

Latent Dreaming: Let Language Think Yet Latent See in Driving VLA

Zhixuan You^{1,2*} Ruixu Zhang^{1*} Junjie Wang^{1*}
Yuxin Meng¹ Lisheng Wu³ Lei Zhou³ Wei-Bin Kou^{1†} Yujiu Yang^{1†}

¹Tsinghua University ²Sichuan University

³Yinwang intelligent Technology Co., Ltd

youzhixuan@stu.scu.edu.cn rx-zhang25@mails.tsinghua.edu.cn

{wangjunjie, wbkouqvb, yang.yujiu}@sz.tsinghua.edu.cn

Abstract

Vision-language-action (VLA) models for autonomous driving typically express intermediate reasoning entirely in text, which may discard decision-relevant visual information. We propose Latent Dreaming (LatDM), a latent-text reasoning framework that complements textual chain-of-thought with continuous latent states. LatDM uses latent blocks to preserve visually grounded spatial and temporal information, while text handles semantic abstraction and decision communication. A pretrained visual encoder provides latent supervision during training, and the model autonomously interleaves latent and textual reasoning at inference. Experiments on VLADBench, CODA-LM, and NuInstruct show consistent improvements over the underlying VLA backbone, while ablations confirm that the latent channel contributes beyond standard textual fine-tuning.

1 Introduction

Textual chain-of-thought (CoT) has become a common interface for vision-language-action (VLA) models in autonomous driving (AD). In general, a VLA model first describes its observations and reasoning process in natural language, then generates a driving action. This paradigm implicitly assumes that the hidden states required for VLA reasoning can be adequately expressed in textual space. For dynamic AD tasks, this assumption is fragile (Yu et al., 2026). Critical information such as pedestrian motion trends and local interaction intentions is inherently continuous, spatial, and temporally correlated, rather than naturally represented as discrete textual tokens.

However, existing AD VLA models (Sima et al., 2024; Tian et al., 2025; Wang et al., 2024; Xu

et al., 2024; Shao et al., 2024; Hao et al., 2025) overwhelmingly adopt textual CoT (Sprague et al., 2025; Stechly et al., 2024) as the sole reasoning channel. Textual CoT is asked to simultaneously serve three roles: (i) *visual evidence*, such as describing lanes, agents, and spatial relations with sufficient fidelity for downstream decisions; (ii) *semantic abstraction*, including inferring intent, invoking traffic rules, and weighing risk; and (iii) *answer formatting*, such as producing the final action label, bounding box, or free-form description. Forcing a single discrete token stream to carry all three burdens is representationally inefficient.

We additionally observe that this representational tension is measurable. In a separate diagnostic on NuInstruct reasoning actions under a six-camera, multi-frame protocol, a linear probe on the final hidden states of MiMo-Embodied-7B (Hao et al., 2025) achieves 93.2% accuracy, while its greedily decoded text reaches 72.0%. This ~21-point difference reveals a hidden-to-text readout gap: the decoded response does not recover all decision-relevant information that is linearly available in the hidden states. The diagnostic does not, by itself, identify the vocabulary projection as the unique cause of the remaining errors.

To address this hidden-to-text readout gap, we propose **Latent Dreaming (LatDM)**, an autoregressive latent-text decomposition that assigns each representational medium the role it is best suited for. Concretely, LatDM consists of three stages: visual encoder pre-training, dual-track supervised fine-tuning, and hybrid-mode inference.

Visual Encoder Pre-training. We first train a self-supervised visual encoder that maps each driving image to a fixed-length set of continuous latent vectors. The encoder adopts SAM ViT-B as its visual backbone, followed by a linear projection and a cross-attention module that encodes each frame into 16 vectors of dimension 4096. The training ob-

*Equal contribution. Zhixuan You conducted this work as a research assistant at Tsinghua University.

†Corresponding Author.

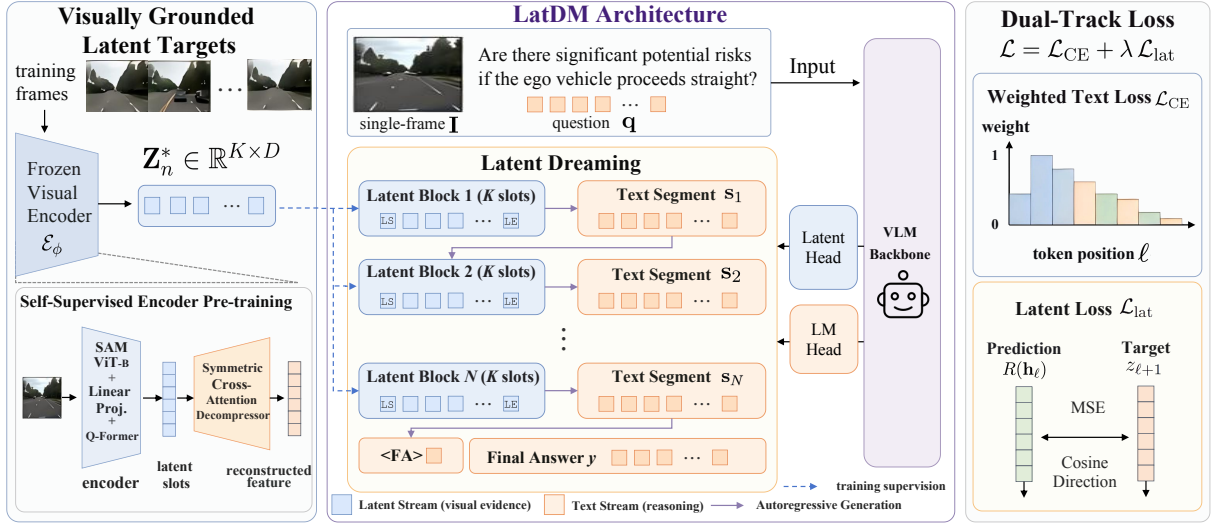


Figure 1: Overview of the proposed LatDM.

jective is to enable the encoder to preserve spatial structure and perceptual detail on a per-frame basis without label supervision. Once trained on $\sim 209k$ unlabeled driving frames, the encoder is frozen for all subsequent stages.

Dual-Track Supervised Fine-tuning. We then fine-tune the VLA base model using LoRA adaptation. Concretely, we introduce N cascading latent-text blocks within a single assistant turn to provide intermediate supervision. Each block contains one `<latent-start>` token, K `<latent-pad>` tokens, and one `<latent-end>` token, followed by a textual reasoning segment. The text segments carry abstraction and communication, while the latent blocks retain visual information in continuous form. We additionally insert a `<final-answer>` boundary token between the last reasoning segment and the final answer span, marking the transition into answer mode. At `<latent-pad>` positions, a continuous regression head predicts latent vectors supervised by the frozen encoder’s output on the corresponding training frame, using a combined MSE and cosine-direction loss. At text positions, the standard language head generates reasoning segments and the final answer under cross-entropy supervision. The two losses are combined to form the fine-tuning objective. To close the train–test exposure gap within latent blocks, we employ a two-phase schedule: (i) a teacher-forcing warm-up that lets the regression head learn a stable next-latent prediction mapping, followed by (ii) autoregressive unrolling where, within each latent block, the first slot is anchored to the ground-truth latent and the subse-

quent $K - 1$ slots use the regression head’s own previous predictions as input embeddings.

Hybrid-Mode Inference. At inference time, the model autonomously switches between text mode and latent mode via a lightweight state machine triggered by boundary tokens. Upon generating `<latent-start>`, the model enters latent mode and autoregressively produces K continuous vectors through the regression head; after the block, it returns to text mode. The assistant turn thus alternates latent blocks and text segments in a structured sequence, terminated by a purely textual final answer following the `<final-answer>` token. The external interface is unchanged: the model receives the observed first frame and a text question, and returns a text answer. No test-time oracle latent is required.

Prior AD-oriented implicit-CoT methods (Lu et al., 2026; Tan et al., 2025; Shang et al., 2026; Peng et al., 2025; Zhang et al., 2026) typically supervise latent variables with teacher trajectories or bounding boxes. In contrast, LatDM derives its latent supervision entirely from a frozen visual encoder independently applied to the corresponding training frames. In sequential SFT examples, later frames serve only as training targets; at inference, only the first frame is observed, and later latent blocks are self-generated. This design requires no planning annotations, making LatDM applicable to VQA tasks beyond trajectory planning.

In summary, the proposed LatDM is illustrated in Figure 1, and the main contributions are:

- We identify and quantify the hidden-to-text information bottleneck in driving VLA models,

revealing a ~ 21.2 gap between hidden-state probe accuracy and text-decoded accuracy.

- We propose LatDM, an autoregression-oriented latent-text decomposition that carries visually supervised hypotheses in continuous latent blocks across the generated sequence, while reserving text for intent inference and decision communication.
- We design a three-stage pipeline comprising self-supervised visual encoder pre-training, dual-track VLA fine-tuning, and automatically switched hybrid-mode inference. Experiments on VLADBench, CODA-LM, and NuInstruct show consistent improvements, and ablations support the contribution of the latent channel.

2 Related Work

2.1 Language-Centric Reasoning in AD VLA

VLAs (Yang et al., 2026) emerged as a prominent paradigm for AD action generation. Works such as DriveLM (Sima et al., 2024), DriveVLM (Tian et al., 2025), OmniDrive (Wang et al., 2024), DriveGPT4 (Xu et al., 2024), LMDrive (Shao et al., 2024), and MiMo-Embodied (Hao et al., 2025) adapt general-purpose multimodal models to driving question answering, scene comprehension, and planning tasks, enabling models to express environmental understanding, behavioral intent, and decision rationale in text. Further methods such as Reason2Drive (Nie et al., 2024) and RAG-Driver (Yuan et al., 2024) improve driving reasoning through textual CoT, retrieval augmentation, or structured rationales. This trend shows that language provides a unified interface for supervision, evaluation, and human-machine interaction in AD VLA systems.

While language-centric reasoning is effective for high-level rules, intent explanation, and answer organization, dynamic AD decisions often depend on continuous, spatially and temporally correlated visual evidence. More broadly, temporal intention localization studies model hidden intent as the cumulative effect of subtle pre-event visual cues rather than a single overt action (Qi et al., 2024). Accordingly, LatDM retains textual reasoning in AD VLAs while introducing continuous visual states into the reasoning process.

2.2 Continuous States for Multimodal Reasoning

In parallel with language-centric reasoning, recent research has begun to question whether intermediate reasoning states must be expressed entirely in textual form. Continuous or implicit CoT methods (Hao et al., 2024; Zelikman et al., 2024; Goyal et al., 2024; Lv et al., 2026) attempt to replace the textual reasoning process with hidden states, continuous embeddings, or latent vectors, demonstrating that language is not necessarily the optimal representational space for all computational steps. In AD, several works introduce world-model objectives (Wang et al., 2023; Hu et al., 2023; Gao et al., 2024), implicit action representations (Peng et al., 2025; Zhang et al., 2026), or trajectory-level latent states (Tan et al., 2025; Shang et al., 2026) to support planning and embodied decision-making. These studies reflect an emerging trend: multimodal decision models are evolving from purely textual reasoning chains toward continuous states.

We focus on employing continuous visual states in reasoning. LatDM interleaves continuous latent vectors with textual tokens along the reasoning trajectory. This design preserves the external interface while allowing intermediate visual information to remain continuous rather than requiring every reasoning step to be rendered as vocabulary tokens.

3 Latent Dreaming (LatDM)

3.1 Overview

We present LatDM, a latent-text decomposition scheme that assigns visual evidence to continuous latent blocks and reserves discrete text tokens for abstraction and communication. The interface is unchanged: the model consumes a driving image and a textual question, and outputs a textual answer.

In LatDM, the **latent channel**’s continuous vectors retain compact geometric and photometric information. The **text channel**’s discrete tokens express traffic rules, intent inference, risk assessment, and the final formatted answer. The two channels play complementary roles. The interleaved structure supports cross-channel information flow: text segments can attend to preceding latent vectors, while later latent blocks condition on preceding text and latent context.

The architecture comprises four components: (i) the *frozen MiMo visual tower*, which encodes the externally provided image; (ii) a separate *frozen latent-target encoder*, which produces com-

pact continuous supervision targets during training; (iii) the *MiMo-Embodied-7B language backbone*, augmented with special boundary tokens and a latent regression head; and (iv) a *hybrid generation protocol* that alternates between text decoding and latent vector regression.

3.2 Visually Grounded Latent Targets

The visual encoder $\mathcal{E}_\phi: \mathbb{R}^{3 \times H \times W} \rightarrow \mathbb{R}^{K \times D}$ maps a single frame into $K = 16$ continuous vectors of width D (matching the backbone hidden size). It is trained once on driving frames and then frozen for all subsequent fine-tuning stages. For any training frame, it produces a fixed supervision target in $\mathbb{R}^{K \times D}$ for the LLM’s latent predictions.

Architecture. Inspired by the SAM ViT-B backbone (Kirillov et al., 2023) and the cross-attention compression design used in Vary (Wei et al., 2024), the visual encoder contains three components: (i) a SAM ViT-B backbone followed by two convolutional downsampling stages, producing a 16×16 spatial feature grid; (ii) a linear projection to the LLM hidden dimension, yielding patch features $\mathbf{p} \in \mathbb{R}^{256 \times D}$; (iii) a Q-Former-style cross-attention bottleneck with $K = 16$ learnable queries that compresses $\mathbf{p}$ into $\mathbf{c} \in \mathbb{R}^{K \times D}$.

Self-supervised pre-training. We train the visual encoder $\mathcal{E}_\phi$ end-to-end with a self-supervised reconstruction objective. A symmetric cross-attention head with 256 learnable queries decompresses $\mathbf{c}$ back to $\hat{\mathbf{p}} \in \mathbb{R}^{256 \times D}$, and we minimize:

$$\mathcal{L}_{\text{rec}} = \text{MSE}(\hat{\mathbf{p}}, \text{sg}[\mathbf{p}]) + 0.1 \cdot (1 - \cos(\hat{\mathbf{p}}, \text{sg}[\mathbf{p}])), \quad (1)$$

where $\text{sg}[\cdot]$ denotes stop-gradient. The pre-training corpus consists of 209,501 unlabeled AD frames sampled from nuScenes (Caesar et al., 2020) and the dynamic-scene subset of VLADBench (Li et al., 2025). Once trained, the visual encoder is retained and *frozen* for all downstream stages.

Design rationale and underlying merit. Using same-frame reconstruction as the encoder objective makes a latent target available for any individual driving frame without 3D boxes, trajectories, or planning annotations. This differs from learning a future-frame predictor: each target is the frozen encoding of the frame being represented. In sequential SFT examples, later observed frames are nevertheless used as training-only targets for later latent blocks.

3.3 Latent-Text Interleaved Trajectory

We introduce the following additional special tokens: $\langle \text{LS} \rangle$ ($\langle | \text{latent_start} | \rangle$), $\langle \text{LP} \rangle$ ($\langle | \text{latent_pad} | \rangle$), and $\langle \text{LE} \rangle$ ($\langle | \text{latent_end} | \rangle$) for latent block opening, slot placeholders, and closing, respectively, and $\langle \text{FA} \rangle$ ($\langle | \text{final_answer} | \rangle$) for the final answer boundary.

The model input consists of the system prompt, the question $\mathbf{q}$, and only the first frame $\mathbf{I}_1$. For a sequential training example, block n is supervised by the frozen target $\mathcal{E}_\phi(\mathbf{I}_n)$; frames $\mathbf{I}_{n>1}$ are used only to construct training targets and are not exposed through the VLA visual input. At inference time, no target frames or cached encoder outputs are available, and all latent blocks are generated by the model. The assistant turn contains N latent blocks, each followed by a text reasoning segment $\mathbf{s}_n$, and terminates with $\langle \text{FA} \rangle$ and the answer y :

$$\begin{aligned} & \underbrace{\langle \text{LS} \rangle \underbrace{\langle \text{LP} \rangle \cdots \langle \text{LP} \rangle}_K \langle \text{LE} \rangle}_{\text{Latent block 1}} \mathbf{s}_1 \\ & \quad \cdots \\ & \underbrace{\langle \text{LS} \rangle \underbrace{\langle \text{LP} \rangle \cdots \langle \text{LP} \rangle}_K \langle \text{LE} \rangle}_{\text{Latent block } N} \mathbf{s}_N \langle \text{FA} \rangle y, \end{aligned} \quad (2)$$

where $\langle \text{LP} \rangle$ at slot k is supervised by a continuous vector $\mathbf{Z}_n^k \in \mathbb{R}^D$ produced by the frozen latent-target encoder for the corresponding training frame, $\mathbf{s}_n$ denotes textual reasoning tokens, and y is the final answer.

3.4 Training Objective and Strategy

Dual-track loss. The training loss jointly supervises text tokens and latent vectors:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{lat}}, \quad (3)$$

where λ balances the two terms across training.

Let $\mathcal{M}_{\text{txt}}$ denote the set of text token positions that contribute to the language modeling loss. We assign per-position weights $w_\ell \in \{1, w_{\text{ans}}\}$, where w_{ans} is applied to the $\langle \text{FA} \rangle$ boundary, the answer tokens, and the closing $\langle | \text{im_end} | \rangle$ token. The text loss is a weighted next-token CE loss:

$$\mathcal{L}_{\text{CE}} = \sum_{\ell \in \mathcal{M}_{\text{txt}}} w_\ell \text{CE}(\mathbf{W} \mathbf{h}_\ell, \mathbf{x}_{\ell+1}) / \sum_{\ell \in \mathcal{M}_{\text{txt}}} w_\ell, \quad (4)$$

where $\mathbf{W}$ is the LLM head and $\mathbf{h}_\ell$ is the hidden state at position ℓ .

Let $\mathcal{M}_{\text{lat}} = \{\ell : \mathbf{x}_{\ell+1} = \langle \text{LP} \rangle\}$ denote the positions whose next token is a latent placeholder in *one* assistant turn. The latent loss supervises each predicted vector against the frozen encoder target, combining MSE with a cosine direction term:

$$\mathcal{L}_{\text{lat}} = \frac{1}{|\mathcal{M}_{\text{lat}}|} \sum_{\ell \in \mathcal{M}_{\text{lat}}} \left[\frac{1}{D} \|R(\mathbf{h}_{\ell}) - \mathbf{Z}_{\ell+1}\|_2^2 + 0.1(1 - \cos(R(\mathbf{h}_{\ell}), \mathbf{Z}_{\ell+1})) \right], \quad (5)$$

where $R : \mathbb{R}^D \rightarrow \mathbb{R}^D$ is a single linear projection layer and $\mathbf{Z}_{\ell+1}$ denotes the ground-truth latent vector at the corresponding $\langle \text{LP} \rangle$ slot, obtained from the frozen visual encoder, and $\mathbf{h}_{\ell}$ is the hidden state at position ℓ . The MSE component encourages magnitude matching, while the cosine component directly penalizes directional mismatch between the predicted and target vectors.

Two-phase training schedule. A fundamental challenge in training autoregressive latent generation is exposure bias: during training, the model receives ground-truth latent vectors as input context, but at inference time it must condition on its own imperfect predictions. Within continuous latent sequences of $K = 16$ steps, errors compound rapidly because there is no discrete argmax to snap predictions to valid tokens.

We address this with a two-phase schedule. In **Phase 1** (the first training epoch), latent blocks are fully teacher-forced: at each position k , the input embedding is set directly to the ground-truth target $\mathbf{Z}_n^{k-1}$ produced by the frozen encoder. This allows the regression head to learn a stable mapping without compounding prediction errors. In **Phase 2** (the subsequent four epochs), we switch to autoregressive unrolling: within each block, the first slot remains anchored to the ground-truth target $\mathbf{Z}_n^0$, and for $k \geq 1$ the model’s own prediction $\hat{\mathbf{Z}}_n^{k-1}$ is used as input for position k . This phase reduces within-block exposure bias and teaches the model to be more robust to its own prediction noise, while retaining the first-slot training anchor.

Backbone adaptation. The backbone LLM is fine-tuned with LoRA adapters throughout both phases. Three additional components are trained from scratch: (i) the regression head $R : \mathbb{R}^D \rightarrow \mathbb{R}^D$; (ii) three trainable input-embedding rows for $\langle \text{LS} \rangle$, $\langle \text{LE} \rangle$, and $\langle \text{FA} \rangle$; and (iii) the corresponding three output rows of the LM head. The $\langle \text{LP} \rangle$ embedding row is never referenced because its input position is always overridden by an injected continuous vector. Because the visual encoder’s output dimension D

matches the backbone hidden size, no auxiliary embedding projection is needed.

3.5 Free-Mode Inference

At inference time, the model autonomously alternates between text and latent modes. It emits text tokens via the language model head until producing $\langle \text{LS} \rangle$, then switches to latent mode and generates exactly $K = 16$ continuous vectors through the regression head R . After K slots, the state machine deterministically emits $\langle \text{LE} \rangle$ and resumes text decoding. This alternation may repeat multiple times within a single response. The token $\langle \text{FA} \rangle$ ends the reasoning phase and triggers the final answer span.

Each latent slot requires one autoregressive Transformer step plus a lightweight linear regression head; it does not invoke a separate diffusion process or visual decoder. No oracle latent vectors are injected, and all continuous states are generated by the model itself.

4 Experiments and Analyses

Experiments are designed to answer: (i) whether LatDM improves over representative driving-specialized and general-purpose VLA/VLM baselines; (ii) to what extent the gains can be attributed to the proposed latent-text training; (iii) where the gains emerge across perception-, relation-, and decision-centric categories; and (iv) how the learned latent channel affects downstream decision beyond textual reasoning alone.

4.1 Experimental Setup

Benchmarks and metrics. We evaluate **LatDM** on three driving-oriented benchmarks: VLAD-Bench (Li et al., 2025), CODA-LM (Chen et al., 2025), and NuInstruct (Ding et al., 2024). VLAD-Bench evaluates fine-grained autonomous-driving understanding across five domains and ten aspect-level scores; we report all aspect-level scores together with the official **Total** score. CODA-LM evaluates open-ended driving-scene responses with three task-level scores: General Perception, Regional Perception, and Driving Suggestion; we report all three scores and additionally report **Total**, computed as their unweighted average. NuInstruct evaluates instruction-following driving tasks; we evaluate the full test set and report **Action Accuracy** (Act. Acc.), defined as the percentage of samples whose predicted action matches the reference action. Predicted actions are extracted with

the same output parser across all evaluated models. All metrics are reported in percentage points.

Evaluation protocol. For baseline models, we follow the original input protocol of each benchmark: CODA-LM uses single-frame inputs, while VLADBench and NuInstruct use multi-frame inputs. In contrast, LatDM is evaluated under a stricter single-frame setting across all benchmarks: only the first frame is provided as external visual input, and subsequent frame-level latent states are generated internally. For reproduced results, including InternVL3.5-8B and LatDM, we follow the official benchmark scoring procedures while retaining the input settings described above.

Baselines and result sources. We compare LatDM with both general-purpose VLMs and driving-specialized VLA/VLMs, including GPT-4o, Qwen2.5-VL-7B (Bai et al., 2025), LLaVA-Video-7B (Zhang et al., 2024), DriveLM-B-4B (Sima et al., 2024), MiniCPM-V-2.5-8B (Yao et al., 2024), CODA-VLM-8B (Chen et al., 2025), InternVL3.5-8B (Wang et al., 2025), and MiMo-Embodied-7B (Hao et al., 2025). Published results are taken from the corresponding benchmark or model papers when available, while InternVL3.5-8B and LatDM are reproduced under the same evaluation protocols. Since LatDM is built on MiMo-Embodied-7B, MiMo-Embodied-7B serves as the closest backbone reference.

Training setup. Unless otherwise specified, LatDM is trained on the full 53,672-sample latent-text supervision mixture with LoRA adaptation. It uses a frozen latent target encoder to provide frame-level latent supervision and follows a two-phase schedule with teacher-forcing warm-up followed by autoregressive latent unrolling. At inference time, LatDM generates latent vectors by itself and produces a standard textual answer; no test-time oracle latent or external visual target is used. Additional implementation details are provided in Appendix A.

4.2 Main Results

Table 1 summarizes the main comparison across three driving-oriented benchmarks. LatDM consistently improves over its MiMo-Embodied-7B backbone and the general-purpose InternVL3.5-8B baseline on available benchmarks. Notably, LatDM uses only single-frame external visual input across all benchmarks, while the baseline results follow the original benchmark input protocols.

The largest gains appear on VLADBench.

LatDM improves the official Total score from 61.95 to 79.43 over its MiMo-Embodied-7B backbone, and also outperforms all reported VLADBench baselines by a large margin. The gains are especially pronounced on relation- and decision-centric aspects, including signal relation, lane relation, intention judgment, and meta decision-making. The complete VLADBench breakdown in Table 3 further shows that the improvement is not limited to the selected aspects in the main table.

On NuInstruct, LatDM improves action accuracy from 81.43 to 85.19 over MiMo-Embodied-7B, indicating that the latent-text interface benefits action-level driving decisions rather than only descriptive scene understanding. On CODA-LM, LatDM achieves the best score on Driving Suggestion and the second-best Total score among models with all three task-level scores available. Although CODA-VLM-8B remains the strongest CODA-LM-specialized reference, LatDM remains competitive while achieving clear advantages on VLADBench and NuInstruct. Overall, these results suggest that driving VLA reasoning benefits from augmenting textual reasoning traces with a continuous latent channel, especially for relation- and decision-centric driving tasks.

4.3 Component Ablations

We ablate key components of LatDM, including its latent prediction interface and boundary design, under the same backbone, training data, and evaluation protocols. All ablation variants are evaluated at Epoch 4 to isolate component effects under a fixed training budget. Table 2 reports compact results on VLADBench and NuInstruct, which directly evaluate temporal driving decisions and action-level prediction.

The ablation results show that LatDM outperforms both the text-only and non-autoregressive latent-training variants under the same Epoch-4 protocol. Removing the latent channel decreases VLADBench Total from 79.02 to 74.64 and NuInstruct action accuracy from 85.04 to 84.44, suggesting that the gains are not explained by standard textual fine-tuning alone. The non-autoregressive variant remains close on VLADBench but drops substantially on NuInstruct. This variant retains continuous latent targets and the regression loss but injects ground-truth latents at every slot during SFT, thereby isolating the contribution of within-block autoregressive rollout.

Removing the explicit <FA> boundary causes

Method	Visual Input	VLADBench						CODA-LM				NuInst.
		Sig. Rel.	Lane Rel.	Int. Judg.	Ego Act.	Meta Dec.	Total	Gen.	RP	Sug.	Total	Act. Acc.
General-purpose VLMs												
GPT-4o	Single/Multi	41.25	51.18	47.79	65.75	48.04	56.00	–	–	–	–	–
Qwen2.5-VL-7B	Single/Multi	28.78	40.14	53.42	58.87	35.48	50.75	–	–	–	–	–
LLaVA-Video-7B	Single/Multi	30.58	53.61	59.52	56.50	41.19	51.73	–	–	–	–	–
MiniCPM-V-2.5-8B	Single/Multi	–	–	–	–	–	–	41.12	<u>57.20</u>	48.48	48.93	–
InternVL3.5-8B	Single/Multi	39.85	52.75	52.40	62.27	45.24	56.58	26.52	45.95	25.04	32.50	59.25
Driving-specialized VLA/VLMs												
DriveLM-B-4B	Single/Multi	30.56	44.06	60.89	56.60	46.31	49.83	–	–	–	–	–
CODA-VLM-8B	Single/Multi	–	–	–	–	–	–	<u>55.04</u>	77.68	58.14	63.62*	–
MiMo-Embodied-7B	Single/Multi	42.02	56.57	57.18	71.57	49.52	61.95	–	–	–	58.55 [†]	81.43
LatDM (Ours)	Single	81.01	84.92	86.08	79.96	72.14	79.43	47.74	56.05	75.51	<u>59.77</u>	85.19

Table 1: Main results on three driving VLA benchmarks. Visual Input follows the definition in Section 4.1. For VLADBench, Total denotes the official benchmark score. For CODA-LM, Total denotes the unweighted average of three task-level scores when all are available. * denotes a CODA-LM-specialized reference result; † indicates that only the CODA-LM Total score is available from the source.

Variant	VLADBench Total	NuInst. Act. Acc.
MiMo-Embodied-7B	59.27	81.43
LatDM (ours)	79.02	85.04
w/o latent channel	74.64	84.44
w/o autoregressive latent rollout	78.10	81.05
w/o (FA) boundary	43.93	83.98

Table 2: Matched component ablations of LatDM at Epoch 4. All rows use the same training budget and evaluation protocol; the LatDM control is distinct from the Epoch 5 checkpoint in Table 1.

VLADBench Total to drop sharply from 79.02 to 43.93, while NuInstruct action accuracy also decreases from 85.04 to 83.98. The boundary marks the transition from interleaved reasoning to answer-only decoding; this ablation therefore evaluates the importance of this boundary-and-decoding control mechanism.

4.4 Fine-grained Analyses

We further analyze where LatDM gains come from using the complete VLADBench aspect breakdown and epoch-level training dynamics.

Complete VLADBench breakdown. Table 3 reports the complete ten-aspect breakdown of VLADBench across its five official domains. LatDM improves over both MiMo-Embodied-7B and InternVL3.5-8B on all ten aspects. The largest gains appear in relation-, intention-, and decision-centric aspects, including Signal Element Relation, Lane Element Relation, Intention Judgment, and Meta Decision-Making. This trend is consistent with our motivation that continuous latent blocks

Domain	Aspect	MiMo	InternVL	LatDM
TKU	Road Traffic Signals	67.78	67.77	74.25
	Road Passage Provisions	79.55	72.94	82.72
GER	Background	69.29	71.07	86.61
	Foreground	71.89	59.44	77.44
TGG	Signal Element Relation	42.02	39.85	81.01
	Lane Element Relation	56.57	52.75	84.92
TAC	Intention Judgment	57.18	52.40	86.08
	Behavior Understanding	51.96	40.11	57.99
EDP	Ego Action Reasoning	71.57	62.27	79.96
	Meta Decision-Making	49.52	45.24	72.14
Total		61.95	56.58	79.43

Table 3: Complete aspect-level results on VLADBench.

provide an additional channel for preserving structured visual evidence required by traffic interactions and driving decisions.

Training dynamics. Figure 2 shows downstream performance across training epochs on the full 53K mixture. VLADBench improves after warm-up and then approaches its final score, while CODA-LM and NuInstruct action accuracy fluctuate within a narrow range in later epochs. This suggests that autoregressive latent unrolling does not lead to brittle checkpoint behavior.

4.5 Hidden-to-Text Bottleneck Analysis

We further examine the **hidden-to-text bottleneck** discussed in Sec. 1 after LatDM training. This diagnostic separates two factors: whether the correct driving action is encoded in the model’s continuous hidden states, and whether the same action can be recovered from the model’s final text output. To this end, we compare two readouts from the same input: a linear probe trained on hidden

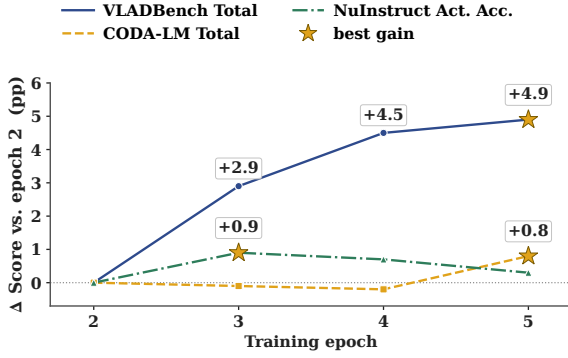


Figure 2: Training dynamics across three benchmarks.

Model	Pre-LLM	Post-LLM	Text	Gap
Frozen MiMo (Think)	91.42	90.45	56.44	34.01
LatDM (Free-Mode)	90.77	91.01	76.19	14.82
Δ vs. frozen	-0.65	+0.56	+19.75	-19.19

Table 4: Hidden-to-text gap before and after LatDM on NuInstruct binary action prediction. Post-LLM probe accuracy remains nearly unchanged, while text-decoded action accuracy improves substantially.

states and the action parsed from greedy text decoding. If the hidden-state probe is accurate but the decoded text is inaccurate, then decision-relevant information is present internally but lost during hidden-to-text projection. We define the hidden-to-text gap as Post-LLM probe accuracy minus greedy text-decoded action accuracy.

We conduct this diagnostic on NuInstruct under a matched single-frame binary action setting for both the frozen MiMo backbone and LatDM. For both models, we use the same front-camera frame as visual input and normalize the output space into binary driving actions. We fit 5-fold stratified linear probes on Pre-LLM visual features and Post-LLM final hidden states, and compare them with greedy text-decoded action accuracy.

Table 4 shows that the frozen MiMo backbone already contains strong action information in its continuous hidden states, but much of this information is not recovered by text decoding. Under the matched single-frame setting, the frozen model achieves 90.45% Post-LLM probe accuracy but only 56.44% text-decoded accuracy, resulting in a 34.01-point hidden-to-text gap. After LatDM training, Post-LLM probe accuracy changes only marginally, from 90.45% to 91.01%, whereas text-decoded accuracy rises from 56.44% to 76.19%. This reduces the hidden-to-text gap

from 34.01 to 14.82 points. These results suggest that LatDM mainly improves the readout of decision-relevant information into textual answers, rather than merely making the internal hidden state more linearly separable. Additional protocol details, full per-position probes, and representation geometry are provided in Appendix D.2.

4.6 Case Studies

Figure 3 shows a representative VLADBench *Intention Judgment* case where the boxed motorcycle gradually moves toward the ego lane and the ground truth is yes. This case requires temporal evidence across frames: the cut-in intention is ambiguous from a single image. LatDM observes only the first frame and generates three future-oriented latent hypotheses. Its interleaved latent-text trace anticipates the possible lane change and answers yes. In contrast, the frozen MIMO-EMBODIED-7B backbone observes all four frames, but its text chain of thought concludes that no lane-changing behavior is visible, leading to an incorrect *no*. This example illustrates that self-generated latent hypotheses can support action reasoning from a single observation.

5 Conclusion

We introduced **LatDM**, a latent-text decomposition framework for AD VLA models. LatDM augments textual reasoning with a continuous latent channel supervised by a frozen visual encoder and performs hybrid latent-text generation without oracle latent inputs. Across VLADBench, CODA-LM, and NuInstruct, LatDM improves over representative baselines, while ablations and analyses show that the latent channel benefits relation- and decision-centric reasoning beyond text-only fine-tuning. These results suggest that continuous latent reasoning can serve as a useful complement to text-only CoT for visually grounded driving decision-making.

6 Limitations

LatDM currently adopts a single-frame external-input protocol at inference time: only the first frame is provided as visual input, while subsequent frame-level latent states are generated internally. This setting is stricter than standard multi-frame or multi-view benchmark protocols, but it may also introduce accumulated errors over long temporal horizons, heavy occlusion, or rare dynamic scenar-

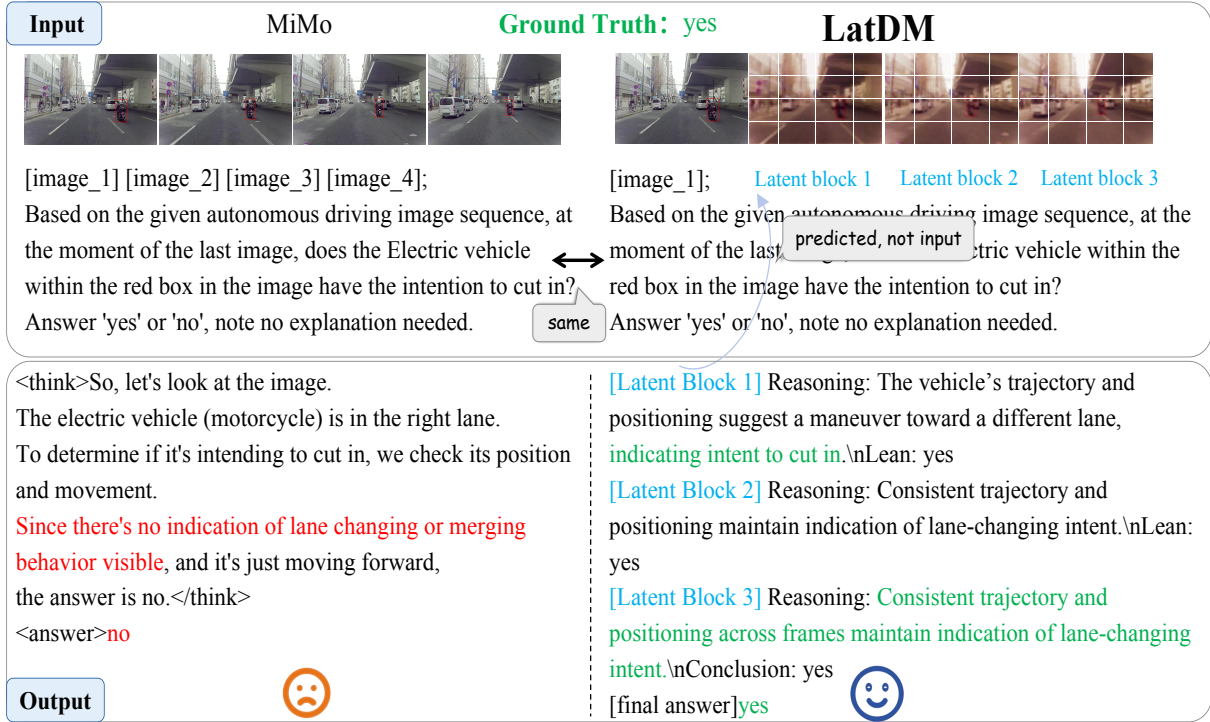


Figure 3: Case study on VLADBench *Intention Judgment*.

ios. Future work may combine LatDM with sparse additional visual observations for more robust long-term driving reasoning.

Acknowledgments

This work was partly supported by the National Natural Science Foundation of China (Grant No. 62576191) and the Tsinghua SIGS KA Cooperation Fund. This work was also partly supported by the Open Research Fund of the State Key Laboratory of Multimodal Artificial Intelligence Systems (Grant No. MAIS2025080).

References

- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *CoRR*, abs/2502.13923.
- Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, pages 11618–11628. Computer Vision Foundation / IEEE.
- Kai Chen, Yanze Li, Wenhua Zhang, Yanxin Liu, Pengxiang Li, Ruiyuan Gao, Lanqing Hong, Meng Tian, Xinhai Zhao, Zhenguo Li, Dit-Yan Yeung, Huchuan Lu, and Xu Jia. 2025. Automated evaluation of large vision-language models on self-driving corner cases. In *WACV*, pages 7817–7826. IEEE.
- Xinpeng Ding, Jianhua Han, Hang Xu, Xiaodan Liang, Wei Zhang, and Xiaomeng Li. 2024. Holistic autonomous driving understanding by bird’s-eye-view injected multi-modal large models. *CoRR*, abs/2401.00988.
- Shenyuan Gao, Jiazhi Yang, Li Chen, Kashyap Chitta, Yihang Qiu, Andreas Geiger, Jun Zhang, and Hongyang Li. 2024. Vista: A generalizable driving world model with high fidelity and versatile controllability. In *NeurIPS*.
- Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. In *ICLR*. OpenReview.net.
- Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *CoRR*, abs/2412.06769.
- Xiaoshuai Hao, Lei Zhou, Zhijian Huang, Zhiwen Hou, Yingbo Tang, Lingfeng Zhang, Guang Li, Zheng Lu, Shuhuai Ren, Xianhui Meng, Yuchen Zhang, Jing Wu, Jinghui Lu, Chenxu Dang, Jiayi Guan, Jianhua Wu, Zhiyi Hou, Hanbing Li, Shumeng Xia, and 25 others. 2025. Mimo-embodied: X-embodied foundation model technical report. *CoRR*, abs/2511.16518.
- Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton,

- and Gianluca Corrado. 2023. GAIA-1: A generative world model for autonomous driving. *CoRR*, abs/2309.17080.
- Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. 2023. Segment anything. In *ICCV*, pages 3992–4003. IEEE.
- Yue Li, Meng Tian, Zhenyu Lin, Jiangtong Zhu, Dechang Zhu, Haiqiang Liu, Yueyi Zhang, Zhiwei Xiong, and Xinhai Zhao. 2025. Fine-grained evaluation of large vision-language models in autonomous driving. In *ICCV*, pages 9431–9442. IEEE.
- Jinghui Lu, Jiayi Guan, Zhijian Huang, Jinlong Li, Guang Li, Lingdong Kong, Yingyan Li, Han Wang, Shaoqing Xu, Yuechen Luo, Fang Li, Chenxu Dang, Junli Wang, Tao Xu, Jing Wu, Jianhua Wu, Xiaoshuai Hao, Wen Zhang, Tianyi Jiang, and 31 others. 2026. Onevl: One-step latent reasoning and planning with vision-language explanation. *CoRR*, abs/2604.18486.
- Bo Lv, Yasheng Sun, Junjie Wang, and Haoxiang Shi. 2026. Onelntent: Single-token compression for visual latent reasoning. *CoRR*, abs/2602.13738.
- Ming Nie, Renyuan Peng, Chunwei Wang, Xinyue Cai, Jianhua Han, Hang Xu, and Li Zhang. 2024. Reason2drive: Towards interpretable and chain-based reasoning for autonomous driving. In *ECCV (26)*, Lecture Notes in Computer Science, pages 292–308. Springer.
- Qihang Peng, Xuesong Chen, Chenye Yang, Shaoshuai Shi, and Hongsheng Li. 2025. Colavla: Leveraging cognitive latent reasoning for hierarchical parallel trajectory planning in autonomous driving. *CoRR*, abs/2512.22939.
- Zehao Qi, Ruixu Zhang, Xinyi Hu, Wenxuan Liu, and Zheng Wang. 2024. Predicting the unseen: A novel dataset for hidden intention localization in pre-abnormal analysis. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 9690–9698. ACM.
- Shuyao Shang, Bing Zhan, Yunfei Yan, Yuqi Wang, Yingyan Li, Yasong An, Xiaoman Wang, Jierui Liu, Lu Hou, Lue Fan, Zhaoxiang Zhang, and Tieniu Tan. 2026. Dynvla: Learning world dynamics for action reasoning in autonomous driving. *CoRR*, abs/2603.11041.
- Hao Shao, Yuxuan Hu, Letian Wang, Guanglu Song, Steven L. Waslander, Yu Liu, and Hongsheng Li. 2024. Lmdrive: Closed-loop end-to-end driving with large language models. In *CVPR*, pages 15120–15130. IEEE.
- Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. 2024. Drivelm: Driving with graph visual question answering. In *ECCV (52)*, Lecture Notes in Computer Science, pages 256–274. Springer.
- Zayne Rea Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2025. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. In *ICLR*. OpenReview.net.
- Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024. Chain of thoughtlessness? an analysis of cot in planning. In *NeurIPS*.
- Shuhan Tan, Kashyap Chitta, Yuxiao Chen, Ran Tian, Yurong You, Yan Wang, Wenjie Luo, Yulong Cao, Philipp Krähenbühl, Marco Pavone, and Boris Ivanovic. 2025. Latent chain-of-thought world modeling for end-to-end driving. *CoRR*, abs/2512.10226.
- Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xian-Peng Lang, and Hang Zhao. 2025. Drivevlm: The convergence of autonomous driving and large vision-language models. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 4698–4726. PMLR.
- Shihao Wang, Zhiding Yu, Xiaohui Jiang, Shiyi Lan, Min Shi, Nadine Chang, Jan Kautz, Ying Li, and José M. Álvarez. 2024. Omnidrive: A holistic llm-agent framework for autonomous driving with 3d perception, reasoning and planning. *CoRR*, abs/2405.01533.
- Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, and 56 others. 2025. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *CoRR*, abs/2508.18265.
- Xiaofeng Wang, Zheng Zhu, Guan Huang, Xinze Chen, Jiagang Zhu, and Jiwen Lu. 2023. Drivedreamer: Towards real-world-driven world models for autonomous driving. *CoRR*, abs/2309.09777.
- Haoran Wei, Lingyu Kong, Jinyue Chen, Liang Zhao, Zheng Ge, Jinrong Yang, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. 2024. Vary: Scaling up the vision vocabulary for large vision-language model. In *ECCV (4)*, Lecture Notes in Computer Science, pages 408–424. Springer.
- Zhenhua Xu, Yujia Zhang, Enze Xie, Zhen Zhao, Yong Guo, Kwan-Yee K. Wong, Zhenguo Li, and Hengshuang Zhao. 2024. Drivegpt4: Interpretable end-to-end autonomous driving via large language model. *IEEE Robotics Autom. Lett.*, 9(10):8186–8193.
- Tianhang Yang, Yanze Zheng, Junjie Wang, Wei-Bin Kou, Ruotong Li, and Yujiu Yang. 2026. Route

by kinematics, act by observation: Kinematics-supervised expert routing in moe-augmented vla. *arXiv preprint arXiv:2607.26807*.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, and 1 others. 2024. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*.

Kejin Yu, Yuhan Sun, Taiqiang Wu, Ruixu Zhang, Zhiqiang Lin, Yuxin Meng, Junjie Wang, and Yujiu Yang. 2026. A survey of reasoning in autonomous driving systems: Open challenges and emerging paradigms. *Trans. Mach. Learn. Res.*, 2026.

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

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. 2024. Quiet-star: Language models can teach themselves to think before speaking. *CoRR*, abs/2403.09629.

Tianqiu Zhang, Muyang Lyu, Yufan Zhang, Fang Fang, and Si Wu. 2026. Dila: Disentangled latent action world models. *arXiv preprint arXiv:2605.15725*.

Yuanhan Zhang, Bo Li, Haotian Liu, Yong Jae Lee, Liangke Gui, Di Fu, Jiashi Feng, Ziwei Liu, and Chunyuan Li. 2024. *Llava-next: A strong zero-shot video understanding model*.

Appendix

This appendix provides supplementary details and analyses that support the main paper. Appendix A describes implementation details of LatDM, including visual-encoder pre-training, dual-track SFT, and free-mode inference. Appendix B summarizes evaluation details such as benchmark splits, metrics, and input protocols. Appendix C reports additional experimental results, including hyperparameter sensitivity. Appendix D provides mechanistic analyses of the latent channel, including attention rollout, hidden-state probing, and slot-level probing.

A Method and Implementation Details

A.1 Visual Encoder Pre-training

We train the visual encoder $\mathcal{E}_\phi$ (Sec. 3.2) once with a self-supervised feature reconstruction objective and then keep it frozen. It uses a SAM ViT-B backbone followed by two convolutional downsampling stages, a linear projection to the LLM hidden width $D = 4096$, and a Q-Former-style cross-attention

bottleneck with $K = 16$ learnable queries to produce latent slots $\mathbf{c} \in \mathbb{R}^{16 \times D}$. A symmetric decompressor reconstructs the projected patch features for the feature-space loss in Eq. 1.

The encoder is pre-trained on 209,501 unlabeled driving frames from nuScenes (Caesar et al., 2020) and VLADBench (Li et al., 2025). We train for two epochs with AdamW, fp16 precision, a cosine schedule, and an effective batch size of 24 on six RTX 4090 GPUs. The learning rates are 1×10^{-5} for the SAM backbone and 1×10^{-4} for the newly initialized modules. After pre-training, the encoder is frozen and used to construct frame-level latent targets $\{\mathbf{Z}_n^*\}_{n=1}^T \in \mathbb{R}^{T \times K \times D}$ for SFT. The encoder is never trained to predict a different future frame. During SFT, however, sequential examples use the frozen encodings of later observed frames as training-only targets for subsequent latent blocks; only the first frame is exposed through the VLA visual input.

A.2 Latent-Text Dual-Track SFT

The SFT stage adapts MiMo-Embodied-7B with the dual-track objective in Eq. 3. The base backbone and visual tower are kept frozen; we train only LoRA adapters, the latent regression head R , and the input/output embedding rows of $\langle \text{LS} \rangle$, $\langle \text{LE} \rangle$, and $\langle \text{FA} \rangle$. The $\langle \text{LP} \rangle$ row remains frozen because every latent-slot input is replaced by a continuous vector.

Textual-reasoning supervision. We construct intermediate textual reasoning targets with the frozen MiMo model before SFT. For each target segment, the model receives the question, the corresponding training frame, the reference answer, and any preceding reasoning segments. These targets are therefore answer-conditioned rationales rather than independently generated solutions. The conclusion of each target is deterministically aligned with the reference answer.

The box below reports the representative prompt used for a multi-frame, discrete-answer training example. Placeholders in braces are populated from the training manifest. This prompt is used only to construct SFT targets offline; later training frames are not supplied to LatDM at inference time.

Representative multi-segment reasoning prompt

System message

You are an autonomous driving reasoning assistant. Look at the scene, reason about other road users and traffic rules, then produce the appropriate ego-vehicle action.

User message for segment t

Question: {question}

Reference answer: {reference answer}

Previous reasoning so far: {segments 1, ..., $t - 1$ }

The previous-reasoning field is omitted for the first segment.

Frame-specific instruction

- $t = 1$: This is the first frame (frame 1 of N). The final answer is known to be "{reference answer}." Start the reasoning trace toward this answer.
- $1 < t < N$: This is frame t of N . State what is new in this frame's contribution, building on the accumulated reasoning without restating it.
- $t = N$: This is the final frame. Consolidate the accumulated reasoning across all frames and commit to the reference answer.

Reasoning constraint

Do not directly describe visual content. Reason from intent inference, traffic rules, and prior expectations. For example, prefer an abstract statement such as "Standard right-of-way rules and the absence of conflicting intent support proceeding" to a literal inventory of objects, colors, or positions in the frame.

Required output

Reasoning: <1-2 abstract sentences focused on this step>

Lean: {reference answer} ($t < N$)

Conclusion: {reference answer} ($t = N$)

For a five-frame manifest example, this produces four *Reasoning-Lean* segments followed by one *Reasoning-Conclusion* segment.

Hyperparameter	Value
Base model	MiMo-Embodied-7B
Trainable modules	LoRA, R , <LS>/<LE>/<FA> rows
LoRA rank / α / dropout	128/256/0.05
LoRA targets	attention and MLP projections
Regression head R	Linear($D \rightarrow D$), $D = 4096$
Latent slots K	16
Latent loss	MSE + 0.1(1 - cos)
Latent loss weight λ	10.0
Answer-span weight w_{ans}	3.0
Max sequence length	2048
Visual max pixels	802,816
Training samples	53,672
Epochs	5.0
Batch / accum. / GPU's	1/8/8
LoRA LR / head LR	$2 \times 10^{-5} / 2 \times 10^{-4}$
Optimizer	AdamW, wd 0.01
Precision	bf16
Phase 1 / Phase 2	first 1 / subsequent 4 epochs

Table 5: Dual-track SFT hyperparameters. The base backbone and visual tower are frozen throughout training.

Training is performed on 8 NVIDIA H20 GPUs. We use a cosine learning-rate schedule with 5% warm-up and DeepSpeed ZeRO-2. The latent targets are direct outputs of the frozen latent-target encoder without additional normalization. During Phase 1, ground-truth latent targets are provided at every latent position. During Phase 2, the first slot in each block remains anchored to its target and the remaining slots are generated autoregressively.

A.3 Free-Mode Inference

For LatDM, all results in Tab. 1 are obtained with the free-mode inference protocol in Sec. 3.5. The model autonomously alternates between text tokens and latent blocks before emitting the final answer. No oracle latent vectors or encoder-generated targets are provided at inference time.

Each latent block contains $K = 16$ slots. Free-mode inference imposes no hard lower bound on the number of latent blocks, and the model may emit <FA> before four blocks have been generated. The four-block setting acts only as a fallback for long pre-answer text segments: while fewer than four blocks have been emitted, the state machine inserts <LS> when a pre-answer text segment reaches its allocated budget. It does not prevent the model from emitting <FA> earlier. We cap the maximum number of latent blocks at eight. After the model emits <FA>, latent-control tokens are forbidden and decoding continues only in the textual answer mode until <|im_end|>. We use greedy decoding with repetition penalty disabled. The text-token budget is 512 for VLADBench and NuInstruct, and 1024 for CODA-LM.

Complexity and parameter overhead. LatDM does not introduce a separate model-scale branch. If standard decoding has length L , adding B latent blocks with K slots extends the sequence to approximately $L + BK$, so the Transformer attention complexity remains $O((L + BK)^2)$. Each latent slot requires one autoregressive Transformer step followed by a linear $D \rightarrow D$ regression head. Beyond LoRA, the trainable additions comprise this head and the input/output rows for three special tokens. No diffusion process, visual decoder, oracle latent, or future observation is used at inference time.

B Benchmark and Evaluation Details

We evaluate LatDM on three autonomous driving VLA benchmarks: VLADBench, CODA-LM, and

Measure	NVIDIA	Ascend	Difference or agreement
Successful latent-target construction (%)	100.0	100.0	0.0 pp
Mean latent-target slot norm	60.6615	60.6615	$2.36 \times 10^{-6}\%$
Paired latent-target cosine similarity	–	–	0.999999999986
Mean relative latent-target ℓ_2 error	–	–	5.30×10^{-6}
Mean reasoning segments per example	2.73	2.73	100.0% agreement
Mean tokens per reasoning segment	35.76	35.78	0.07%
Reasoning segments shorter than 10 tokens (%)	0.0	0.2	0.2 pp
Examples containing repeated reasoning segments (%)	6.0	8.5	2.5 pp
Reasoning segments containing visual terms (%)	3.5	3.7	0.2 pp
Greedy-decoded reasoning-segment ROUGE-L	–	–	0.731
Mean tokens per complete training sequence	1297.7	1297.5	0.02%
Training sequences truncated at 2,048 tokens (%)	0.0	0.0	0.0 pp

Table 6: Cross-accelerator comparison of training-target generation on the same 200-example evaluation subset. The final column reports either the difference between aggregate statistics or paired agreement between outputs, as appropriate. The comparison assesses consistency across the two hardware platforms and does not establish equivalence between trained models or downstream performance.

NuInstruct. For each benchmark, we follow the official evaluation protocol when available and report the metric used in Tab. 1.

For VLADBench, following the official scoring protocol, we evaluate the 28 scored tertiary tasks with 11,193 QA instances; the Trajectory task is excluded from the reported Total. LatDM is evaluated with the first frame as its visual input. We report the official rule-based score.

For CODA-LM, we evaluate on the official test split with 2,123 QA instances. This benchmark is image-based, so LatDM follows the standard single-image input setting. We report the official GPT-based S1/S2/S3 scores.

For NuInstruct, we evaluate on the official test split with 1,330 samples. Since the original benchmark contains multi-view driving inputs, LatDM uses the first front-camera frame under its single-frame latent interface. We report action accuracy by extracting the final driving action from each free-form response and mapping it to the canonical action labels before comparison.

Overall, LatDM is evaluated under a constrained visual interface for benchmarks that originally provide sequential or multi-view visual observations.

C Additional Experimental Results

C.1 Hyperparameter Sensitivity

Figure 4 reports hyper-parameter sweeps on the 7K subset at epoch 5. Across the tested settings, LatDM remains stable around the selected configuration, suggesting that the observed gains are not caused by a brittle hyper-parameter choice.

C.2 Cross-Accelerator Consistency of Training-Target Generation

To assess whether training-target generation is consistent across accelerator platforms, we repeat the construction of latent and textual reasoning targets on Ascend hardware. The evaluation uses a fixed, stratified sample of 200 training examples, comprising 80 examples from CODA-LM, 60 from VLADBench, and 60 from NuInstruct. The VLADBench portion spans all scored task categories present in the sampled data, and the full evaluation includes both single-frame and multi-frame inputs. With identical model weights and preprocessing configurations, we execute the frozen latent-target encoder in FP32 and MiMo-Embodied-7B in BF16 on two Ascend 910B2 NPUs. On NVIDIA hardware, the latent targets are generated on H20 GPUs and the textual reasoning targets on RTX 4090 GPUs. The Ascend textual-reasoning generation uses random seed 20260828 and retains the original prompting and decoding protocol: greedy decoding for the first and final segments and sampling with $T = 0.6$ and $p = 0.95$ for intermediate segments. The comparison is restricted to the construction of training targets before SFT and does not evaluate the equivalence of trained models or benchmark performance.

All 200 examples yield valid targets on Ascend. The latent targets are numerically consistent across platforms: tensor shapes match for every example, the mean paired cosine similarity is 0.999999999986, and the mean relative ℓ_2 error is 5.30×10^{-6} . Because text generation is autoregressive, small hardware-dependent numeri-

cal differences may alter surface forms even under matched decoding settings. We therefore compare distributional statistics and paired text similarity rather than requiring byte-identical outputs. For greedily decoded segments, the mean ROUGE-L is 0.731. Mean segment length and mean full-sequence length differ by 0.07% and 0.02%, respectively. These results support consistent training-target generation across the two hardware platforms for this fixed subset; they do not establish equivalence for complete training runs, learned models, or downstream performance.

D Mechanistic Analysis of the Latent Channel

D.1 Attention Rollout

Sec. 3.1 posits an information flow between the latent and text channels: text reasoning tokens can attend to latent vectors during generation. During SFT, the per-frame supervision in Sec. 3.3 aligns latent block t with the target encoding of frame t ; under free-mode inference, however, that block is self-generated from the preceding context and receives no frame- t pixels. We examine these properties by recording per-layer attention weights for generated reasoning-text tokens. The analysis uses a 60-sample task-balanced subset: 20 NuInstruct multi-frame samples, 20 VLADBench dynamic samples, and 20 CODA-LM single-frame samples. To analyse attention along the generated sequence, we use a two-pass procedure: we first generate each response with the standard free-mode protocol and then replay the resulting sequence while recording attention weights.

Generated latent hypotheses receive more non-text attention on sequential examples. Aggregated across reasoning-text tokens, the 7K LatDM checkpoint allocates 13.2% of its attention to latent slots on dynamic VLADBench samples, compared with 2.9% to ViT image patches on the same samples. Thus, reasoning reads the self-generated latent trajectory more strongly than the observed first-frame patch features; this comparison concerns attention allocation, not access to later frames. On the single-frame CODA-LM split, the latent attention share drops below 2% across all checkpoints, consistent with less reliance on an extended sequence of future-oriented latent hypotheses.

Latent attention is block-local. Fig. 5 shows that reasoning tokens emitted after latent block b

attend $6.4\text{--}10.3\times$ more to block b 's slots than to earlier blocks, with off-diagonal cells consistently below 1% on the 53K checkpoints. This attention pattern is consistent with block-local readout under the temporal correspondence taught by the SFT targets. It does not establish a causal mechanism or indicate that the corresponding frame is observed at inference. The 53K main checkpoint attains the sharpest alignment ratio ($9.6\times$) despite assigning fewer attention share to latent overall, indicating more concentrated block-local attention in this checkpoint.

D.2 Additional Details for Hidden-State Probing

This section provides additional details for the hidden-state probing analysis in Sec. 4.5. The main text reports the hidden-to-text bottleneck diagnostic and its headline result: LatDM largely preserves Post-LLM hidden-state decodability while substantially improving text-decoded action accuracy. Here we supplement that analysis with (i) the full diagnostic protocol, (ii) token-position definitions along the LatDM assistant trace, (iii) full per-position probing results with Wilson confidence intervals, and (iv) representation-geometry measurements between the latent and text channels.

Protocol for the hidden-to-text bottleneck diagnostic. The hidden-to-text bottleneck analysis in Sec. 4.5 is designed as a controlled diagnostic rather than as a main benchmark result. We evaluate both the frozen MiMo backbone and LatDM on the same NuInstruct planning-with-reasoning examples, restrict the output space to binary driving actions, and feed both models the same single front-camera frame. This matched setting removes differences caused by camera coverage, label normalization, or benchmark-specific decoding. Therefore, the relevant quantity is the gap between Post-LLM hidden-state decodability and text-decoded action accuracy, rather than the absolute accuracy under the original benchmark protocol.

For the hidden-state readout, we train 5-fold stratified linear probes with per-fold standardization. We probe two representations: the mean-pooled Pre-LLM visual features and the Post-LLM hidden state at the user-prompt boundary. Text accuracy is obtained by greedily decoding the same example and normalizing the generated answer into the same binary action space. We use the frozen model's Think mode as the text comparator be-

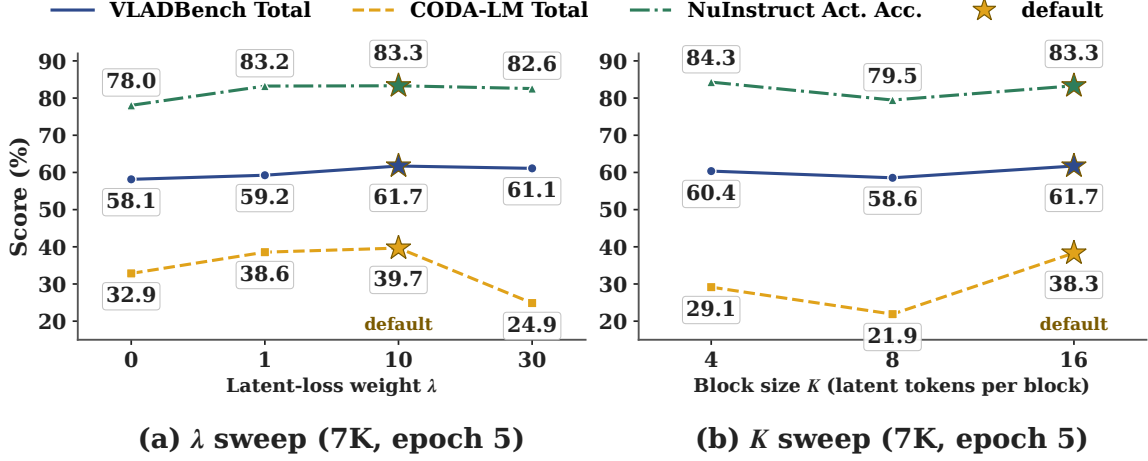


Figure 4: **Hyperparameter sweeps on the 7K subset.** We evaluate LatDM at epoch 5 under different hyperparameter settings. Performance remains stable around the selected configuration, indicating that the method is not overly sensitive to a narrow hyper-parameter choice.

cause LatDM’s free-mode generation also emits an explicit reasoning trace, consisting of latent blocks, text reasoning, and a final-answer boundary. These are the closest available reasoning-enabled settings. A suppressed NoThink mode is not used as the main comparator because it removes the reasoning trace rather than testing whether a reasoning trace can faithfully externalize the hidden decision signal.

Position taxonomy. To analyse where action-predictive information appears along the trained model’s generation trace, we bucket every last-layer hidden vector produced during free-mode inference by the role of its token. *Pre-LLM* denotes the mean-pooled output of the frozen MiMo visual tower. *Post-LLM* denotes the last hidden state at the user-prompt boundary. $\langle \text{LS} \rangle$ and $\langle \text{LE} \rangle$ denote the latent-block open and close boundary tokens. $\langle \text{LP} \rangle\text{-b1}$ denotes the K slots of the first latent block, which re-encode the visible first frame. $\langle \text{LP} \rangle\text{-brest}$ denotes the $K \times (N-1)$ self-generated slots of subsequent latent blocks; during SFT their targets are encodings of later frames, but those frames are not provided as pixel inputs at inference. Text-reasoning denotes the text span between the last $\langle \text{LE} \rangle$ and the $\langle \text{FA} \rangle$ boundary, while *answer* denotes the span after $\langle \text{FA} \rangle$.

Full per-position probing results. Table 7 reports the 5-fold linear-probe accuracy and Wilcoxon 95% confidence intervals for every position bucket. The imagined latent positions, denoted as $\langle \text{LP} \rangle\text{-brest}$, achieve the highest accuracy in the en-

Bucket	n	Acc.	95% CI
Pre-LLM	1235	90.77	[89.03, 92.26]
Post-LLM	1235	91.01	[89.29, 92.48]
$\langle \text{LS} \rangle$	1235	86.64	[84.63, 88.42]
$\langle \text{LP} \rangle\text{-all}$	1235	87.21	[85.23, 88.95]
$\langle \text{LP} \rangle\text{-b1}$	1235	87.04	[85.06, 88.80]
$\langle \text{LP} \rangle\text{-brest}$	675	92.15	[89.87, 93.95]
$\langle \text{LE} \rangle$	1235	89.72	[87.90, 91.29]
text-reasoning	1235	89.88	[88.07, 91.44]
$\langle \text{FA} \rangle$	1235	88.91	[87.03, 90.54]
answer span	1235	86.80	[84.80, 88.58]

Table 7: Per-token-bucket probing accuracy along the trained model’s free-mode trace on NuInstruct binary actions. Pre-LLM denotes mean-pooled frozen visual-encoder outputs; Post-LLM denotes the prompt-end hidden state; $\langle \text{LP} \rangle\text{-b1}$ denotes the visible-frame latent block; and $\langle \text{LP} \rangle\text{-brest}$ denotes latent blocks for subsequent imagined frames. Unless otherwise specified, $n = 1,235$ and the majority baseline is 74.1%. These are the binary-labeled examples from the 1,240-sample validation reasoning subset; five non-binary labels are excluded.

tire trace (92.15%), exceeding both the Post-LLM hidden state (91.01%) and the visible-frame latent block $\langle \text{LP} \rangle\text{-b1}$ (87.04%). Thus, action labels are most linearly decodable at the later generated latent positions in this diagnostic. This association does not establish that these positions simulate future observations. The overall trace also exhibits a double-peak pattern: a prompt-end peak, a later latent-block peak, and then a gradual decrease toward the discrete answer, showing that decision-relevant information remains linearly decodable across the latent-to-text transition.

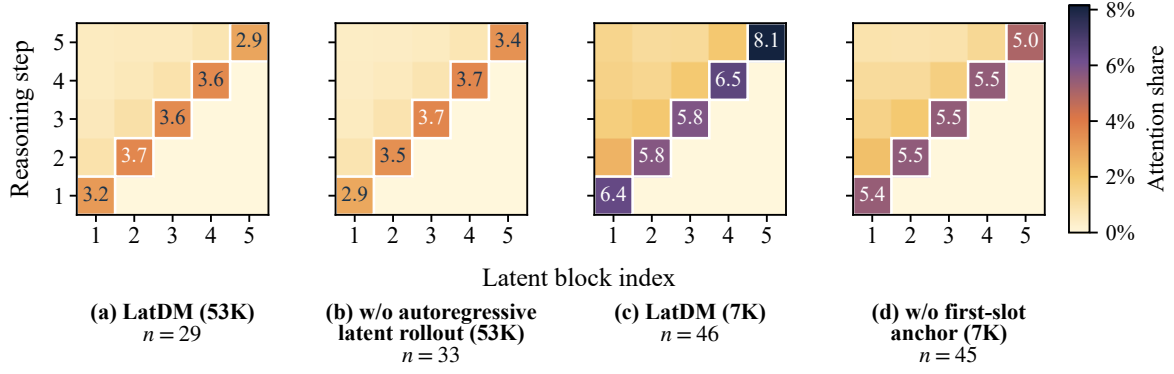


Figure 5: **Block-local attention follows the training-time temporal alignment.** Panels (a)–(d) show LatDM trained on 53K examples, the matched 53K control without autoregressive latent rollout, LatDM trained on the 7K subset, and the 7K control without the first-slot training anchor, respectively. For each multi-block sample, we record the average attention from reasoning-text tokens emitted after latent block b (row) to the latent slots of block x (column). The upper triangle ($x > b$) is structurally empty due to causal masking. Within the lower triangle, the diagonal $M[b, b]$ dominates the off-diagonal by $6.4\text{--}10.3\times$ across the evaluated LatDM checkpoints ($p \ll 10^{-3}$, paired bootstrap), supporting block-local use consistent with the temporal alignment imposed during SFT (Sec. 3.3). This pattern does not imply access to frame- b pixels at inference. The 53K checkpoints attain the sharpest alignment.

Geometric separation between latent and text channels. Table 8 reports representation-geometry measurements on the dynamic subset of VLADBench ($n = 684$ multi-frame samples), without using ground-truth labels. The latent channel lives in a low-norm, low-rank compressed domain ($\|\cdot\|_2 \approx 68$, effective rank ≈ 51), while the text-reasoning channel spans a larger-amplitude space ($\|\cdot\|_2 \approx 215$, effective rank ≈ 62). Alignment with the frozen visual-encoder subspace is slightly positive for latent positions ($+0.024$) and near zero for text-reasoning positions (-0.010 ± 0.011). Within the same sample, the two channels have low cosine similarity ($\cos = 0.16$). Finally, latent positions across blocks of the same sample are almost perfectly co-directed ($\cos = 0.99999$), indicating that inter-block differences are subtle within a low-rank region of the 4096-d latent space.

Protocol differences from the motivation probe. The motivation probe in Sec. 1 (93.2% $\rightarrow$ 72.0%, ~ 21 pp gap) was measured on the frozen backbone under the original six-camera, multi-frame protocol with permissive parsing. In contrast, the controlled diagnostic in Sec. 4.5 uses a strict single-front-camera protocol with strict binary parsing, so that the frozen and trained models are compared under the same input distribution. This explains why the frozen baseline gap in Table 4 is larger (34 pp): the single-camera strict setting amplifies the conservatism of Think-mode text decoding on the frozen backbone. In the matched diagnostic, the reduction

in the gap is driven numerically by higher text-decoded accuracy while Post-LLM hidden-state decodability remains nearly unchanged.

D.3 Slot-Level Probing of Latent Blocks

We further analyse what each of the $K = 16$ slots inside a latent block encodes. This probe is a post-hoc diagnostic on the same LatDM checkpoint used in Sec. 4; no additional training is performed.

Setup. We collect the regression-head outputs at inference time on VLADBench (10,941 valid samples after excluding 252 entries without valid model outputs) and CODA-LM (2,123 samples), yielding 13,064 block-1 tensors of shape 16×4096 . For each slot $k \in [0, 15]$, we train an independent 5-fold logistic-regression probe (ℓ_2 , balanced class weights, standardized features) under two protocols: (i) a *task-family* one-vs-rest binary probe over 12 semantic attributes, including vulnerable-road-user, lane-related, traffic-light, and CODA-LM regional-perception categories; and (ii) a *within-task* multi-class probe predicting the ground-truth answer label inside a single task. We use 8 tasks with discrete answer vocabularies, each containing 5–6 answer classes. Three baselines control for slot aggregation: *full block* (a flattened 65,536-dimensional vector), *slot average* (a 4,096-dimensional mean vector), and *random slot* (one slot sampled per example).

Measure	latent (<LP>)	text-reasoning
mean $\ h\ _2$	67.98 $\pm$ 0.02	215.40 $\pm$ 3.44
effective rank (PCA-ent.)	51.40 $\pm$ 17.78	61.73 $\pm$ 21.16
$\cos(\cdot, \text{frozen ViT})$	+0.0238 $\pm$ 0.003	-0.0095 $\pm$ 0.011
$\cos(\text{latent}, \text{text})$ (in-sample)	0.1614 $\pm$ 0.0226	
$\cos(\text{block-1 latent}, \text{blocks 2..N latent})$	0.99999 $\pm$ 6×10^{-7}	

Table 8: Representational geometry of the latent and text channels on dynamic VLADBench samples. The channels differ in norm and effective rank; latent positions show slight positive alignment with the frozen visual encoder, while text-reasoning positions are near zero.

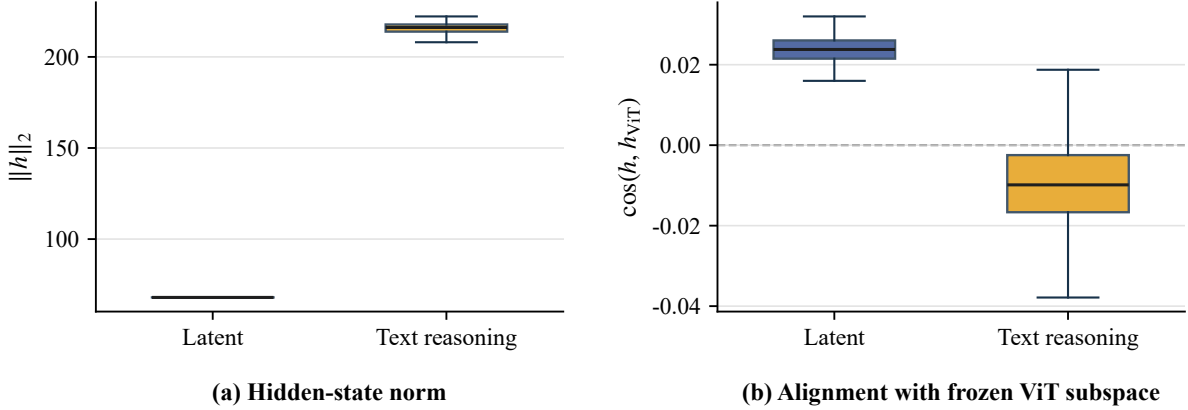


Figure 6: Per-sample hidden-state norm and cosine alignment with the frozen visual encoder on dynamic VLAD-Bench samples. Latent positions form a low-norm region with slight positive alignment to the frozen ViT subspace, while text-reasoning positions span a larger-amplitude region with near-zero alignment.

Task identity is redundantly decodable across all slots. Every one of the $16 \times 12 = 192$ task-family probes reaches $\geq 99\%$ accuracy (Fig. 7a). The three slot-aggregation baselines also yield nearly identical accuracy: *random slot* $\approx$ *slot average* $\approx$ *full block* $\approx 100\%$. Therefore, task identity is linearly recoverable from any individual slot. After subtracting each row’s mean (Fig. 7b), the residual per-slot deviations are below 1 pp, suggesting that task identity is redundantly replicated across the block. This is consistent with the low-rank geometry reported above: task conditioning is broadly broadcast along the latent manifold rather than isolated in a single slot.

Slots specialize on within-task visual content. The picture changes once task identity is held fixed. Within-task probes reach accuracies far above the majority-class floor, and slots are no longer interchangeable. The gap between the best and mean slot ranges from +1.7 pp on Lane Recognition to +26.7 pp on VRU Recognition (Table 9). Fig. 7c visualizes the full 16×8 accuracy matrix.

Three patterns emerge. First, **slot 0** yields the highest probe accuracy for five of the eight tasks, all

Task	n	maj.	max	top-1	$\bar{a}$	Δ
VRU Recognition	255	.322	.718	0	.450	26.7
Vehicle Status	194	.361	.912	0	.786	12.6
Traffic Light	658	.444	.799	0	.728	7.1
Right of Way	309	.757	.825	0	.755	7.0
Long/Short Parking	247	.352	.886	0	.839	4.8
Driving Efficiency	182	.341	.660	15	.621	3.8
Pavement Marking	270	.485	.978	2	.945	3.3
Lane Recognition	704	.398	.797	14	.780	1.7

Table 9: Within-task slot probing summary. *maj.*: majority-class accuracy; *max*: accuracy of the best slot; $\bar{a}$: mean across 16 slots; Δ : max–mean (pp).

of which emphasize direct visual recognition: VRU recognition, vehicle status, traffic-light recognition, right-of-way assessment, and long/short parking. This matches the training design in Sec. 3.4: slot 0 is most directly tied to the ground-truth latent during the autoregressive rollout, so the model naturally treats it as the block’s visual summary. Second, slots 2, 14, and 15 yield the highest probe accuracy for geometric and rule-based tasks, including pavement-marking recognition, lane recognition, and driving efficiency. This pattern is consistent

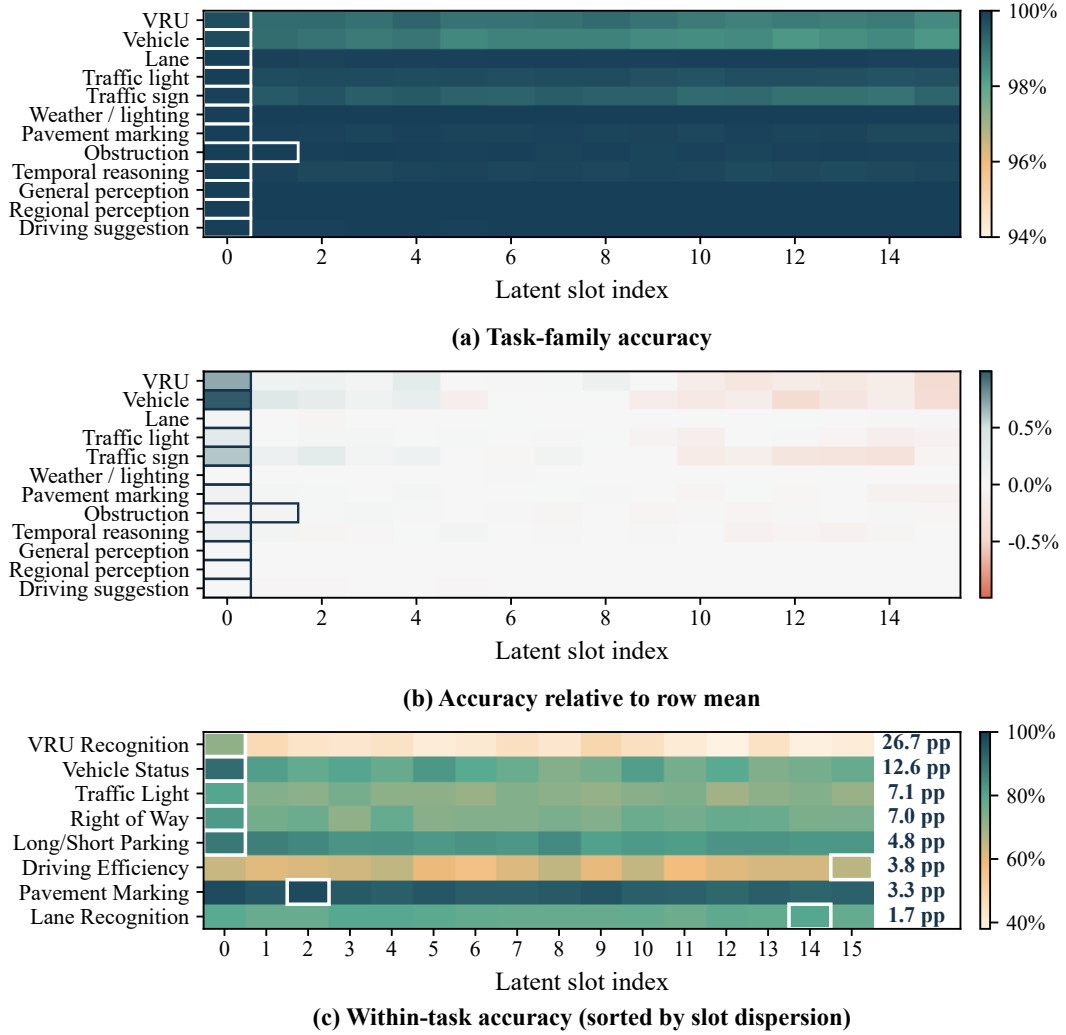


Figure 7: **Slot-level probing of latent blocks.** (a) Task-family one-vs-rest probe accuracy across 16 slots $\times$ 12 attributes; every cell reaches $\geq 99\%$. (b) The same matrix after subtracting each row’s mean: per-slot deviations are below 1 pp, indicating that task identity is redundantly broadcast across slots. (c) Within-task multi-class probe accuracy across 8 VLADBench tasks $\times$ 16 slots, with rows ordered by $\Delta_{\max-\text{mean}}$. White outlines mark the per-task top-1 slot; right-margin annotations report Δ . Slot 0 yields the highest probe accuracy on direct visual-recognition tasks, whereas slots (2, 14, 15) do so on geometry- and rule-based tasks.

with the within-block autoregressive rollout capturing more abstract reasoning content in some slots. Third, **slot 12** yields the lowest probe accuracy on three independent tasks, which suggests that the latent block contains a low-rank specialization pattern rather than a fully uniform 16-channel code.

Implication for the main narrative. Together, the two probes reveal a two-layer encoding inside each latent block. Task identity is broadcast redundantly across all 16 slots, so any single slot suffices to recover which task the assistant is answering. In contrast, the visual content needed to answer that task is distributed non-uniformly across slots, with a tendency for the anchor slot to favor direct visual recognition, while other slots are stronger for some

rule- and geometry-driven tasks. The latent block is therefore not a 16-fold copy of the same vector; it carries structured visual evidence that the text channel does not have to encode by itself, supporting the “latent-for-evidence, text-for-communication” decomposition introduced in Sec. 3.1.