to-end latency of our method and representative driving VLA baselines. For both diffusion backbones, our method shifts the operating point toward higher RFS and lower latency relative to the corresponding native decoding pipeline, demonstrating that the proposed scene-adaptive inference improves planning quality while reducing inference cost.

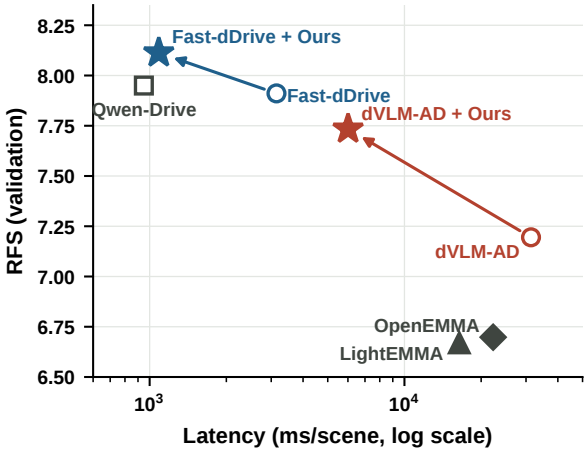


Figure A1: **Planning performance versus end-to-end inference latency.** Validation RFS is plotted against latency per scene on a logarithmic x-axis. Arrows connect each native diffusion VLA with its corresponding scene-adaptive variant.

C Scene Stratification and Category Examples

The scene groups in Figure 3 are defined post hoc using four attributes computed for each validation scene. *Difficulty* is defined by the RFS of the backbone's native decode and partitioned into easy ($\geq$ 9), mid (7–9), and hard ($<$ 7) groups. This quantity reflects the difficulty of a scene for the unmodified backbone rather than a benchmark annotation. *Intent* denotes the manoeuvre of the logged trajectory (straight, left turn, or right turn). *Lateral* is the absolute lateral displacement of the logged trajectory at 5 s and is partitioned into small ($<$ 1 m), mid (1–4 m), and large ($\geq$ 4 m). *Speed* is the ego speed at the current frame and is partitioned into slow ($<$ 3 m/s), mid (3–8 m/s), and fast ($\geq$ 8 m/s). These attributes are derived only from the logged data or the native decode and are used exclusively for post hoc analysis; none is provided to the readout gate or decoding policy.

Figure A2 presents one representative scene from each category. All examples are selected from daytime frames and are chosen to illustrate the corresponding attribute while avoiding overlap with Figure 3c. The examples qualitatively reflect the scene characteristics captured by each stratification criterion, including straight driving, turning manoeuvres, larger lateral displacement, and increased traffic or interaction complexity. Consistent with the quantitative analysis in Figure 3a and Figure 3b, turning and large-lateral groups exhibit more frequent deviations from the default readout layer and decoding stage, and correspondingly larger oracle gains.

D Scene-Adaptive Readout and Decoding Details

D.1 Scene-Adaptive Readout Gate

The scene-adaptive readout gate operates entirely on information available after the first backbone sweep. As described in Sec. 4.2, we construct the scene representation $u(x)$ from both latent scene features and trajectory-level predictions. Specifically, we mean-pool the trajectory-slot and vision-token hidden states to summarize the current scene representation, and concatenate them with the ego history and navigation command. We further include the one-pass draft $a(x)$ and the candidate trajectories $\{\hat{\tau}_\ell(x)\}_{\ell\in\mathcal{B}}$ produced by the layer-specific experts.

To expose disagreement among the experts to the gate, we additionally include the deviation of each candidate trajectory from the mean prediction across experts. These features allow the gate to condition its mixture weights not only on the visual and driving context, but also on how strongly the layer-specific trajectory predictions agree or disagree for a given scene. All features are computed before any additional native decoding is performed.



Figure A2: **Representative examples for each scene stratification category.** Each panel concatenates the front-left, front, and front-right camera views. Rows correspond to the four attributes used in Figure 3, with category definitions provided in Appendix C.

Given $u(x)$, the readout gate g_ψ produces one score for each layer-specific expert in the readout band $\mathcal{B}$. A softmax converts these scores into normalized mixture weights $w_\ell(x)$, and the resulting trajectory is

$$\hat{\tau}_{\text{moe}}(x) = \sum_{\ell \in \mathcal{B}} w_\ell(x) \hat{\tau}_\ell(x). \tag{A1}$$

The gate is trained exclusively on the training split while the diffusion backbone and layer-specific experts are kept fixed. Its objective is to minimize the trajectory loss between the fused prediction and the ground-truth trajectory. Thus, the gate learns scene-dependent weighting over the candidate trajectories rather than using oracle layer assignments as supervision.

Auxiliary reference branch. In addition to the scene-adaptive mixture, we retain an auxiliary readout branch based on a fixed reference layer $\ell^\star$. The reference layer is selected using the training split and is fixed to $\ell^\star = \text{L27}$ in our experiments. The auxiliary branch provides an independent trajectory prediction from the reference representation and is combined with the MoE prediction as described in Sec. 4.2. This branch does not participate in scene-dependent expert routing and is used only to stabilize the final readout prediction.

D.2 Scene-Adaptive Decoding Gate

The scene-adaptive decoding gate determines whether the trajectory obtained from the first-sweep readout is sufficient or whether the scene should enter the native refinement path. Its input $v(x)$ is constructed entirely from quantities available before additional decoding. We mean-pool the pre-decoding trajectory-slot and vision-token hidden states and combine these features with the navigation context and statistics derived from the scene-adaptive readout.

In particular, the gate receives information about the confidence of the readout prediction and the disagreement among the layer-specific experts. These signals complement the latent scene representation: high agreement among experts indicates that multiple readout depths support a similar trajectory, whereas larger disagreement provides evidence that additional native refinement may be useful. Importantly, no intermediate or final native decoder output is required to form $v(x)$ at inference time.

The decoding gate g_ω maps $v(x)$ to a scalar score

$$q_{\mathrm{dec}}(x) = g_\omega(v(x)), \tag{A2}$$

which estimates the benefit of entering the native refinement path. The gate is trained exclusively on the training split using a binary target indicating whether native refinement reduces the ground-truth trajectory error relative to the readout prediction. Ground-truth trajectories are used only to construct this training target and are never accessed by the gate at inference time.

At inference time, a scene is routed according to

$$G(x) = \mathbb{I}[q_{\mathrm{dec}}(x) \geq \lambda]\,. \tag{A3}$$

Scenes with $G(x) = 0$ terminate after the scene-adaptive readout, whereas scenes with $G(x) = 1$ enter the native fallback path. Under the learned decoding policy, 8.35% of validation scenes are routed to the native fallback path. This routing rate is an empirical outcome of the trained gate, not a manually specified decoding budget or hyperparameter.

For scenes entering the fallback path, the gate determines only whether additional decoding should begin. The amount of subsequent computation is determined separately by the stage-wise stopping criterion in Sec. 4.3: intermediate native representations are converted to trajectory predictions by stage-specific probes, and decoding terminates at the earliest stage satisfying the trajectory-agreement criterion. If no intermediate stage satisfies the criterion, the native decoding schedule proceeds to completion.

E Scene-Adaptive Routing and Decoding Statistics

Table A2 summarizes the empirical routing behavior of the learned readout and decoding policies on both validation and test splits. For both backbones, L27 receives the largest mixture weight for most scenes, while neighboring layers receive the dominant weight for a smaller fraction. The resulting layer distributions remain similar across validation and test, indicating that the learned routing behavior is stable across splits.

The decoding policy exhibits a similarly sparse allocation of computation. Approximately 89–92% of scenes terminate immediately after the readout path, while only a small fraction proceeds to additional native decoding. Among the scenes entering the fallback path, most terminate within the first few decoding stages, and only a small percentage reaches native completion. This supports the intended behavior of our scene-adaptive framework, which avoids unnecessary decoding for most scenes while retaining additional refinement for the subset that benefits from it.

Table A2: Empirical routing distributions (%) of the learned readout and decoding policies. For decoding stages, *Readout* denotes termination after the readout path without additional native decoding, while *end* denotes completion of the native decoding schedule.

Readout layer		L21	L22	L23	L24	L25	L26	L27	L28	L29
Fast-dDrive	val	0.2	0.0	2.1	0.0	0.0	0.4	94.4	0.4	2.5
Fast-dDrive	test	0.4	0.0	2.5	0.0	0.0	0.6	93.8	0.7	2.1
dVLM-AD	val	0.6	0.4	3.2	0.4	0.5	1.3	89.7	0.9	3.1
dVLM-AD	test	0.9	0.4	3.8	0.5	0.6	1.3	88.0	1.4	3.0

Decoding stage		Readout	1	2	3	4	5	end	–
Fast-dDrive	val	91.6	5.2	0.8	1.3	0.4	0.0	0.6	–
Fast-dDrive	test	89.6	6.0	2.0	1.1	0.2	0.2	0.9	–
dVLM-AD	val	91.6	5.0	0.3	1.5	0.5	0.1	1.0	–
dVLM-AD	test	88.5	5.6	1.3	1.0	0.6	0.7	2.3	–

F LIMITATIONS

(i) The readout band is backbone-specific and must be identified for each backbone using the analysis in Section 3. (ii) The trainable readout, probe, and routing components are supervised primarily with trajectory-level objectives, whereas RFS is a separate human-aligned evaluation metric; improving alignment between training objectives and RFS may further improve routing quality. (iii) Our evaluation is limited to open-loop trajectory planning, and the benefits of scene-adaptive computation remain to be validated in closed-loop driving. (iv) The oracle analyses rely on evaluator outcomes unavailable at inference and should therefore be interpreted as upper bounds on the potential gain from scene-adaptive routing rather than achievable policy performance.