How Should VLMs Represent Ego-State? A Diagnostic Study of Task Accuracy and Context Cost

Anonymous authors
Paper under double-blind review

Abstract

Vision-language models increasingly receive numerical state, such as vehicle kinematics, alongside images, yet the trade-off between how accurately and how compactly this state can be delivered is rarely measured. We measure it for four interfaces that receive the same ego-state history in Qwen3-VL-2B: decimal text, LLMTime-style text, xVal-style scalar embeddings and learned sixteen-position adapters. All share downstream supervision, train on CARLA and are evaluated on held-out CARLA and real nuScenes clips. Decimal text is the most accurate at 96.00%. Our staged adapter, pretrained on reconstruction and captions before task fine-tuning, reaches 88.57% (three-run mean 89.30%) while shortening the state segment $48.8\times$ and the full input $2.16\times$, and joint continuation raises it to 89.95%. Beyond aggregate accuracy, the two sixteen-position recipes fail on different question types, and three staged runs within 1.63 points of each other reach recall from 8.0% to 98.3% on 175 nearly stationary clips, which explain most of the difference between the weakest and strongest run. State interfaces should therefore be chosen by accuracy and context cost and evaluated per task across runs.

1 Introduction

Vision-language models (VLMs) in autonomous systems increasingly receive numerical state alongside images, including vehicle kinematics and robot proprioception (Yurtsever et al., 2020). Consider a car waiting at a traffic light for three seconds. Its speed history is a row of near-zeros, so whether the speed peaks in the first or the second half has a clear answer: there is no peak. Answering it means mapping small numbers precisely onto a concept, not only recognizing the scene.

There are several ways to hand such a history to a VLM. Decimal and digit-based serialization reuse the language model's numerical vocabulary (Gruver et al., 2023). Scalar embeddings encode magnitudes directly (Golkar et al., 2024), and learned state adapters map state vectors into compact sequences of language-model embeddings (Driess et al., 2023, Tencent Robotics X et al., 2026). Adapters are attractive because a history spanning hundreds of text positions fits in sixteen embeddings. Whether this compactness costs accuracy after task adaptation, however, is rarely measured against text on the same history, and control-success comparisons (Zhao et al., 2026) leave it open.

We measure this trade-off on EgoDyn-Bench (Schäfer et al., 2026), whose fourteen question types connect vehicle kinematic histories to semantic motion concepts. We chose this setting for three reasons. First, every label is a deterministic function of the state, so errors reflect how state is mapped to concepts, not label noise. Second, all interfaces receive the same sixteen-step history, backbone and downstream supervision, so differences stem from the interface and its training recipe. Third, training on simulated driving and testing on held-out simulated and real clips shows which interfaces transfer. The task is not easy for VLMs: zero-shot models including Gemini 3 Pro and GPT-5.1, given the history as text, score at most 75.8% (Appendix L).

Our measurements map the trade-off and reveal what aggregate scores hide. Trained decimal text reaches 96.00%, and a staged adapter, pretrained before LoRA fine-tuning, reaches 88.57% with a $48.8\times$ shorter state segment, a $2.16\times$ shorter input and $1.54\times$ faster prefill (Figure 1, Appendix F). Joint continuation of the same adapter reaches 89.95%, and a state-only MLP reaches 90.77%,

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.

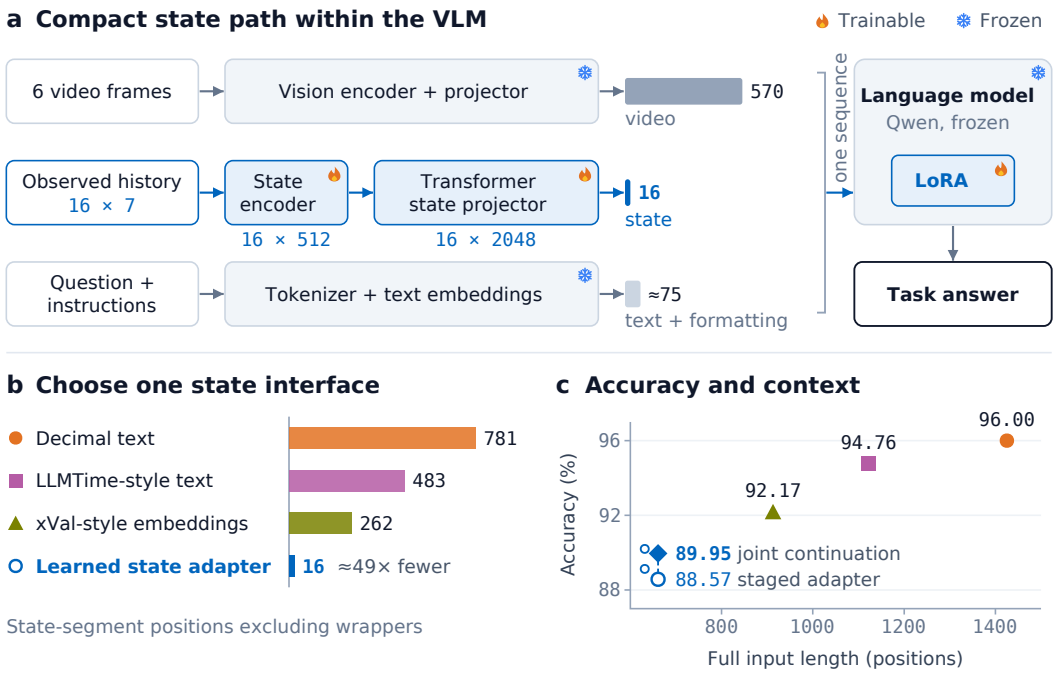


Figure 1: State interfaces and observed operating points. (a) The staged adapter within the VLM, with schematic ordering. Flames mark modules trained in at least one stage (Figure 2 gives the schedule). (b) Alternative interfaces for the same history. (c) Selected-checkpoint benchmark accuracies. Small circles mark two further staged-adapter seeds, offset horizontally for visibility at unchanged input length. State-segment means count positions between the `<trajectory>` tags (18,679 arrays). Full-input means include video/instructions (49,616 validation prompts).

so the supplied values support better state-to-concept mapping than either achieves. Below the aggregate, the two sixteen-position recipes fail on different question types, and three staged runs whose aggregate accuracy differs by only 1.63 points reach recall from 8.0% to 98.3% on 175 nearly stationary clips, as the one introduced. Our contributions are:

- **A controlled comparison of state interfaces.** We deliver one ego-state history to the same VLM as decimal text, LLMTime-style text, xVal-style embeddings and learned adapters under shared supervision, with controls and state-only baselines, and map the resulting trade-off between accuracy and context cost.
- **A compact staged state adapter.** Our pathway gives each timestep one language-model position and is prepared by reconstruction and by alignment on captions generated from motion facts, delivering the history in $48.8\times$ fewer positions than text.
- **Diagnostics that expose what aggregates hide.** Question-, class- and subgroup-level analyses, input interventions and a second backbone show where each interface fails, and that learned interfaces need per-task evaluation across training runs.

2 Related Work

Numerical interfaces for language models. LLMTime (Gruver et al., 2023) rescales each series, fixes its precision, and adapts digit separation to the backbone tokenizer, making pretrained LLMs competitive zero-shot forecasters. xVal (Golkar et al., 2024) replaces numerical strings with continuously scaled embeddings, and Time-LLM (Jin et al., 2024) maps temporal patches through text prototypes into a frozen language model. NuTime (Lin et al., 2024) makes magnitude explicit per window. CADE (Wu et al., 2026) combines direct timestep embedding with contrastive alignment for time-series question answering and outperforms byte-pair-encoded numbers.

Modality alignment and state conditioning. PaLM-E (Driess et al., 2023) maps continuous state

through an MLP into language embeddings interleaved with vision and text. LLaVA (Liu et al., 2023) aligns an image projector on captions before instruction tuning. Other connectors (Alayrac et al., 2022, Li et al., 2023) use cross-attention or learned queries. Zhao et al. (2026) compare five proprioceptive interfaces, from text prompts to action-side conditioning, for closed-loop manipulation. Their text prompt carries only the current state, whereas all our interfaces share a sixteen-step history. ThinkProprio (Wang et al., 2026a) reports higher performance on a language-conditioned manipulation benchmark with discretized VLM-vocabulary state tokens than with an MLP projector. DynaDreamer (Wang et al., 2026b) conditions a driving world model on physics-informed context from vehicle state and action histories for cross-embodiment adaptation. These studies evaluate control and prediction, whereas ego-motion QA tests individual numerical distinctions.

Driving question answering and diagnostic generalization. Driving QA spans visual explanation in LingoQA (Marcu et al., 2024), camera/LiDAR questions in NuScenes-QA (Qian et al., 2024), and spatial/temporal reasoning in NuScenes-SpatialQA, DriveSpatial and STRIDE-QA (Tian et al., 2025, Vo et al., 2026, Ishihara et al., 2026). TCogMap (Cannons et al., 2026) compares raw textual ego poses with threshold-derived, temporally indexed motion summaries. Our trained interfaces instead share a numerical history. EgoDyn-Bench (Schäfer et al., 2026) targets semantic ego-kinematics, finding encoding-dependent gains from explicit trajectory text in zero-shot video QA. BeTTER (Xu et al., 2026) finds shortcuts behind high embodied-task accuracy.

Summary. None of these studies compares trained text, scalar embeddings and learned state adapters on the same multivariate ego-state history. We close this gap by holding history, backbone and questions fixed, which complements zero-shot QA and closed-loop policy evaluation.

3 METHODOLOGY

Our method follows the introduction's question: we deliver one numerical history to the same VLM through different interfaces and ask what each costs and what a compact interface needs to succeed. We define the task (§3.1), the interfaces and controls (§3.2), how we prepare our staged adapter (§3.3) and how we measure both (§3.4). Throughout, an *interface* specifies how states enter the model, a *recipe* adds architecture and optimization, and *state-to-concept mapping* means answering the state-defined questions, each mapping the numerical history to a semantic motion concept.

3.1 TASK AND EGO-STATE

Given a video clip $\mathbf{l}$, a question $\boldsymbol{q}$ and synchronized ego-state $\boldsymbol{X} \in \mathbb{R}^{T\times D}$ with T timesteps and $D = 7$ channels, where $X_{t,c}$ is channel c at timestep t, the model answers one of fourteen ego-motion question types. The channels are origin-translated position (x, y) [m], velocity (v_x, v_y) [m/s], heading relative to initial yaw ψ [rad], signed longitudinal acceleration a [m/s^2] and jerk j [m/s^3]. Endpoint-aligned stride-two sampling of the three-second, 31-state history gives $T = 16$ states at effective 5 Hz, and every primary interface receives this 16×7 grid (Appendix B). Contrastive sequence, for example, asks "Comparing the first and second halves of the sequence, which half has more dynamic driving?" with options `first_half`, `second_half` and `similar`. When the waiting car of the introduction pulls away at green, the answer is `second_half`.

The primary backbone is Qwen3-VL-2B-Instruct (Bai et al., 2025), with six RGB frames per clip. Its vision tower, visual projector and base language model stay frozen, and all trained VLM recipes share one LoRA configuration (Hu et al., 2021).

3.2 STATE INTERFACES

All interfaces receive the same grid and differ only in how it enters the language model. Let E_0 be the frozen token embedding table of width $d_{\mathrm{LLM}} = 2{,}048$ and S_r the rounded serialization for text recipe r. The interfaces supply

$$\begin{aligned} \boldsymbol{Z}_r(\boldsymbol{X}) &= E_0\big(\mathrm{Tok}(S_r(\boldsymbol{X}))\big), && r \in \{\mathrm{decimal}, \mathrm{LLMTime}\}, \\ \boldsymbol{z}^{\mathrm{xVal}}_{\nu(t,c)} &= (X_{t,c}/s_c)\,\boldsymbol{e} && \in \mathbb{R}^{d_{\mathrm{LLM}}}, \quad t \le T,\ c \le D, \\ \boldsymbol{Z}_{\mathrm{adapter}}(\boldsymbol{X}) &= g_{\boldsymbol{\omega}}\big(f_{\boldsymbol{\theta}}(\boldsymbol{X})\big) && \in \mathbb{R}^{K\times d_{\mathrm{LLM}}}, \quad K = T = 16. \end{aligned} \tag{1}$$

Here s_c is xVal's fixed train-fitted divisor, $\boldsymbol{e}$ its shared learned vector and $\nu(t, c)$ the position of $X_{t,c}$ among the embedded formatting tokens. The adapter's state encoder $f_{\boldsymbol{\theta}}$ and state projector $g_{\boldsymbol{\omega}}$, with

parameters $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$, are described below. The context combines video, question, options, state and instructions, and all recipes share frozen backbone weights $\boldsymbol{\phi}_0$.

Interfaces adapted from prior work. Decimal text serializes the grid as rounded values inside the adapter's prompt, and LLMTime-style text adapts Gruver et al. (2023)'s formatting to the same grid under our training protocol. Both round positions and velocities to 0.1, heading to 0.001 and acceleration and jerk to 0.01 in the units above, whereas xVal and the adapters receive unrounded values (Appendix M.1). The xVal-style interface (Golkar et al., 2024) replaces each number with a magnitude-scaled copy of one learned 2,048-dimensional embedding, with fixed train-set channel scaling and language-token supervision instead of xVal's numeric regression head. CADE (Wu et al., 2026) embeds each timestep directly and projects it pointwise into the language model, trained end-to-end with an auxiliary contrastive loss. The CADE-style adapter keeps this pointwise design, a linear $7 \to 512$ encoder and a per-position MLP projector to sixteen positions, trained end-to-end with LoRA. Our multivariate adaptation omits the contrastive loss, per-series standardization and appended statistics, and the original work also reports a variant without the contrastive loss (Appendix B.3). These interfaces show how far existing designs go. They leave open whether a pathway built for numerical state and prepared before the task can close the gap to text.

Our staged adapter. We design the adapter around one principle: each of its sixteen positions carries one timestep, so the language model receives the history in temporal order at one position per timestep instead of about 49 (Figure 1a). The encoder $f_{\boldsymbol{\theta}}$ maps the grid to 512-dimensional features per timestep. The projector $g_{\boldsymbol{\omega}}$ maps them to width 2,048 through two eight-head temporal self-attention blocks and a final LayerNorm, without pooling. We compare four encoder families: Transformer, 1D convolutional (Conv1D), a per-timestep MLP, and an xLSTM-inspired bidirectional recurrent network whose cell adopts the exponential input and forget gates of sLSTM (Beck et al., 2024), normalized by their per-step sum, without its normalizer state, stabilizer or matrix memory (Appendix A.1). It thus differs from the CADE-style adapter in preparation, temporal mixing and capacity.

Controls that make the scores interpretable. An accuracy alone does not reveal whether an interface uses the state, or how much its learned pathway contributes. Two controls separate these effects while keeping sixteen state positions. The no-state control is prepared and trained like the staged adapter but with the grid zeroed in training and evaluation, so it measures what video, question priors and a learned constant state block achieve without state. The frozen-random control receives the state through an untrained, frozen encoder and projector, so its gap to the staged adapter estimates what pathway learning adds. State-only baselines without the VLM, a random forest and an MLP, show how well the concepts can be predicted from the numbers alone.

3.3 Preparing the Staged Adapter

The interfaces above meet the task directly. Our staged adapter is prepared first, so that the language model receives state positions it can already interpret (Figure 2). We follow the alignment idea of LLaVA (Liu et al., 2023), which aligns a projector on captions before instruction tuning, and precede it with a reconstruction stage for numerical state.

Stage 1: preserving the state. The encoder learns to reconstruct scaled targets from raw inputs through a temporary head, $\widehat{\boldsymbol{Y}}_i = r_{\boldsymbol{\eta}}(f_{\boldsymbol{\theta}}(\boldsymbol{X}_i))$, without the language model. **Stage 2: meeting the concepts in language.** The projector then learns to express the encoded state in the language model's embedding space. For each training clip we generate three captions, holistic, moderate and detailed, from its deterministic motion facts, and align the projector by caption likelihood through the frozen language model without video. This is where numerical state first meets the motion concepts that the benchmark asks about. The objectives are

$$\begin{aligned} \mathcal{L}_{\mathrm{S1}}(\boldsymbol{\theta}, \boldsymbol{\eta}) &= \frac{1}{|\mathcal{B}|TD} \sum_{i \in \mathcal{B}, t, c} h_1\left(\widehat{Y}_{i,t,c} - \frac{X_{i,t,c}}{\max(\widehat{\sigma}_c, 0.5)} \right), \\ \mathcal{L}_{\mathrm{S2}}(\boldsymbol{\omega}) &= -\frac{\sum_{i \in \mathcal{B}} \sum_{u=1}^{L_i} \log p_{\boldsymbol{\phi}_0}(c_{i,u} \mid \boldsymbol{c}_{i,<u}, \boldsymbol{C}_i^{\mathrm{cap}})}{\sum_{i \in \mathcal{B}} L_i}. \end{aligned} \tag{2}$$

Here h_1 is Huber loss with threshold one, $\widehat{\sigma}_c$ a fixed training-set channel standard deviation and $\mathcal{B}$ a minibatch. Stage 1 updates only $\{\boldsymbol{\theta}, \boldsymbol{\eta}\}$. The projector receives no reconstruction gradient, and the

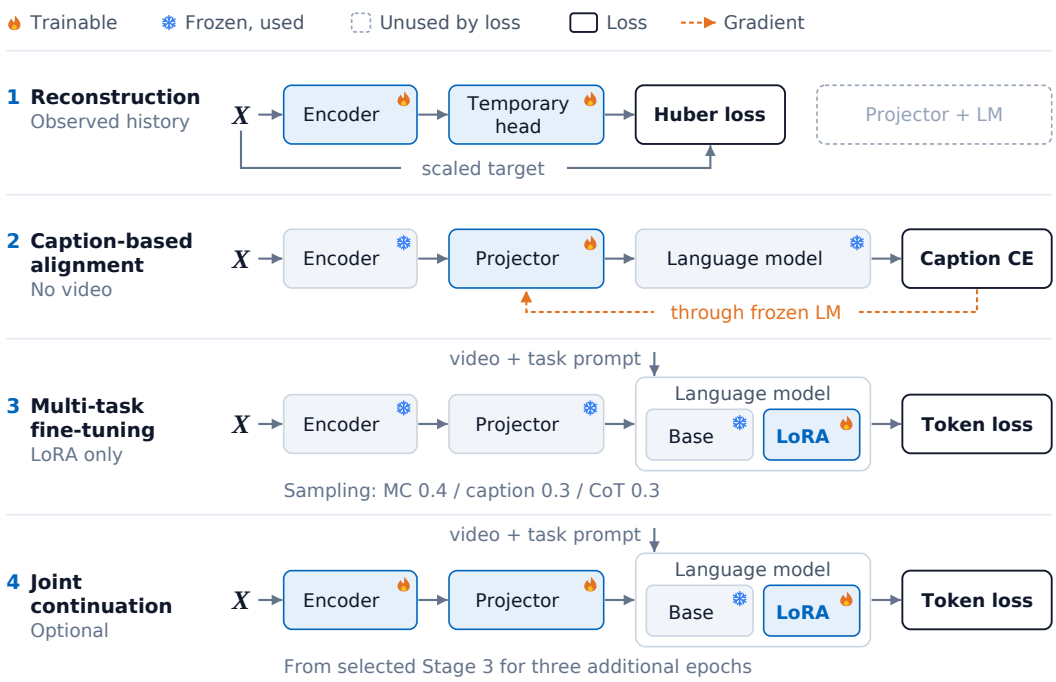


Figure 2: Qwen training. Stage 1 uses raw inputs, channel-scaled targets and encoder-specific temporary heads. The projector is forwarded without reconstruction gradients and the LM is skipped. Stage 2 backpropagates through the frozen LM. Vision remains frozen in Stages 3 and 4. Stage 4 continues selected Stage 3.

temporary head is discarded (Appendix A.1). Stage 2 updates only ω, with the selected encoder $\bar{\theta}$ and the language model frozen. Its context C_i^{cap} contains the caption instruction and $g_\omega(f_{\bar{\theta}}(X_i))$, and L_i counts supervised caption tokens including the terminal token.

Stage 3: task adaptation, shared by all recipes. Only then does the model meet the task. Every recipe samples multiple-choice, caption and chain-of-thought records with probabilities 0.4, 0.3 and 0.3 and minimizes the same token-level negative log-likelihood, length-weighted over each gradient-accumulation window (Appendix C). Stage 3 updates only LoRA for the staged adapter, the controls and the text recipes. xVal also trains its scalar embedding, and the CADE-style adapter trains its randomly initialized pathway. **Optional Stage 4: joint continuation.** We resume the selected Stage-3 checkpoint for three epochs, jointly updating encoder, projector and LoRA with the same objective, while base language and vision weights remain frozen (Appendix A).

3.4 Evaluation and Resource Protocol

Finally, every recipe is measured the same way. **Data and selection.** Training uses 14,090 synthetic CARLA clips and 3,544 scene-disjoint validation clips. Shared rationales include decision thresholds, whose interaction with the interface is examined in Appendix K. In-domain development accuracy uses 1,200 validation clips from 177 scenes, with all fourteen questions ($n = 16{,}800$). EgoDyn-Bench contains 1,000 held-out clips, half CARLA and half real nuScenes ($n = 14{,}000$).

Qwen checkpoints are selected through repeated draws of 1,400 records from the full 49,616-record validation pool, including the reported development set (Appendix A). Stage 4 selects among three epochs on the full reported development set. We train the staged adapter in three complete runs, which differ in seed and preparation length, to measure variation across training runs. Seed 42 has the highest development accuracy and remains the reference without benchmark-based reselection, although it scores lowest on the benchmark.

Accuracy and diagnostics. Table ranks are descriptive, not significance tests. We select the option with highest summed teacher-forced token log-probability, including its terminal token, and report

the fraction correct. Main benchmark contrasts use 95% percentile intervals from 10,000 paired scene-level bootstrap draws that resample driving scenes with all their clips, stratified by source with fixed 50:50 weights. They quantify saved-checkpoint uncertainty, excluding training variability and selection effects. Input interventions use an earlier staged-adapter checkpoint (development accuracy 93.82%) and their own bootstrap protocols (Appendix I).

Resources. State-segment length counts positions between the `<trajectory>` tags, whereas state-block length also counts this wrapper of tags and adjacent formatting. Full inputs add video and instructions. Lengths average over trajectories or 49,616 validation questions (Appendix F).

4 RESULTS: ACCURACY, CONTEXT AND CONCEPT MAPPING

Our results follow one thread. Accuracy falls as the state is compressed (§4.1), compression saves context and computation (§4.2), recipe choices set how much accuracy the adapter gives up (§4.3), and the remaining errors concentrate on identifiable questions and clips (§4.4). Unless stated otherwise, results use *Qwen3-VL-2B* and the seed-42 staged adapter (three-run mean 89.30%). Comparisons rank complete recipes (Section 3), and intervals exclude retraining variability.

4.1 ACCURACY FALLS AS THE STATE IS COMPRESSED

Written numbers remain the most reliable way to hand state to the language model. Decimal text reaches 96.00% on the benchmark, 7.43 points above the staged adapter's 88.57% (paired 95% interval $[6.84, 8.03]$, Table 1). The lead is broad: text is ahead on all fourteen question types and above every adapter seed, the best of which reaches 90.20%.

Table 1: Accuracy ($\%$, $\uparrow$): 16,800 CARLA development and 14,000 held-out *EgoDyn-Bench* questions, 7,000 per source with equal CARLA/nuScenes weight. Groups: text and scalar interfaces adapted from prior work, our staged adapter with the CADE-style adaptation, controls, and state-only baselines without the VLM. The staged recipe uses reconstruction and caption alignment, then trains LoRA with the pathway frozen (Stages 1 to 3). Joint continuation updates encoder, projector and LoRA from the selected Stage-3 checkpoint for three additional epochs (Stage 4). Bold/underlined: best/second per column across all eleven rows, ranked descriptively.

Recipe	Development	Benchmark	CARLA	nuScenes
Comparison interfaces				
Decimal text	**97.40**	**96.00**	**96.14**	**95.86**
LLMTime-style text	<u>97.00</u>	<u>94.76</u>	<u>93.96</u>	<u>95.57</u>
xVal-style scalar embeddings	96.36	92.17	93.61	90.73
Sixteen-position recipes				
Staged adapter (ours)	94.39	88.57	88.69	88.46
+ Joint continuation (ours)	94.83	89.95	89.71	90.19
CADE-style adapter	93.57	84.84	84.71	84.96
Controls				
Frozen random pathway	92.15	81.36	81.94	80.77
No-state (zeroed grid)	78.82	55.10	55.31	54.89
State-only baselines				
Random forest (300 trees per question type)	95.27	90.06	91.13	88.99
MLP (per question type)	94.93	89.91	90.70	89.11
MLP, single model (state + question type)	95.21	90.77	91.40	90.14

Between these two ends, accuracy falls step by step with the length of the state segment. LLMTime-style text writes the same values in 483 positions and reaches 94.76%, 1.24 points behind decimal text's 781 positions (interval $[0.82, 1.65]$, Table 10). xVal keeps one scaled embedding per value, 262 positions with formatting, and reaches 92.17%. The staged adapter's sixteen positions give 88.57%. Across these four interfaces, accuracy rises with the number of positions the history occupies.

Text also transfers more evenly. Its advantage grows from 3.02 points on in-domain development clips to 7.43 on the held-out benchmark, and per question type the adapter's accuracy differs between CARLA and real nuScenes clips by 7.57 points on average, against 2.69 for text (Appendix E).

The missing accuracy is not missing information. A state-only MLP, one network over state and question type without the VLM, reaches 90.77% (three seeds, 90.56 to 90.99%), above every staged-adapter seed. It is strongest on speed peak half (95.7%) and stop-and-go (96.3%), yet trails the staged adapter on braking intensity, driving smoothness and turn direction (Table 16). The values thus contain what the adapter misses, so its deficit lies in delivery and training.

4.2 Context cost across interfaces

Context matters because state competes with video and instructions for the same sequence and compute budget, and richer state, such as longer histories, higher rates or surrounding agents, only enlarges its share. The four interfaces span a wide range of cost (Table 2): decimal text needs 781 state-segment positions, LLMTime-style text 483 and xVal 262, whereas the staged adapter needs sixteen, $48.8\times$ fewer than decimal text. Because video alone occupies about 570 positions, full inputs shrink less, from 1,425 through 1,123 and 913 to 661 positions, and estimated forward arithmetic follows the same order, from 4,250 to 1,917 GFLOP.

Table 2: Mean input lengths (positions) and estimated forward arithmetic (↓). State segments exclude wrappers (18,679 CARLA trajectories), while blocks include them (49,616 validation questions). Full sequences include video and instructions, over the same questions. GFLOP uses one output-position vocabulary projection, excludes vision and is not multiple-choice scoring cost. Bold/underlined: lowest/second-lowest per column across four configurations.

Interface	State segment↓	State block↓	Full sequence↓	GFLOP↓
Staged adapter	**16**	**26**	**661**	**1917**
xVal-style scalar embeddings	<u>262</u>	<u>278</u>	<u>913</u>	<u>2670</u>
LLMTime-style text	483	488	1123	3311
Decimal text	781	790	1425	4250

To test these savings in practice, we time each interface on one fixed prompt shape on an RTX 5090 in bf16, with vision features cached so that only the language model and the state pathway run (Appendix F). Measured costs keep the same order. Prefill takes 56.6 ms for decimal text, 39.1 ms for xVal and 36.8 ms for the adapter, and the largest prefill batch that fits in memory grows from 42 to 63 to 89, while 64-token generation time changes little. The adapter is therefore the most compact interface and the fastest in prefill, for 104.9M additional parameters.

4.3 What sets the adapter's price

If the information is present and sixteen positions are cheap, what limits the adapter? Learning the pathway matters first: at the same budget, the staged adapter exceeds the frozen random pathway by 7.21 benchmark points and the CADE-style adapter by 3.74 (Table 1).

The two adapters capture different parts of the state. The CADE-style adapter is stronger on speed and braking questions: speed peak half (95.5% versus 79.5%), stop-and-go (90.0% versus 81.6%) and braking intensity (93.4% versus 87.0%). The staged adapter is stronger on heading and lateral motion: heading change (90.6% versus 62.6%), turn direction (85.5% versus 57.3%) and motion axis (88.3% versus 78.1%). Neither exhausts what sixteen positions can carry (Figure 3a).

Unfreezing the pathway recovers the weak questions. Joint continuation (Stage 4) raises benchmark accuracy to 89.95% (gain 1.38 points, interval $[0.96, 1.80]$) and narrows the deficit to text to 6.05 points (interval $[5.47, 6.65]$). The gain concentrates where the staged adapter was weakest: contrastive sequence rises from 70.2% to 76.8%, speed peak half from 79.5% to 82.9% and stop-and-go from 81.6% to 84.7%, together 131 of 193 net additional correct answers (Figure 3b). Unfreezing coincides with further training, but the pattern points to the frozen pathway as the limit.

Better preparation does not mean better task accuracy. Table 3 compares preparation metrics with benchmark accuracy. The xLSTM-inspired encoder reconstructs more accurately than

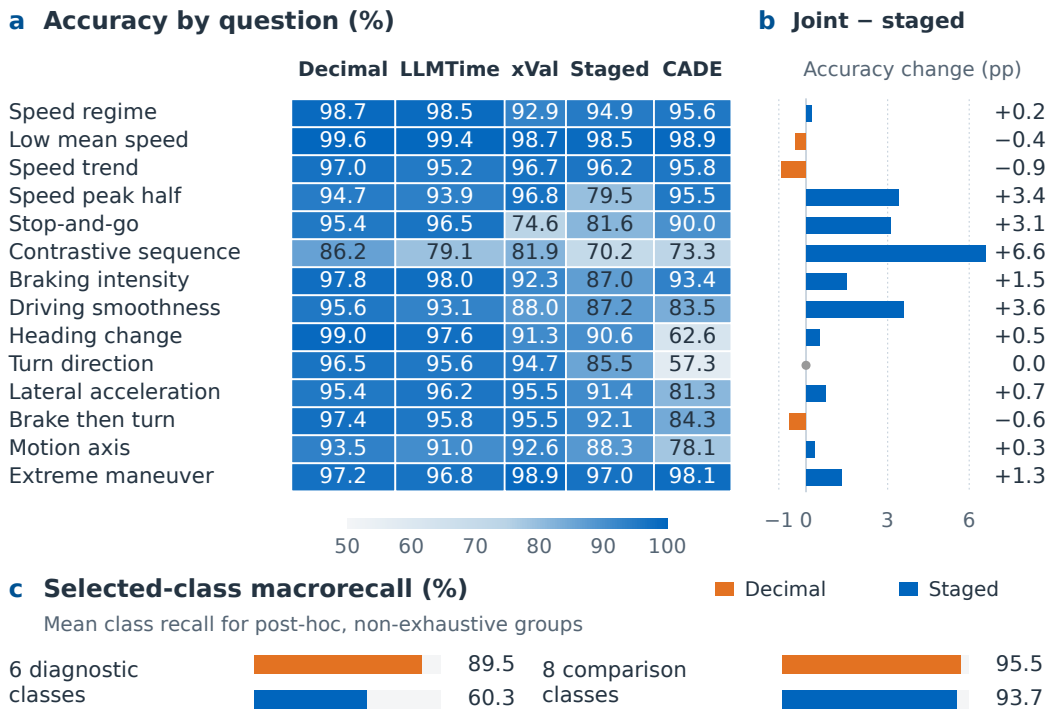


Figure 3: Benchmark diagnostics at selected checkpoints. Staged denotes seed 42. (a) Option-log-likelihood accuracy: 1,000 items per type. Columns: decimal text, LLMTime-style text, xVal-style embeddings, staged adapter and CADE-style adapter. (b) Joint continuation minus staged adapter. Contrastive sequence, speed peak half and stop-and-go contribute 131/193 net additional correct answers (67.9%). (c) Recall of decimal text and the staged adapter, averaged without weighting over six post-hoc diagnostic classes (e.g. speed peak half with answer `no_peak`) and eight comparison classes (Appendix H.2). These classes do not define a general difficulty taxonomy.

the Transformer but has lower benchmark accuracy, and Conv1D has lower caption loss than the MLP encoder but lower task accuracy. Reconstruction and caption likelihood measure whether the pathway preserves and verbalizes the state, not whether it separates the concepts that the questions require, and the encoders also differ in capacity (Appendix G). A second backbone sharpens this reading. On InternVL3.5-2B the xLSTM-inspired encoder is both better prepared and 5.55 points more accurate than the Transformer (87.44% versus 81.89%), the reverse of their order on Qwen. With encoder capacity also differing, neither preparation metrics nor the encoder ranking can be assumed to carry over between backbones (Appendix N).

Table 3: Qwen encoder recipes: Stage 1 reconstruction average displacement error (ADE), Stage 2 caption cross-entropy loss (CE), and benchmark accuracy. Bold/underlined: best/second per column across four recipes. Arrows give the preferred direction. Full metrics: Table 21.

Encoder	ADE (m)↓	Caption CE↓	Benchmark (%)↑
Transformer	0.353	**0.4395**	**88.6**
xLSTM-inspired	**0.084**	0.6086	<u>87.5</u>
MLP	<u>0.180</u>	0.5166	84.0
Conv1D	0.533	<u>0.5008</u>	80.2

4.4 Where the remaining errors are

Aggregate accuracy tells how much the adapter gives up, not where. We return to the waiting car.

A small subgroup separates training runs. The waiting car is not a thought experiment. The benchmark contains 175 clips whose correct speed-peak-half answer is `no_peak` and whose maximum speed stays below 4 m/s, 172 of them below 0.5 m/s, where the answer hinges on reading tiny speeds precisely. Interfaces that carry the exact numbers solve them, with 96.0% recall for text and 99.4% for xVal, and so does the CADE-style adapter (95.4%). The three staged runs, which differ in preparation as well as seed, do not agree: they label 14, 94 and 172 of these clips correctly, recalls from 8.0% to 98.3%, although their aggregate accuracies differ by only 1.63 points. These clips hold 1.25% of the benchmark's questions yet explain 158 of the 228 answers that separate the weakest from the strongest run. The architecture can thus solve the waiting car, but an aggregate score does not reveal whether a given run did. The same failure appears on InternVL, where the staged adapters with Transformer and xLSTM-inspired encoders stay below 18% recall on these clips against 61.5% and 64.1% at 8 to 14 m/s, and joint continuation lifts the weakest Qwen run only to 34.3%. We selected this subgroup after inspecting errors (Appendices H and N).

The shortfall concentrates on questions that compare parts of the history. Beyond this subgroup, three question types that relate one part of the clip to another, contrastive sequence, speed peak half and stop-and-go, account for 55% of the growth in text's lead from development to benchmark, and the same concentration appears for selected answer classes (Figure 3c). The gap is not inherent to compact numbers: xVal, also compact, exceeds decimal text on speed peak half (96.8% versus 94.7%) and extreme maneuver (98.9% versus 97.2%, Appendix E).

Only some of them need temporal order. When every timestep of an earlier staged checkpoint is replaced by the clip mean, contrastive-sequence accuracy on development items falls from 83.0% to 41.8%, but speed peak half only from 93.4% to 89.2%. Speed peak half can survive through the repeated clip means and the video, whereas contrastive sequence compares the two halves and collapses (Appendix I). Label shift plays a minor role: reweighting predictions for it recovers at most 0.90 adapter points, mostly on stop-and-go, and excluding stop-and-go leaves a 6.94-point gap.

4.5 Discussion and outlook

Choosing a state interface. The interfaces trace a trade-off, with decimal text for accuracy, the staged adapter for context and xVal in between. The balance shifts as state grows. Numerical interfaces pay per value, so text adds about 49 positions per timestep and xVal about 16, while the adapter adds one. Longer horizons multiply this cost, and scenes with several traffic participants, as needed to understand or predict their motion, multiply it again by their number. At the same per-agent encoding, ten agents over three seconds would occupy about 7,800 positions as decimal text against 160 for the adapter. The savings concern conditioning an existing VLM, while the state-only MLP is a strong alternative for the fixed classification task.

Evaluating the learned interface. Rankings differ between recipes per task, and subgroup recall varies across staged runs despite similar aggregates, which motivates per-task evaluation across runs.

Scope and outlook. We study compact 2B backbones, the scale suited to onboard use, where context and latency budgets are tightest, and InternVL3.5-2B staged adapters (81.9% to 87.4%) reproduce the low-speed failure on a second architecture. Deterministic labels rule out label noise, and training in simulation while testing on real clips measures transfer directly. More training runs would show how often such subgroup failures occur.

5 Conclusion

We compared four ways of giving a vision-language model the same ego-state history. Trained decimal text was the most accurate. The staged adapter shortened the input $2.16\times$ and the state segment $48.8\times$ for 7.43 benchmark points, a gap that joint continuation narrowed to 6.05. The remaining gap was not uniform. Recipes with the same sixteen positions failed on different questions, and runs of one recipe that agreed in aggregate disagreed sharply on nearly stationary clips. We therefore believe that numerical-state interfaces should be judged by accuracy and context cost together, and evaluated per task and across training runs. Because text grows with every timestep and every traffic participant, the adapter's context advantage widens for longer horizons and multi-agent scenes, and testing whether its accuracy keeps pace there is the natural next step.

AI USE STATEMENT

Generative AI tools were used for five purposes. They polished and edited the writing, and drafted parts of sections and the interface diagram, which the authors revised. They supported retrieval and discovery, such as finding related work. They supported research ideation and execution by discussing methodological choices, suggesting analyses and revising analysis code. A language model (Qwen3-8B) also generated the wording of the Stage-2 captions from the deterministic motion facts of each clip (Section 3.3). The authors designed the study, checked all results and citations, and remain responsible for the submission.

ETHICS STATEMENT

Training uses simulated CARLA trajectories. Evaluation also uses existing real-world nuScenes clips through EgoDyn-Bench. This study does not involve new collection of participant data. The experiments evaluate state-defined question answering, and their accuracy does not establish readiness for autonomous control or safety-critical deployment. Any release must respect the underlying dataset and simulator terms.

REPRODUCIBILITY STATEMENT

Section 3 and Appendices A through C specify data processing, architectures, optimization and checkpoint selection. Appendix D defines scoring and uncertainty estimation. Anonymized code, saved predictions and a replay script are provided as supplementary material. The compact bundle covers seven recipes on 30,800 questions and replays central statistics, separately from reproducing training or inference, and its README specifies coverage and regeneration commands. Upon acceptance, we will open-source everything needed to reproduce the study: the complete training and evaluation code, trained checkpoints of all recipes and training runs, all prediction files and results, and the training data we generated, including the CARLA state histories, fact-derived captions and rationales. EgoDyn-Bench is already public, and releases of nuScenes-derived material follow the nuScenes terms.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikoł aj Bińkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: a visual language model for few-shot learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 23716–23736. Curran Associates, Inc., 2022. doi: 10.52202/068431-1723. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/960a172bc7fbf0177ccccbb411a7d800-Paper-Conference.pdf.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: extended long short-term memory. In *Proceedings of the 38th International Conference on Neural*

Information Processing Systems, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Kevin Cannons, Saeed Ranjbar Alvar, Mohammad Asiful Hossain, Ahmad Rezaei, Mohsen Gholami, Alireza Heidarikhazaei, Zhou Weimin, Yong Zhang, and Mohammad Akbari. From segments to scenes: Temporal understanding for agentic autonomous driving via vision-language models, 2026. URL https://arxiv.org/abs/2512.05277.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: an embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles Cranmer, Geraud Krawezik, Francois Lanusse, Michael McCabe, Ruben Ohana, Liam Parker, Bruno Régaldo-Saint Blancard, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. xVal: A continuous numerical tokenization for scientific language models, 2024. URL https://arxiv.org/abs/2310.02989.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. Large language models are zero-shot time series forecasters. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 19622–19635. Curran Associates, Inc., 2023. doi: 10.52202/075280-0861. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/3eb7ca52e8207697361b2c0fb3926511-Paper-Conference.pdf.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Keishi Ishihara, Kento Sasaki, Tsubasa Takahashi, Daiki Shiono, and Yu Yamaguchi. STRIDE-QA: visual question answering dataset for spatiotemporal reasoning in urban driving scenes. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'26/IAAI'26/EAAI'26. AAAI Press, 2026. ISBN 978-1-57735-906-7. doi: 10.1609/aaai.v40i7.37441. URL https://doi.org/10.1609/aaai.v40i7.37441.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models, 2024. URL https://arxiv.org/abs/2310.01728.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Chenguo Lin, Xumeng Wen, Wei Cao, Congrui Huang, Jiang Bian, Stephen Lin, and Zhirong Wu. NuTime: Numerically multi-scaled embedding for large-scale time-series pretraining, 2024. URL https://arxiv.org/abs/2310.07402.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916. Curran Associates, Inc., 2023. doi: 10.52202/075280-1516. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6dcf277ea32ce3288914faf369fe6de0-Paper-Conference.pdf.

Ana-Maria Marcu, Long Chen, Jan Hünermann, Alice Karnsund, Benoit Hanotte, Prajwal Chidananda, Saurabh Nair, Vijay Badrinarayanan, Alex Kendall, Jamie Shotton, Elahe Arani, and

Oleg Sinavski. LingoQA: Visual question answering for autonomous driving. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision – ECCV 2024*, pp. 252–269, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-72980-5.

Tianwen Qian, Jingjing Chen, Linhai Zhuo, Yang Jiao, and Yu-Gang Jiang. NuScenes-QA: a multi-modal visual question answering benchmark for autonomous driving scenario. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i5.28253. URL https://doi.org/10.1609/aaai.v38i5.28253.

Finn Rasmus Schäfer, Yuan Gao, Dingrui Wang, Thomas Stauner, Stephan Günnemann, Mattia Piccinini, Sebastian Schmidt, and Johannes Betz. EgoDyn-Bench: Evaluating ego-motion understanding in vision-centric foundation models for autonomous driving. In *Computer Vision – ECCV 2026*, pp. 688–705, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-37321-2.

Tencent Robotics X, HY Vision Team, Xumin Yu, Zuyan Liu, Ziyi Wang, He Zhang, Yongming Rao, Fangfu Liu, Yani Zhang, Ruowen Zhao, Oran Wang, Yves Liang, Haitao Lin, Minghui Wang, Yubo Dong, Kevin Cheng, Bolin Ni, Rui Huang, Han Hu, Zhengyou Zhang, Linus, and Shunyu Yao. HY-Embodied-0.5: Embodied foundation models for real-world agents, 2026. URL https://arxiv.org/abs/2604.07430.

Kexin Tian, Jingrui Mao, Yunlong Zhang, Jiwan Jiang, Yang Zhou, and Zhengzhong Tu. NuScenes-SpatialQA: A spatial understanding and reasoning benchmark for vision-language models in autonomous driving, 2025. URL https://arxiv.org/abs/2504.03164.

Hao Vo, Khoa Vo, Phu Loc Nguyen, Sieu Tran, Duc Minh Nguyen, Ngo Xuan Cuong, Gladys Gawugah, Sreevenkata Anjani Tishita Godavarthi, Chase Rainwater, Nghi D. Q. Bui, Anh Nguyen, Duy Minh Ho Nguyen, and Ngan Le. DRIVESPATIAL: A benchmark for spatiotemporal intelligence in VLMs for autonomous driving, 2026. URL https://arxiv.org/abs/2605.23176.

Fangyuan Wang, Peng Zhou, Jiaming Qi, Shipeng Lyu, Chengyang He, David Navarro-Alarcon, and Guodong Guo. Think proprioceptively: State-grounded visual token selection for VLA policies, 2026a. URL https://arxiv.org/abs/2602.06575.

Zhidong Wang, Jingsong Liang, Zirui Li, Zhan Chen, Han Yu, and Chen Lv. Ego-dynamics-augmented world model for autonomous driving with zero-shot cross-embodiment adaptation, 2026b. URL https://arxiv.org/abs/2607.13410.

Yafeng Wu, Huu Hiep Nguyen, Thin Nguyen, and Hung Le. Beyond tokenization: Direct timestep embedding and contrastive alignment for time-series question answering, 2026. URL https://arxiv.org/abs/2606.18986.

Haiweng Xu, Sipeng Zheng, Hao Luo, Wanpeng Zhang, Ziheng Xi, and Zongqing Lu. Unmasking the illusion of embodied reasoning in vision-language-action models, 2026. URL https://arxiv.org/abs/2604.18000.

Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020. doi: 10.1109/ACCESS.2020.2983149.

Yiren Zhao, Ziyang Chen, Ziyang Rao, Pengteng Li, He Zhang, Weiyu Guo, Yandong Guo, and Rushi Dai. How should vision-language-action models use proprioceptive state?, 2026. URL https://arxiv.org/abs/2608.03052.

WHERE EACH CLAIM IS SUPPORTED

Main-text claim	Evidence
Trained-text advantage and its intervals (§4.1)	Tables 10, 11
Text advantage on CARLA and nuScenes (§4.1)	Tables 12 to 14
State-only MLP at 90.77, by question (§4.1)	Appendix E.3, Tables 15 to 16
Context, arithmetic, prefill and memory (§4.2)	Appendix F, Tables 17 to 19
Encoder variants and preparation vs. accuracy (§4.3)	Appendix G, Tables 20 to 21
Staged vs. CADE-style by question (§4.3)	Table 13, Figure 3a
Low-speed recall across recipes and staged runs (§4.4)	Appendix H.2, Tables 24, 28
Class errors and label shift (§4.4)	Appendix H.2, Tables 26 to 27
Temporal and channel dependence (§4.4)	Appendix I, Tables 29 to 31, Figure 5
Threshold disclosure in supervision (§3.4)	Appendix K
Rounding and resampling sensitivity (§3.2)	Appendix M.1
Eight-position budget (supplementary)	Appendix M.2, Table 36
Second backbone (§4.3 to §4.5)	Appendix N, Tables 37 to 41
Selection and training-run variation (§3.4)	Appendices A, E
Joint continuation (§4.3)	Appendix A, Figure 3b
Zero-shot and unadapted comparisons (§1)	Appendix L, Table 35
Worked benchmark items	Appendix E.4, Figure 4

A TRAINING RUNS AND CHECKPOINT SELECTION

This section supports Section 3.4's comparison protocol by specifying completed training budgets and checkpoint selection. Input interventions (Appendix I) use an earlier checkpoint of the staged adapter (development accuracy 93.82). The primary accuracy and cost comparisons use the selected checkpoint (94.39).

Which training stages are shared. Each staged encoder family has its own reconstruction and caption-alignment stages, which downstream variants of that architecture can share. Text, frozen-random and CADE-style recipes bypass both stages. xVal also begins at Stage 3, jointly learning its 2,048-parameter scalar embedding and LoRA. Table 4 summarizes the runs. The seed-42 staged and eight-position recipes complete 61 caption-alignment epochs and select epoch 46. Seeds 43 and 44 complete 67 and 57, respectively. The eight-position reconstruction batch is 240 rather than 120, and its Stage-3 microbatch/accumulation is 2×16 rather than 4×8, both effective batch 32.

Recorded training time. Durations cover the logged training runs through their completed epochs, including validation and checkpointing within each timer's interval. Trainer timers start after model and dataset setup. W&B runtimes have different start and end boundaries. These durations therefore reflect each run's stopping point and logging boundaries. The resumed CADE-style run uses its cumulative logger runtime. Complete timers are unavailable for the recurrent downstream run, whose log ends within epoch twelve, and for eight-position preparation.

The thirteen distinct downstream runs with recovered durations sum to 1,097.4 single-GPU hours. Seven separately trained preparation chains add 47.0 hours, giving a documented subtotal of 1,144.3 RTX 5090 hours from unrounded durations. Shared preparation and the two eight-position checkpoints are counted once. This partial accounting excludes stages without recovered timers, InternVL, state-only classifiers, other diagnostics, exploratory runs, preprocessing, cache construction and scoring. The available budget supported broad interface coverage and three staged training runs. Between-run variability remains unmeasured for the single-run recipes, and checkpoint-conditioned bootstrap intervals do not estimate it.

Corpus composition. Table 5 distinguishes trajectories, indexed records and caption variants.

Three distinctions in that table matter when reading an epoch. Stage 1 uses the multiple-choice indexed trajectory loader, 197,260 training rows and 49,616 validation rows, and neither the question nor its answer enters the reconstruction objective, so each of the 14,090 unique training trajectories is encountered fourteen times per epoch. Stage 2 holds one item per trajectory and samples one of that trajectory's three caption variants uniformly on access, so an ordinary Stage 2 epoch contains 14,090 examples rather than 42,270 independent trajectory and caption pairs. The chain-of-thought

Table 4: Training runs and selected checkpoints for the Qwen comparisons. Each run uses one RTX 5090. S1 gives optimizer updates / epochs reached, where 2^* denotes stopping within epoch two. S2 is the selected caption epoch, and S3 gives completed / selected downstream epochs. Prep. h gives S1/S2 durations. Preparation times and † downstream entries use W&B runtimes, while other durations use trainer wall-clock timers. Steps cover downstream training. Continuation counts only its additional training. "None" means no preparation, "Shared" reuses the staged seed-42 preparation, and "n/a" marks an unavailable entry. The two $K = 8$ rows share one run. No rank emphasis. Updates, epochs and hours.

Recipe	S1 upd./ep.	S2	S3	Steps	Prep. h	Downstream h	Seed
Decimal text	None	n/a	11/6	92,147	None	$190.9^{\dagger}$	42
LLMTime	None	n/a	7/2	58,639	None	88.4	42
xVal	None	n/a	8/3	67,016	None	84.6	42
Staged adapter	2,478/2^*	46	14/9	117,278	0.5/4.7	113.5	42
Staged adapter	4,105/5	52	14/9	117,278	1.3/4.9	$111.1^{\dagger}$	43
Staged adapter	4,105/5	42	10/5	83,770	1.3/4.1	$78.9^{\dagger}$	44
Joint continuation	Shared	46	3/2	25,131	Shared	23.8	42
CADE-style adapter	None	n/a	9/4	75,393	None	$73.9^{\dagger}$	42
Frozen random	None	n/a	9/4	75,393	None	$79.5^{\dagger}$	42
Trained no-state	4,105/5	91	9/4	75,393	1.3/7.2	$71.5^{\dagger}$	42
$K = 8$ selected	n/a/5	46	8/3	67,016	n/a	66.3	42
$K = 8$ supplementary	n/a/5	46	8/7	67,016	n/a	66.3	42
xLSTM-inspired	4,105/5	64	11/7	n/a	1.2/6.3	n/a	42
MLP encoder	4,105/5	48	6/1	50,262	1.2/4.9	$46.2^{\dagger}$	42
Conv1D	4,105/5	78	9/4	75,393	1.3/6.7	$68.8^{\dagger}$	42

Table 5: Scene-safe training and validation datasets. Caption variants count the three alternative texts associated with each trajectory. The caption dataset itself contains one index per trajectory. The scene-safe splits contain 14,090 training and 3,544 validation trajectories. Counts.

Dataset unit	**Train**	**Validation**
Unique trajectories	14,090	3,544
Multiple-choice records	197,260	49,616
Caption records	14,090	3,544
Candidate caption variants	42,270	10,632
CoT rationale records	56,723	14,004

set contains 56,723 rationale records over 12,537 unique trajectories, 89.0 percent coverage of the training clips, of which 50,097 are per-question and 6,626 multi-question, with 14,004 validation records over 3,105 clips.

Model configuration. Both backbones keep the visual tower frozen. LoRA uses rank $r = 16$, scaling $\alpha_{\mathrm{LoRA}} = 32$ and dropout 0.05. Unless stated otherwise, the state adapter uses $T = K = 16$, encoder output width 512, and a linear input projection followed by two Transformer blocks with eight attention heads.

What the encoder comparison holds fixed. The four Qwen3-VL-2B-Instruct encoders use the three-stage procedure at seed 42. State-position budget, projector output width, downstream prompts, LoRA and evaluation are shared, while internal encoder widths and parameter counts differ.

The two backbone implementations. Qwen and InternVL differ in encoder width, gate clipping, projector normalization, reconstruction-head pooling, Stage-1 target scaling and selection sampling. Appendix N gives InternVL adapter results and preparation metrics. The trained interface comparison is Qwen-specific.

Checkpoint selection in full. Qwen draws a new uniform 1,400-record sample from the complete 49,616-record validation pool at each validation call. This pool includes all 16,800 reported de-

Table 6: Qwen optimization settings. Batch is microbatch size times gradient accumulation. The Stage-1 row specifies the staged Transformer, with encoder exceptions described in the text. Learning rates are dimensionless, warmup counts optimizer updates, and caps count epochs.

Stage	Updated parameters	Peak learning rate	Batch	Warmup	Cap
1	Encoder, reconstruction head	10^{-3}	30×4	200	5
2	Projector	10^{-4}	8×1	500	100
3	LoRA, also scalar embedding for xVal	2×10^{-4}	32 eff.	200	20
3 (CADE)	LoRA / encoder and projector	2×10^{-4} / 2.5×10^{-5}	4×8	200	20
4	LoRA / encoder and projector	5×10^{-5} / 2.5×10^{-5}	4×8	200	3

velopment questions. Draws are neither clip-complete nor question-stratified, and their IDs and RNG states were not saved, so exact overlap cannot be reconstructed. Development accuracy is not selection-disjoint. Training, validation and benchmark remain disjoint by clip and relevant parent scene. Stage 3 uses a twenty-epoch cap, patience five and minimum improvement 0.005.

Stage 4 selects among three full-development evaluations: 93.8452, 94.8274 and 94.7560 percent. Epoch 2 is selected before benchmark evaluation (89.95 percent), with no benchmark-based reselection. InternVL uses a fixed seed-specific 200-question selection subset without question stratification. Its preceding configuration used 1,400 questions. Only completed epoch checkpoints are eligible. Intermediate, caption and CoT validation metrics are diagnostic, while final accuracy uses the fixed complete evaluation lists.

A.1 Optimization and implementation settings

This subsection supports Section 3.3 by specifying optimization and state-module implementations (Tables 6 and 7).

Shared optimization settings and exceptions. All stages use AdamW with $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01 and gradient-norm clipping at 1.0. Learning rates warm up linearly and then follow cosine decay towards 0.1 times their peak. The schedule spans the configured budget, so early stopping need not reach that minimum. The Stage-1 Transformer run uses effective batch $30 \times 4 = 120$. MLP, Conv1D and xLSTM-inspired use $30 \times 8 = 240$. Stage 2 uses batch eight without accumulation. Stage 3 uses $4 \times 8 = 32$ for the sixteen-position state adapters, frozen-random and no-state controls, and $2 \times 16 = 32$ for decimal, LLMTime and xVal. xVal trains its scalar embedding at the same learning rate as LoRA. The CADE-style Stage-3 recipe trains its encoder and projector end-to-end at 2.5×10^{-5}, alongside LoRA at 2×10^{-4}, using batch $4 \times 8 = 32$. Stage 4 uses separate LoRA and pathway parameter groups.

Stage 1 has a five-epoch cap and monitors a reconstruction-loss EMA with coefficient 0.05: a 0.5% relative improvement resets patience of 500 updates, with stopping disabled before update 1,000. The final Stage-1 encoder is frozen for Stage 2, which has a 100-epoch cap, patience 15 and minimum absolute validation-loss improvement 0.001. Completed budgets and selected epochs are summarized in Table 4.

State inputs and modules. The encoder represents the observed clip history. Positions are translated to the first sample without rotating the coordinate axes. Velocity components use the source yaw, while heading is relative to its initial value (Appendix B). The MLP applies the same pointwise mapping at each timestep, with learned positional embeddings. Conv1D uses three undilated, symmetrically padded temporal convolutions without residual blocks. It does not implement a causal, dilated TCN. All four encoder families feed the shared Transformer-based state projector, whose self-attention mixes timesteps. The CADE-style pathway instead uses a pointwise MLP projector.

Reconstruction and LoRA modules. The temporary reconstruction head is $512 \to 256 \to d_{\text{out}}$, with GELU and float32 LayerNorm between its two linear layers and a zero-initialized output layer.

Table 7: State modules for both backbones: sixteen width-512 encoder outputs and a width-2,048 projector. Encoder capacity and normalization differ. Each Transformer allocates a $1{,}000 \times 512$ position table but uses sixteen rows, leaving 503,808 parameters inactive. Counts are millions of parameters and dimensions are features.

Module	Params	Architecture
Qwen3-VL-2B *four encoder families, one shared projector*		
Transformer	3.15M	Input $7 \to 512$, learned positions, two post-normalized blocks, eight heads, feed-forward width 256, ReLU, dropout 0.1, and output LayerNorms.
MLP	0.14M	Per-timestep $7 \to 256$, learned positions, ReLU and LayerNorm, then $256 \to 512$ and output LayerNorm.
Conv1D	0.048M	Channels $7 \to 16 \to 32 \to 64$, kernel 5, stride 1, padding 2, ReLU and BatchNorm per layer, then per-timestep $64 \to 512$ and output LayerNorm.
xLSTM-inspired	2.90M	Input $7 \to 256$, two bidirectional recurrent layers, hidden width 256 per direction, dropout 0.1 between layers, then $512 \to 512$ output projection and LayerNorms.
InternVL3.5-2B *same interface widths, different encoder and normalization details*		
Transformer	3.67M	Input $7 \to 512$, learned positions, two post-normalized blocks, eight heads, feed-forward width 512, GELU, dropout 0.1, and one float32 output LayerNorm.
MLP	0.28M	Per-timestep $7 \to 512$, learned positions, then $512 \to 512$ and output LayerNorm.
Conv1D	0.048M	Channels $7 \to 16 \to 32 \to 64$, kernel 5, stride 1, padding 2, then per-timestep $64 \to 512$ and output LayerNorm.
xLSTM-inspired	11.03M	Input $7 \to 512$, two bidirectional recurrent layers, hidden width 512 per direction, the second layer consuming the 1,024-wide concatenation, then $1{,}024 \to 512$ output projection and LayerNorm.
Shared projector *identical dimensions across backbones and encoders*		
Projector	101.77M	Input $512 \to 2{,}048$, two self-attention blocks, eight heads, feed-forward width 8,192, GELU, dropout 0, and final float32 LayerNorm. Qwen uses pre-normalization. InternVL uses post-residual normalization.

In Qwen, Transformer and xLSTM-inspired predict seven channels separately at each timestep ($d_{\text{out}} = 7$), whereas the executed MLP and Conv1D wrappers mean-pool the sixteen encoder tokens and predict the entire sequence ($d_{\text{out}} = 112$). Both InternVL encoders use the latter pooled head. These heads are used only for Stage-1 reconstruction. During Stage-2 caption-based alignment, language-loss gradients pass through the frozen language model to update the projector while the language-model weights remain fixed. For Qwen, LoRA targets the language-model attention projections `q_proj`, `k_proj`, `v_proj`, `o_proj` and feed-forward projections `gate_proj`, `up_proj`, `down_proj`, with no bias adaptation. Rank 16, scaling 32 and dropout 0.05 give 17,432,576 LoRA parameters. Vision weights remain frozen.

The xLSTM-inspired bidirectional recurrent encoder. The cell carries hidden and vector cell states. It divides exponential input and forget gates by their per-step sum plus 10^{-8}, applies a tanh to the cell update, and uses a sigmoid output gate. This differs from canonical sLSTM (Beck et al., 2024): it has no accumulated normalizer state or recurrent log-domain stabilizer, and its cell recurrence includes the additional tanh. It also has no mLSTM matrix memory. The distinction from sLSTM concerns the recurrence and normalization, not the vector shape of its memory. The artifact identifier is `xlstm`. Qwen uses width 256 per direction and unclipped gate preactivations. InternVL uses width 512 and clips input/forget preactivations above 15 before exponentiation.

Visual processing, precision and software. The Qwen configuration requests six frames per three-second clip. CARLA frame sampling targets 2 Hz, whereas the benchmark nuScenes loader se-

lects six uniformly spaced, endpoint-inclusive frame indices. The training collator provides aspect-preserving resizing, but the benchmark scorer explicitly resizes both sources to 224×224 before the native processor. The processor's image/video minimum and maximum pixel settings are both $64 \times 28 \times 28 = 50{,}176$. This specifies the pixel budget rather than a fixed square resolution. Downstream training reuses frozen visual features. The backbone uses bfloat16 with gradient checkpointing. State-module parameters and normalization computations are kept in float32, with bfloat16 autocasting in the caption and downstream training loops. The staged adapter, decimal text and Stage 4 use SDPA for downstream training. The earlier staged-adapter run uses FlashAttention 2. These three downstream configurations use Python 3.10.19, PyTorch 2.9.0, Transformers 4.57.6, PEFT 0.17.1 and Accelerate 1.11.0. These versions are not established as common to all recipes, and model-revision fields were unset.

Hardware and software. Training and scoring use two RTX 5090 workstations, a DGX Spark (GB10, aarch64) and an H200 cluster, with different PyTorch, CUDA and attention kernels. A same-checkpoint control on two devices agrees on 16,721 of 16,800 items (99.53 percent), with a 0.08-point accuracy difference.

B State history and interface implementations

This section supports Section 3.2's shared-history comparison by specifying coordinates, sampling and numerical interfaces. The seven channels and sampling grid are shared, while text rounds the numerical values.

Notation. $f_{\boldsymbol{\theta}}$ denotes the state encoder, $g_{\boldsymbol{\omega}}$ the state projector, $r_{\boldsymbol{\eta}}$ the temporary Stage-1 reconstruction head, $\boldsymbol{E}_{\boldsymbol{\phi}}$ the frozen backbone's input-embedding map, and $\mathbb{M}_{\mathrm{traj}}$ the ordered set of reserved trajectory positions. The state-adapter interface replaces, in order, the embeddings at $\mathbb{M}_{\mathrm{traj}}$ with the rows of the projected state block $\boldsymbol{Z}$. The text interfaces contain no reserved state positions, and the encoder and projector are outside their computation graph.

Temporal resampling. Each source trajectory contains 31 uniformly sampled states over three seconds at 10 Hz. Endpoint-aligned one-dimensional interpolation resamples the sequence to 16 states. Because both grids include their endpoints, the target sample indexed by $t \in \{1, \dots, 16\}$ corresponds exactly to zero-based source index

$$n_t = \frac{30(t-1)}{15} = 2(t-1). \tag{3}$$

The implemented interpolation therefore equals stride-two selection for this configuration, producing an effective 5 Hz sequence spanning the complete three-second interval.

Channel construction and coordinate frame. Let $\boldsymbol{p}_n = (p_n^x, p_n^y)^\top$ be the stored position in metres, ϑ_n the stored yaw in radians, and s_n the stored longitudinal speed in $\mathrm{m\,s^{-1}}$. The first five channels are

$$\begin{aligned} x_t &= p_{n_t}^x - p_0^x, & y_t &= p_{n_t}^y - p_0^y, \\ v_{x,t} &= s_{n_t} \cos \vartheta_{n_t}, & v_{y,t} &= s_{n_t} \sin \vartheta_{n_t}, \\ \psi_t &= \mathrm{wrap}_{[-\pi,\pi)} \left(\vartheta_{n_t} - \vartheta_0\right). \end{aligned} \tag{4}$$

The origin is translated to the first position, but the axes are not rotated into the initial heading frame. Consequently (x_t, y_t) and $(v_{x,t}, v_{y,t})$ remain expressed in the source world-coordinate axes, whereas heading is relative to the initial yaw. The remaining channels are the stored signed longitudinal acceleration a_t and jerk j_t, sampled on the same grid. They are not recomputed or additionally filtered during model loading.

Text rendering precision. The *in7* serialization prints position and velocity to one decimal place, heading to three, and acceleration and jerk to two. It preserves the channel set and temporal grid of the state-adapter input but not its full floating-point precision, so the text interface is shown a rounded version of exactly the state the state-adapter interface receives.

B.1 INPUT INTEGRITY CHECKS

This subsection provides supplementary input-scale checks and verifies the heading construction.

Heading-wrap audit. Relative heading is wrapped without first unwrapping the source yaw. A corpus check found no resulting discontinuity among 18,634 clips, comprising 14,090 training, 3,544 validation and 1,000 benchmark clips: the largest single-step relative-heading change was 0.240 rad, the largest cumulative excursion was 1.621 rad, and wrapped and unwrapped constructions agreed to 4.44×10^{-16} rad. The two constructions therefore produce the same float32 model input on this corpus.

Scaling and its consequences. Raw training inputs have heading SD 0.172024 rad and position SDs 11.6931 and 13.8289 m, with a largest/smallest raw SD ratio of 80.39 (approximately 80.4). The reconstruction-loss divisor 0.50 for heading is distinct from its raw input SD. For the trained Qwen Transformer encoder, first-layer column norms invert the channel-scale ordering (Spearman $\rho = -1.000$), compared with a column-norm spread of 1.053 across 32 untrained initializations. This compensation is partial: the fitted log-log slope is -0.55, and the resulting effective-contribution spread is 7.4. Table 8 summarizes these integrity checks. They do not establish that raw-unit input is optimal or isolate scaling from learned pathway weights and interface structure.

Table 8: Relative-heading and channel-scale checks on 18,634 clips. Angular quantities are in radians, while scale spreads and norm ratios are dimensionless.

Measurement	Value
Clips whose relative heading wraps past $\pm\pi$	0 of 18,634
Largest single-step heading change (rad)	0.240
Largest cumulative heading excursion, any clip (rad)	1.621
Wrapped minus unwrapped heading, maximum over corpus (rad)	4.44×10^{-16}
Raw per-channel input scale spread (widest / narrowest)	80.4×
Spearman ρ, channel scale vs first-layer column norm	−1.000
Untrained control, column-norm spread (n=32 inits)	1.053×
Effective per-channel contribution spread after training	7.4×

B.2 STATE ENCODER AND TRANSFORMER-BASED STATE PROJECTOR

This subsection supports Section 3.2's definition of the learned state adapter.

Encoder and projector. The encoder maps the state sequence to latent tokens and the projector maps them to the language-model width,

$$\boldsymbol{H} = f_{\boldsymbol{\theta}}(\boldsymbol{X}) \in \mathbb{R}^{K \times d_{\mathrm{enc}}}, \qquad \boldsymbol{Z} = g_{\boldsymbol{\omega}}(\boldsymbol{H}) \in \mathbb{R}^{K \times d_{\mathrm{LLM}}}. \tag{5}$$

The Transformer-based state projector first maps width 512 to d_{LLM}, then applies two temporal self-attention blocks with eight heads and feed-forward width $4d_{\mathrm{LLM}}$, followed by a float32 LayerNorm.

Implementation differences across backbones. The Qwen state projector uses pre-normalized Transformer blocks. The InternVL state projector uses post-residual normalization. Their dimensions and parameter counts are nearly identical, but the operations are not mathematically identical. After injection each backbone applies its native position encoding: Qwen constructs multimodal rotary indices including video MRoPE, whereas InternVL retains its one-dimensional language positions. Qwen uses fixed training-split channel statistics for the Stage-1 target. InternVL uses an accumulated exponential-moving-average estimate unless fixed values are provided. The two implementations therefore share the encoder input but not the exact Stage-1 target-scaling procedure.

B.3 EXACT NUMERICAL-INTERFACE SETTINGS

This subsection supports Section 3.2's matched precision and input-only adaptations.

xVal-style scalar embeddings. The recipe learns one embedding vector of width 2,048 and scales it by each represented value. Each channel is divided by its own q95 constant fitted on all 31 raw samples of each of the 14,090 training clips. The interface subsequently selects the matched sixteen-sample grid. The constants are shared across clips, preserving magnitude differences rather than normalizing each clip independently. Six channels have no training samples beyond the reference $[-5, 5]$ preprocessing range (Golkar et al., 2024). Jerk has a fraction 0.00049. This implementation does not explicitly clip at that range. It trains the scalar vector and LoRA in Stage 3 and occupies 262.0 state-segment positions, including 112 scalar embeddings and formatting.

Input-only xVal comparison. The published method also has a numeric regression head (Golkar et al., 2024). Here we compare its input-embedding idea under the shared language-token objectives and option scorer. Captions and rationales can contain numeric targets, but those are supervised through the language head. Adding a separate number head and numeric loss would change this training recipe. Its possible generative benefit is not evaluated by the present input-only comparison.

Matched LLMTime rendering. The trained recipe uses precision three, an empty digit separator, commas between timesteps and a plain minus sign. Its fixed channel divisors are $[100, 100, 100, 100, 1, 10, 10]$ for $[x, y, v_x, v_y, \psi, a, j]$. Values are multiplied by 10^3 after division and rounded to the nearest integer. Decoding adds no half-bin offset. Thus the quantization steps match decimal in7: 0.1 for position and velocity, 0.001 for heading, and 0.01 for acceleration and jerk. For example, jerk -4.79 is rendered as -479. This matched branch does not use fitted q95 constants or a beta offset.

The matched rendering occupies 482.9 state-segment positions on average, against 780.7 for decimal text and 262.0 for xVal. The separate original-grid LLMTime reference uses train-fitted q95 scaling and its own quantizer. It is measured for context cost and timing, but has no trained accuracy row here. The empty digit separator follows the original method's guidance for tokenizers that already separate digits (Gruver et al., 2023).

CADE-style adapter. The CADE-style adapter uses a linear $7 \to 512$ map per timestep and a per-token MLP projector, retaining sixteen state tokens. Its 4,096-parameter encoder and 5,251,072-parameter projector are trained end-to-end with LoRA in Stage 3, without Stage-1/2 pretraining or an auxiliary contrastive loss. The original work also reports a variant without its supervised contrastive loss (Wu et al., 2026). Evaluation restores the trained encoder and projector from the selected Stage-3 checkpoint. This is a multivariate, sixteen-token adaptation of the pointwise design, not the original univariate CADE configuration. It retains the shared raw *in7* inputs and QA prompt, omitting the original per-series z-scoring and appended mean, standard deviation, minimum, maximum and median (Wu et al., 2026).

The encoder and projector use learning rate 2.5×10^{-5}, alongside LoRA at 2×10^{-4}. This recipe is outside the four-encoder preparation comparison because it bypasses preparation.

C TRAINING OBJECTIVES: DETAILS OMITTED FROM §3.3

This section supports Section 3.3's objectives with target scaling, caption construction and the induced record weighting.

Reconstruction targets. Stage 1 supplies the uncorrupted history to the encoder, without masking or noise. Only the temporary reconstruction target is scaled, using

$$[\boldsymbol{Y}^{(b)}]_{t,c} = \frac{[\boldsymbol{X}]_{t,c}}{\max\left(\widehat{\sigma}_c^{(b)}, 0.5\right)}. \tag{6}$$

Qwen uses fixed training-split channel statistics, while InternVL uses an accumulated exponential moving average unless fixed values are provided. The temporary head predicts $\boldsymbol{Y}^{(b)}$, which is never an encoder input.

Caption-based alignment. Stage 2 sequences contain an instruction, sixteen state positions, a target caption and an end token, without video. Each trajectory has holistic, moderate and detailed captions generated from deterministic trajectory facts, with one variant sampled on access.

Stage-3 objective. All recipes minimize

$$\mathcal{L}_{\mathrm{S3}} = \frac{\sum_{i\in\mathcal{W}} N_i^{-\alpha} S_i}{\sum_{i\in\mathcal{W}} N_i^{1-\alpha}}, \qquad \alpha = 0.5, \tag{7}$$

where S_i sums negative log-likelihood over N_i supervised assistant tokens, including the terminal token, and normalization spans the complete gradient-accumulation window $\mathcal{W}$. Sampling probabilities are not loss weights: at $\alpha = 0.5$, mean token losses are weighted in proportion to $\sqrt{N_i}$.

Length-normalization endpoints. In the Stage-3 objective, $\alpha = 0$ gives token-mean cross-entropy and $\alpha = 1$ gives the mean of per-example token losses. The configured $\alpha = 0.5$ interpolates between them over the complete gradient-accumulation window.

Length weighting under the configured sampler. For mean token loss $\bar{\ell}_i = S_i/N_i$, each record has coefficient $\sqrt{N_i}/\sum_{j\in\mathbb{W}}\sqrt{N_j}$. Table 9 gives the training-corpus length census with the Qwen tokenizer and reconstructed 32-record windows. Coefficient mass differs from sampling probability because response lengths differ. It describes the objective's weighting rather than measured loss contribution or gradient magnitude.

Table 9: Supervised lengths and reconstructed loss-coefficient mass at $\alpha = 0.5$ for 100,000 independent 32-record windows drawn from the configured task mixture. N counts supervised response tokens including end-of-turn. Mass is the mean coefficient share $\sum_{i\in t}\sqrt{N_i}/\sum_j\sqrt{N_j}$, rather than a gradient contribution. Lengths are token counts. Sampling and coefficient mass are percentages.

Task	Sampling	Mean N	Median	P_{10} to P_{90}	Coeff. mass
MC-QA	40%	2.4	2	2 to 3	13.39%
Caption	30%	41.4	35	22 to 69	39.76%
Rationale	30%	61.6	46	35 to 166	46.85%

D SCORING AND STATISTICAL PROCEDURE

This section supports Section 4's accuracy comparisons by defining option scoring, input-intervention changes and checkpoint-conditioned uncertainty.

Multiple-choice scoring. Questions are evaluated by teacher-forced likelihood. For multimodal context $\boldsymbol{C}$ and tokenized option $\boldsymbol{a}^{(m)} = (a_1^{(m)}, \ldots, a_{L_m}^{(m)})^\top$, including its end-of-sequence token, the option score and prediction are

$$s(\boldsymbol{a}^{(m)} \mid \boldsymbol{C}) = \sum_{u=1}^{L_m} \log P(a_u^{(m)} \mid \boldsymbol{C}, \boldsymbol{a}_{<u}^{(m)}). \tag{8}$$

$$\widehat{m} = \arg\max_m s(\boldsymbol{a}^{(m)} \mid \boldsymbol{C}). \tag{9}$$

The implementation sums token log-probabilities without length normalization. Correctness compares predicted and reference answer strings after lowercasing and removing surrounding whitespace. The primary metric is micro-averaged answer accuracy:

$$\mathrm{Acc} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\{\widehat{y}_i = y_i\}}. \tag{10}$$

Both reported sets have equally many examples of each of the fourteen question types, so overall accuracy also equals the unweighted mean of type-specific accuracies. The question-type majority baseline is fitted to each evaluation set's label distribution and obtains 64.2% on development and 48.1% on EgoDyn-Bench. It describes set-specific class imbalance, rather than an independently trained predictor.

Ablation-table groups and label aggregation. In Table 29, *Semantic* covers all fourteen question types. *Temporal* covers the four types marked temporal in the

Table 10: Paired accuracy differences on 16,800 development questions (177 parent scenes) and 14,000 benchmark questions (271 CARLA and 204 nuScenes scenes). The 95% parent-scene bootstrap intervals condition on checkpoints and weight benchmark sources equally. Percentage points.

Contrast	Dev. Δ	Dev. 95% CI	Bench. Δ	Bench. 95% CI
Text minus staged adapter	3.02	[2.60, 3.45]	7.43	[6.84, 8.03]
Text minus joint continuation	2.58	[2.18, 2.98]	6.05	[5.47, 6.65]
Text minus matched LLMTime	0.40	[0.10, 0.70]	1.24	[0.82, 1.65]
Joint continuation minus staged adapter	0.44	[0.12, 0.78]	1.38	[0.96, 1.80]

question configuration: `speed_trend`, `brake_then_turn`, `speed_peak_half`, and `contrastive_sequence`. Accuracy pools records within the indicated group. Balanced accuracy averages recall over label strings with ground-truth support. Macro-F_1 averages classwise F_1 over the union of reference and predicted label strings, assigning zero where precision and recall give no positive F_1. These secondary metrics are computed on pooled labels, so identical label strings in different question types share a class. These metrics are not averages of per-question-type balanced accuracy or F_1. The intervention grid has complete parsing coverage and 16,800 records per condition, including 4,800 temporal records.

Domain-transfer measures. We report development and EgoDyn-Bench accuracy separately, with transfer gap

$$\Delta_{\text{transfer}} = \text{Acc}_{\text{val}} - \text{Acc}_{\text{EgoDyn}}. \tag{11}$$

Source-specific benchmark results concern new trajectories and visual conditions under a shared task taxonomy. The gap is descriptive because the development set participated in checkpoint selection.

Input-reliance effects. For intervention a, the accuracy change from its unmodified checkpoint is

$$\Delta_a = \text{Acc}_a - \text{Acc}_{\text{full}}. \tag{12}$$

A negative change shows sensitivity to the intervention on the evaluated items. It does not establish causal correctness: replacement inputs can lie outside the training distribution.

Generated-answer and caption metrics. For Table 33, *Fmt* is the fraction of outputs with an `Answer:` marker, and *Ans.* is the fraction of all outputs with extracted answers containing the normalized gold label (missing markers are incorrect, and normalized equality yields identical decisions). *Hit* averages each rationale's fraction of reference numbers with a nearest generated number within $\max(0.05|v|, 0.05)$, while *Num. L1* averages within-rationale mean absolute differences over records with defined matches. Caption token accuracy averages per-caption top-1 accuracy, and perplexity exponentiates mean per-caption cross-entropy including EOS, against the same 570 unique holistic captions. Nearest-number scores allow matches anywhere in a rationale and mix source units, so they do not establish correct quantities or factuality (Appendix J).

Uncertainty estimation. Table 10 uses 10,000 paired parent-scene bootstrap draws. The benchmark's 500 CARLA clips occupy 271 scenes, and its 500 nuScenes clips occupy 204 scenes. Within each source, we sample parent scenes with replacement, retain all questions and clips of each sampled scene, and aggregate by question count. The overall estimate fixes the two source weights at one half. Both models receive identical draws. Percentile intervals use NumPy's PCG64 generator with seed 20260914. The corresponding clip-bootstrap sensitivity analysis retains all fourteen questions per clip but treats different clips as independent. It gives the same direction for these contrasts.

The intervals are conditional on selected checkpoints and the observed data distribution. They do not estimate retraining variability or remove development selection effects.

Multiplicity and training variability. Input-intervention tests were exploratory, without family-wise correction across the full grid. Questions sharing a clip and clips sharing a scene are dependent, so aggregate item-level McNemar tests do not support the main interface contrasts. Three staged-adapter training seeds are reported individually and by their observed range. Other interface and encoder recipes generally have one seed. Their training variability is not estimated.

E ACCURACY IN DETAIL

This section supports Section 4.1's accuracy comparison with seed, question, source and state-only results.

Staged-adapter seed variation. Staged-adapter seeds 42, 43 and 44 score 94.3869, 94.3750 and 93.9286 on development and 88.5714, 89.1357 and 90.2000 on the benchmark, respectively. Their observed ranges are 0.4583 development and 1.6286 benchmark points. Subgroup recall varies more widely (Table 24). Seed 42 remains the reference throughout, without benchmark-based reselection. The 300-tree random forest's 90.06 benchmark accuracy falls inside that benchmark span. The strongest seed scores 91.11 on CARLA and 89.29 on nuScenes, against the baseline's 91.13 and 88.99. These ranges describe three staged-adapter training runs and do not establish equivalence between methods. The post-hoc paired intervals in Table 10 instead condition on the evaluated checkpoints and do not estimate variation under retraining.

E.1 ACCURACY RELATIVE TO EACH QUESTION'S FLOOR

This subsection supports Section 4.1's question-level comparison against the trained-text accuracy reference.

Table 11 gives the breakdown for the four trained interfaces, computed from their predictions on the same items. Its floor column identifies questions where a dominant answer leaves little room to distinguish systems. Text leads the adapter on every benchmark question type. xVal sits between the two on nine types, above text on three, and below the adapter on two: stop-and-go at 74.6 against 81.6 and speed regime at 92.9 against 94.9. Its deficit to text is therefore concentrated rather than uniform.

E.2 SOURCE-SPECIFIC ACCURACY

This subsection supports Section 4.1's comparison across CARLA and nuScenes.

Table 1 reports source accuracies for all eleven headline recipes. Table 12 adds paired source-specific intervals, and Table 13 gives every question/source cell for seven trained configurations. These comparisons condition on the checkpoints and do not estimate retraining variability.

The benchmark contains 500 CARLA and 500 nuScenes clips. This source split measures accuracy on each subset. It does not isolate a causal effect of simulation versus reality, because the trajectories and label distributions also differ.

Source differences by question type. The adapter's two source accuracies differ by 0.23 points overall. Across question types, however, the CARLA-minus-nuScenes difference ranges from -17.0 to $+21.2$ points (Table 14). The adapter's mean absolute difference is 7.57 points, compared with 2.69 for decimal text, the smallest among the compared recipes.

On nuScenes relative to CARLA, xVal and the staged adapter lose accuracy on the same two types. Stop-and-go drops by 21.2 points for the adapter and 34.0 for xVal, and driving smoothness by 17.6 and 19.2. Both are decided by absolute thresholds on speed or jerk. The aggregate source gaps therefore contain a shared task-level pattern, with different compensating gains on other types. This does not identify a common mechanism.

Magnitude shift. Relative to train-fitted divisors, mean absolute positions are 0.095 on nuScenes versus 0.213 on CARLA, and velocities are 0.133 versus 0.277. Values outside xVal's comparison range are uncommon, 0.10 versus 0.19 percent, and the input map does not explicitly clip. The shift motivates scale sensitivity analysis without separating it from class priors or trajectory differences.

E.3 STATE-ONLY CALIBRATION BASELINES

This subsection supports Section 4.1's state-only comparison using the same numerical history.

To calibrate how well the state alone determines each concept, two classical model families are fitted on the identical 16×7 raw float grid the state adapter receives, using the CARLA training split only and applied unchanged to the same fixed development and benchmark item lists. Hyperparameters

Table 11: Question-type accuracy for four interfaces on 14,000 benchmark and 16,800 development questions (1,000 and 1,200 per type). Rows follow the benchmark text-minus-staged gap. Floor is the benchmark majority share. Bold marks aggregate means, not rank. Percentages.

Benchmark					
Question type	**Decimal**	**LLMTime**	**xVal**	**Staged**	**Floor**
Contrastive sequence	86.2	79.1	81.9	70.2	33.4
Speed peak half	94.7	93.9	96.8	79.5	37.4
Stop-and-go	95.4	96.5	74.6	81.6	68.0
Turn direction	96.5	95.6	94.7	85.5	34.4
Braking intensity	97.8	98.0	92.3	87.0	34.5
Heading change	99.0	97.6	91.3	90.6	54.7
Driving smoothness	95.6	93.1	88.0	87.2	42.4
Brake then turn	97.4	95.8	95.5	92.1	65.6
Motion axis	93.5	91.0	92.6	88.3	51.4
Lateral acceleration	95.4	96.2	95.5	91.4	63.3
Speed regime	98.7	98.5	92.9	94.9	31.9
Low mean speed	99.6	99.4	98.7	98.5	54.7
Speed trend	97.0	95.2	96.7	96.2	33.5
Extreme maneuver	97.2	96.8	98.9	97.0	68.0
Mean	**96.0**	**94.8**	**92.2**	**88.6**	48.1

In-domain development				
Question type	**Decimal**	**LLMTime**	**xVal**	**Staged**
Contrastive sequence	91.2	91.2	89.6	83.0
Speed peak half	95.8	95.9	96.0	93.4
Stop-and-go	99.6	98.1	99.2	99.4
Turn direction	97.2	96.8	95.3	86.9
Braking intensity	99.2	98.9	96.8	93.2
Heading change	99.0	98.6	96.8	96.3
Driving smoothness	96.7	96.3	97.3	95.1
Brake then turn	98.2	97.2	97.2	95.1
Motion axis	95.2	94.9	95.3	94.7
Lateral acceleration	96.6	96.8	95.8	93.3
Speed regime	98.5	97.2	96.2	95.7
Low mean speed	99.3	99.5	98.4	98.8
Speed trend	97.7	97.2	97.1	97.8
Extreme maneuver	99.5	99.5	97.8	98.8
Mean	**97.4**	**97.0**	**96.4**	**94.4**

Table 12: Source-specific paired accuracy differences on 7,000 benchmark questions per source. Brackets give 95% parent-scene bootstrap intervals conditional on the checkpoints. Percentage points.

Contrast	CARLA: difference [95% CI]	nuScenes: difference [95% CI]
Decimal minus staged	+7.46 [6.62, 8.32]	+7.40 [6.54, 8.23]
Decimal minus joint	+6.43 [5.61, 7.31]	+5.67 [4.86, 6.48]
Decimal minus single MLP	+4.74 [3.93, 5.63]	+5.71 [4.87, 6.59]
Joint minus staged	+1.03 [0.46, 1.62]	+1.73 [1.13, 2.32]

Table 13: Benchmark accuracy by source and question type: 500 questions per row, 7,000 per source. Joint denotes joint continuation and MLP the single-model seed-0 baseline. Bold/underlined: best/second per row across seven recipes, with ties at displayed precision. Ranks are descriptive. Percentages.

Question	Decimal	LLMTime	xVal	Staged	CADE	Joint	MLP
CARLA							
Speed regime	**98.0**	97.6	95.0	93.0	93.0	93.8	91.8
Low mean speed	**99.8**	**99.8**	98.4	99.0	99.4	99.6	97.4
Speed trend	**96.6**	93.6	95.6	96.4	94.8	94.6	95.8
Speed peak half	93.6	92.0	**96.0**	74.6	95.8	81.0	95.4
Stop-and-go	95.8	95.4	91.6	92.2	95.8	92.8	**97.0**
Contrastive sequence	**84.8**	76.6	79.6	70.2	77.0	74.6	72.6
Braking intensity	**99.8**	99.6	93.0	94.2	95.0	95.2	94.4
Driving smoothness	**98.6**	97.0	97.6	96.0	95.2	95.8	95.4
Heading change	**98.2**	96.2	92.6	83.8	59.0	84.8	91.6
Turn direction	**95.2**	93.6	93.8	85.0	59.6	84.2	85.6
Lateral acceleration	**98.4**	97.4	97.2	90.0	74.6	91.6	93.8
Brake then turn	**97.6**	95.0	95.4	91.0	80.4	90.0	90.8
Motion axis	**90.8**	83.8	86.4	79.8	69.4	80.6	82.8
Extreme maneuver	**98.8**	97.8	98.4	96.4	97.0	97.4	95.2
nuScenes							
Speed regime	**99.4**	**99.4**	90.8	96.8	98.2	96.4	94.6
Low mean speed	**99.4**	99.0	99.0	98.0	98.4	96.6	97.6
Speed trend	97.4	96.8	**97.8**	96.0	96.8	96.0	96.6
Speed peak half	95.8	95.8	**97.6**	84.4	95.2	84.8	96.0
Stop-and-go	95.0	**97.6**	57.6	71.0	84.2	76.6	95.6
Contrastive sequence	**87.6**	81.6	84.2	70.2	69.6	79.0	81.6
Braking intensity	95.8	**96.4**	91.6	79.8	91.8	81.8	77.2
Driving smoothness	**92.6**	89.2	78.4	78.4	71.8	85.8	71.8
Heading change	**99.8**	99.0	90.0	97.4	66.2	97.4	96.0
Turn direction	**97.8**	97.6	95.6	86.0	55.0	86.8	79.8
Lateral acceleration	92.4	**95.0**	93.8	92.8	88.0	92.6	89.4
Brake then turn	**97.2**	96.6	95.6	93.2	88.2	93.0	91.0
Motion axis	96.2	98.2	**98.8**	96.8	86.8	96.6	96.2
Extreme maneuver	95.6	95.8	**99.4**	97.6	99.2	99.2	98.6

Table 14: CARLA minus nuScenes accuracy on 7,000 benchmark questions per source. Aggregate compares full source halves. Remaining columns summarize question-type differences (500 questions per source and type). Positive values favor CARLA, without implying an optimization rank. Percentage points.

Interface	Aggregate	Mean |swing|	Largest drop	Largest gain
Decimal text	+0.29	2.69	+6.0	-5.4
Staged adapter	+0.23	7.57	+21.2	-17.0
xVal-style	+2.89	6.37	+34.0	-12.4

Table 15: Non-VLM baselines on the 112 raw floats. Per-type fits fourteen classifiers, single fits one model with the question type as an input feature. Evaluation uses 16,800 development and 14,000 benchmark questions. Bold/underline mark best/second per column (↑), descriptively. Percentages.

	Per-type (14 models)		Single model	
Family	Dev.	EgoDyn	Dev.	EgoDyn
Random forest (300 trees)	**95.27**	**90.06**	<u>94.49</u>	<u>88.08</u>
MLP (256×128)	<u>94.93</u>	<u>89.91</u>	**95.21**	**90.77**

are fixed in advance at 300 trees for the random forest and a 256×128 hidden stack with a 400-iteration cap for the MLP. There is no hyperparameter search, and no baseline is fitted, selected or early-stopped on development or benchmark data.

Per-type and single-model fits. Table 15 compares per-type and single-model fits. The primary fit trains one classifier per question type, fourteen independent models against one set of VLM weights. Because that asymmetry favours the baseline, we also fit a single model over all fourteen types with the question type appended as a one-hot block, giving a 126-wide input and one classifier over the union of the answer labels. Its argmax is restricted to the labels its question type was trained on, which is the single model's counterpart of a per-type classifier's label space and of the choice list the VLM is shown.

What the single-model fit shows. Switching to one model raises the MLP by 0.86 benchmark points and lowers the random forest by 1.98. The MLP's strong accuracy therefore does not depend on fitting separate models per question type. Over seeds 0, 1 and 2 the single MLP scores 90.77, 90.56 and 90.99 on the benchmark and 95.21, 94.82 and 95.08 in domain, a benchmark range of 0.43 points against the adapter's 1.63. The reported cells use seed 0, the seed the per-type fit also uses. Taken together the four configurations span 88.08 to 90.99 on the benchmark, a range that contains the adapter's 88.57 to 90.20 rather than sitting below it. These baselines receive the same numerical state without video, question text or rationale supervision. Only the per-type fits use fourteen independent parameter sets.

Where the MLP comparison differs. Table 16 reports every question type for the seed-0 MLP, decimal text, staged adapter and joint continuation. The MLP beats the staged adapter on six types, ties one and trails seven. Speed peak half, stop-and-go and contrastive sequence contribute 378 net additional correct answers, exceeding the MLP's overall advantage of 308 because the other types offset 70. Against joint continuation, those three types contribute 247 correct, offset by 132 elsewhere, leaving 115 overall. These are signed contributions to observed count differences, not disjoint causes of error. The results support the MLP as a strong standalone state-defined classifier, while the adapter's context benefit concerns supplying state to an existing VLM.

E.4 Worked examples

This subsection supports Section 4.4's question-level diagnostics through three complete benchmark items (Figure 4). Each card shows six input frames, the 10 Hz sensor track and each recipe's answer. The question types have the largest benchmark gaps between text and the staged adapter (16.0, 15.2 and 13.8 points). A fixed-seed draw from one agreement group supplies each example, whose header states its frequency.

Table 16: Benchmark accuracy of the seed-0 single-model state-only MLP and three VLM checkpoints: 1,000 questions per type, 14,000 total. Bold/underlined: best/second distinct value per row across four recipes. Ranks are descriptive. Percentages.

Question	Decimal text	State-only MLP	Staged	Joint continuation
Speed regime	**98.70**	93.20	94.90	<u>95.10</u>
Low mean speed	**99.60**	97.50	<u>98.50</u>	98.10
Speed trend	**97.00**	<u>96.20</u>	<u>96.20</u>	95.30
Speed peak half	<u>94.70</u>	**95.70**	79.50	82.90
Stop-and-go	<u>95.40</u>	**96.30**	81.60	84.70
Contrastive sequence	**86.20**	<u>77.10</u>	70.20	76.80
Braking intensity	**97.80**	85.80	87.00	<u>88.50</u>
Driving smoothness	**95.60**	83.60	87.20	<u>90.80</u>
Heading change	**99.00**	<u>93.80</u>	90.60	91.10
Turn direction	**96.50**	82.70	<u>85.50</u>	<u>85.50</u>
Lateral acceleration	**95.40**	91.60	91.40	<u>92.10</u>
Brake then turn	**97.40**	90.90	<u>92.10</u>	91.50
Motion axis	**93.50**	<u>89.50</u>	88.30	88.60
Extreme maneuver	<u>97.20</u>	96.90	97.00	**98.30**
Overall (14,000 items)	**96.00**	<u>90.77</u>	88.57	89.95

Specifically, the contrastive-sequence card shows braking from 12.0 to 5.4 m/s, with yaw rate reaching -1.02 rad/s after the midpoint. The oracle labels the second half more dynamic. The speed-peak-half card is one of 179 items where text is correct and the staged adapter is wrong: initial speed 7.67 m/s exceeds a later rise to 7.36 m/s. The stop-and-go card is one of 46 text errors, shared here by all displayed trained recipes: speed falls to 0.23 m/s at 2.0 s and reaches approximately 2.001 m/s at 2.9 s, just above the strict 2 m/s threshold. The sixteen-sample grid omits 2.9 s, and rounding to 2.00 obscures the threshold crossing. These items illustrate individual discrepancies rather than their prevalence or cause. Appendix M gives the aggregate sensitivity checks. External zero-shot answers differ in model, inputs and training.

F Context, arithmetic and timing

This section supports Section 4.2's context, arithmetic and measured prefill comparisons.

Timing protocol. Table 19 repeatedly times one fixed collated prompt shape from an eight-clip input, with cached vision and the state adapter included in prefill. Measurements follow warmup. Median prefill is 36.85 ms for the staged adapter and 56.65 ms for decimal text. The maximum prefill batches, 89 and 42, measure memory feasibility rather than throughput. Batch-one peak memory spans 4.79 to 5.14 GiB, and 64-token generation totals are directly timed. Decode is a residual estimate, not a per-token measurement. The LLMTime timing row uses the original-grid rendering, not the trained matched rendering.

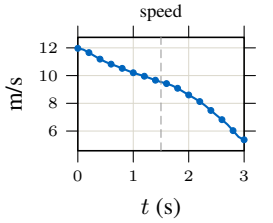
Populations. State-segment lengths use 18,679 available CARLA trajectory arrays, while state-block and full-input means use 49,616 validation questions over 3,544 clips. Thus $780.7/16 = 48.8$ and approximately $1{,}425/661 = 2.16$ describe different populations, and subtracting state-segment from state-block means does not estimate wrapper length. Table 18 also includes untrained renderings. Matched LLMTime uses 482.9 positions versus decimal text's 780.7.

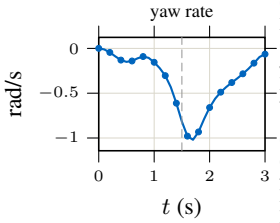
Where the parameters reside. Table 17 separates encoder and projector parameters. The projector comprises approximately 97 percent or more of the Qwen adapter. The InternVL recurrent encoder has a larger 9.8-percent share. Encoder-only changes therefore alter little of the Qwen pathway's total capacity.

Analytical convention. A prompt forward costs two FLOPs per multiply-accumulate with causal attention, one vocabulary-output position and approximate state-pathway arithmetic $2PK$ for stored parameter count P and $K = 16$. Vision is excluded. The option scorer instead evaluates each

(a) CARLA clip `DEU_Herzogenaurach-78_1_T-7__Comfort__w0`, *contrastive sequence*. Cell: every interface right, no-state wrong (373 of 1,000 items of this type).





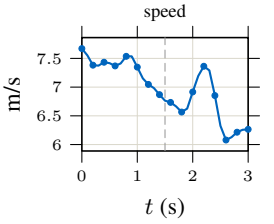


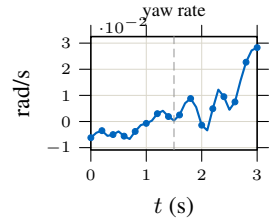
Q: Comparing the first and second halves of the sequence, which half has more dynamic driving?
Options: first_half, second_half, similar

Model	Answer	
Oracle	*second_half*	
Qwen3-VL-2B, zero-shot, video only	second_half	✓
Gemini 3 Pro, zero-shot, video + state as text	first_half	✗
No state (trained)	similar	✗
Frozen random pathway	second_half	✓
Learned adapter, K=16	second_half	✓
xVal scalar embeddings	second_half	✓
LLMTime text, matched grid	second_half	✓
in7 text	second_half	✓

(b) nuScenes clip `clip_04395`, *speed-peak half*. Cell: text right, adapter wrong (179 of 1,000 items of this type).





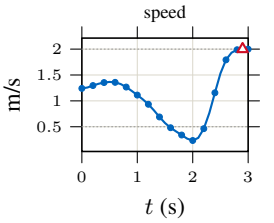


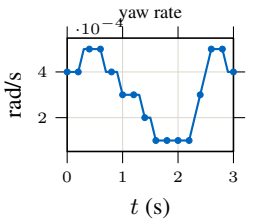
Q: Does the maximum speed occur in the first or second half of the sequence?
Options: first_half, second_half, no_peak

Model	Answer	
Oracle	*first_half*	
Qwen3-VL-2B, zero-shot, video only	second_half	✗
Gemini 3 Pro, zero-shot, video + state as text	first_half	✓
No state (trained)	first_half	✓
Frozen random pathway	second_half	✗
Learned adapter, K=16	second_half	✗
xVal scalar embeddings	first_half	✓
LLMTime text, matched grid	first_half	✓
in7 text	first_half	✓

(c) CARLA clip `DEU_Kitzingen-34_1_T-3__Default__w2`, *stop-and-go*. Cell: text wrong (46 of 1,000 items of this type).







Omitted: 2.9 s, $\approx 2.001 > 2.0$ m/s.
Sampled: 1.988 at 2.8 s and 1.999 at 3.0 s.

Q: Does the vehicle exhibit stop-and-go behavior?
Options: yes, no

Model	Answer	
Oracle	*yes*	
Qwen3-VL-2B, zero-shot, video only	no	✗
Gemini 3 Pro, zero-shot, video + state as text	yes	✓
No state (trained)	no	✗
Frozen random pathway	no	✗
Learned adapter, K=16	no	✗
xVal scalar embeddings	no	✗
LLMTime text, matched grid	no	✗
in7 text	no	✗

Figure 4: Worked examples from the three types with the largest text-minus-adapter gap, sampled with a fixed seed from the named agreement cells. Frames are model inputs. Traces show the 10 Hz track. Blue circles mark sixteen sampled input times $(0.0, 0.2, \ldots, 3.0\,\text{s})$. The red triangle marks the omitted 2.9 s crossing. Dashed verticals denote half boundaries and dotted horizontals denote thresholds. Zero-shot answers are leaderboard outputs.

candidate and materializes logits at all positions, so this convention does not price a complete scored question.

Table 17: Adapter parameters and $2PK$ arithmetic proxies by component for both backbones. Untrained InternVL MLP and Conv1D configurations are included only for size accounting. Stored parameters include inactive positional rows. Parameters are in millions, arithmetic in GFLOP and encoder shares in percent.

Encoder recipe	Encoder	Projector	State adapter	LoRA	Enc./Adapter	Enc. GF	Path. GF
Qwen3-VL-2B *same projector and LoRA budget, four encoder architectures*							
Transformer	3.15	101.77	104.92	17.43	3.0%	0.101	3.36
xLSTM-inspired	2.90	101.77	104.67	17.43	2.8%	0.093	3.35
MLP	0.14	101.77	101.91	17.43	0.1%	0.004	3.26
Conv1D	0.048	101.77	101.82	17.43	0.0%	0.002	3.26
InternVL3.5-2B *same projector and LoRA budget, four encoder architectures*							
Transformer	3.67	101.77	105.44	17.43	3.5%	0.117	3.37
xLSTM-inspired	11.03	101.77	112.80	17.43	9.8%	0.353	3.61
MLP	0.28	101.77	102.05	17.43	0.3%	0.009	3.27
Conv1D	0.048	101.77	101.82	17.43	0.0%	0.002	3.26

Table 18: Input costs: segments exclude wrappers (18,679 CARLA arrays). Blocks include them, and full sequences include video and instructions (49,616 validation questions). Vision arithmetic is excluded. Benchmark-v1.0 text and original-grid LLMTime are untrained cost references with different grids. Bold/underlined: lowest/second-lowest length and GFLOP per column across seven configurations, with ties. Ranks are descriptive. Lengths are positions, shares are percentages and the final column is the GFLOP ratio to the staged adapter.

Interface	State segment↓	State block↓	Full sequence↓	State (%)	Video (%)	GFLOP↓	GFLOP ×staged
Staged adapter	**16**	**26**	**661**	3.9	86.3	**1917**	1.00
No-state (zeroed grid)	**16**	**26**	**661**	3.9	86.3	**1917**	1.00
xVal-style	262	278	913	30.5	62.5	2670	1.39
Benchmark text	342	352	986	35.6	57.8	2891	1.51
LLMTime-style (original grid)	536	538	1173	45.9	48.6	3465	1.81
LLMTime-style (matched grid)	483	488	1123	43.5	50.8	3311	1.73
Decimal text	781	790	1425	55.5	40.0	4250	2.22

Table 19: RTX 5090 bf16 timing on one fixed prompt per interface with cached vision. Prefill uses 10 repetitions after 3 warmups, and generation uses 3 after 1. Peak is batch-one reserved memory and max batch is memory-limited prefill capacity. Decode is a residual estimate, not a per-token measurement. Original-grid LLMTime is cost-only. Bold/underlined: best/second per column across six configurations for sequence length, prefill, peak memory (↓), and max batch (↑). Ranks are descriptive. Units are positions, milliseconds, milliseconds/token and GiB. Ratios use the staged adapter.

Interface	Full sequence positions↓	Prefill (ms)↓	Prefill ×staged	Decode (ms/token)	Peak (GiB)↓	Max batch↑	Positions ×batch
Staged adapter	**655**	**36.8**	1.00	17.02	**4.79**	**89**	58,295
No-state (zeroed grid)	**655**	**36.8**	1.00	16.90	4.80	**89**	58,295
xVal-style	907	39.1	1.06	17.01	4.94	63	57,141
Benchmark text	977	42.5	1.15	16.96	4.94	57	55,689
LLMTime-style (original grid)	1,198	44.3	1.20	17.00	5.07	49	58,702
Decimal text	1,405	56.6	1.54	17.06	5.14	42	59,010

G ENCODER VARIANTS AND PREPARATION

This section supports Section 4.3's comparison of encoder designs and preparation metrics against downstream accuracy.

G.1 ENCODER ARCHITECTURES

Table 20 supports the four-family comparison at sixteen state positions with the same Transformer-based projector. The MLP encoder still feeds the 101.8M-parameter temporal projector. The staged adapter leads xLSTM-inspired by approximately 0.59 development and 1.09 benchmark points. The four designs span approximately 4.3 and 8.4 points, respectively. Training budgets are in Table 4.

Encoder capacity. Stored encoder counts are 3.15M, 2.90M, 0.14M and 0.048M for Transformer, recurrent, MLP and Conv1D, with development accuracies 94.4, 93.8, 92.0 and 90.1. The Transformer allocates a $1{,}000 \times 512$ positional table but reads only sixteen rows, leaving 503,808 parameters unused. Its active count is consequently 2.65M, below the recurrent encoder's 2.90M despite higher accuracy. Capacity describes the coarse separation from the small encoders (4.4 and 1.5 percent of the Transformer's stored count, losing 2.4 and 4.3 development points), but does not resolve the leading pair. The single-seed, unmatched-capacity panel compares complete designs.

Table 20: Qwen encoder accuracy on 16,800 development and 14,000 benchmark questions. Gap is development minus benchmark. Development accuracies of the three cluster runs are rounded to ± 0.05. Bold/underlined: best/second per column across four recipes, with directions shown. Ranks are descriptive. Accuracies are percentages and gaps are percentage points.

Encoder	Dev. acc.↑	EgoDyn Acc.↑	Gap↓
Transformer	**94.4**	**88.6**	**5.8**
xLSTM-inspired	<u>93.8</u>	<u>87.5</u>	<u>6.3</u>
MLP	92.0	84.0	8.0
Conv1D	90.1	80.2	9.9

G.2 PREPARATION METRICS AND DOWNSTREAM ACCURACY

Table 21 supports Section 4.3's finding that preparation and downstream metrics rank completed configurations differently. Stage 1 scores the final encoder used by Stage 2, and Stage 2 scores the selected caption checkpoint used downstream. Development and benchmark scores match Table 20. The recurrent encoder reconstructs more accurately than the Transformer but answers less accurately. Conv1D has the second-lowest caption loss and lowest downstream accuracy.

Comparison scope. Stage 1 uses temporary heads with different pooling and training budgets, rather than a common probe of projected state positions. Stage 2 samples caption variants over 3,544 validation records, without identical targets across checkpoints. These diagnostics rank the tested configurations without establishing a general relationship between preparation quality and downstream performance.

Input standardization. The standardized-input Qwen Transformer adapter reaches 93.58% development accuracy on $16{,}800$ questions and 88.68% on the $14{,}000$ benchmark questions. Its benchmark accuracy is 0.11 percentage points above the selected raw-input staged adapter and within that configuration's three-seed range of 88.57% to 90.20%. The effective Stage-1 batch and preparation run differ, so this is a recipe comparison rather than an isolated scaling ablation.

H WHERE THE REMAINING ERRORS ARE

This section supports Section 4.4's localization of remaining errors by tracing the aggregate deficit to question types, low-speed clips and label shifts. These descriptive comparisons do not identify a unique mechanism.

Table 21: Qwen preparation metrics on 49,616 reconstruction records and 3,544 caption records, alongside accuracy on 16,800 development and 14,000 benchmark questions. Bold/underlined: best/second per column across four recipes. Arrows give the preferred direction. Ties share markings. ADE is in metres, HE in radians, CE in nats/token and accuracies in percent.

Encoder	**Stage 1**		**Stage 2**		**Stage 3**	
	ADE (m)↓	HE (rad)↓	CE↓	Tok. (%)↑	Dev. (%)↑	Bench (%)↑
Transformer	0.353	<u>0.0097</u>	**0.4395**	**84.7**	**94.4**	**88.6**
xLSTM-inspired	**0.084**	**0.0027**	0.6086	81.3	<u>93.8</u>	<u>87.5</u>
MLP	<u>0.180</u>	0.0125	0.5166	<u>83.1</u>	92.0	84.0
Conv1D	0.533	0.0218	<u>0.5008</u>	<u>83.1</u>	90.1	80.2

Table 22: Development-to-benchmark widening of the accuracy gap between text and the staged adapter: 16,800 development questions and 7,000 per benchmark source. The lower panel orders question types by widening. Accuracy is in percent. Gaps and widening are percentage points.

Recipe	**Development**	**Bench CARLA**	**Bench nuScenes**
By evaluation half *accuracy, percent*			
Decimal text	97.40	96.14	95.86
Staged adapter	94.39	88.69	88.46
xVal	96.36	93.61	90.73
text minus adapter	+3.02	+7.46	+7.40
widening against development	n/a	+4.44	+4.38
By question type *text minus adapter, in domain then under shift, then the change*			
Stop-and-go	0.2	13.8	+13.6
Speed peak half	2.3	15.2	+12.9
Contrastive sequence	8.2	16.0	+7.8
Driving smoothness	1.6	8.4	+6.8
Heading change	2.7	8.4	+5.7
Braking intensity	6.1	10.8	+4.7
top three of fourteen	n/a	n/a	55% of the total

H.1 State access and learned encoding

This subsection supports Sections 4.1, 4.3 and 4.4 with controls and development-to-benchmark gap decomposition.

The no-state and frozen-random controls distinguish removing trajectory-specific information from retaining a fixed untrained transformation of it. The staged adapter exceeds frozen random by 2.24 development points and 7.21 benchmark points (Table 1). These are differences between complete trained recipes.

Where the transfer gap widens. To locate the remaining deficit, we compare its size across evaluation sets. The difference between decimal text and the staged adapter grows from 3.02 points on development to 7.43 on the benchmark. Table 22 shows widening on both CARLA and nuScenes, so it is not confined to the real subset. This source comparison does not isolate the effect of simulation versus reality, because trajectories and label distributions differ. Three question types account for 55 percent of the widening, motivating the task-specific diagnostics below.

Confusion patterns. Table 23 records structured rather than arbitrary errors. On the reported ordered scales, most confusions concern neighboring categories. The adapter predicts *no_peak* on 10.8 percent of benchmark clips against a true rate of 27.8, and *similar* on 17.0 against 33.3. Text tracks both rates within four points. These observations describe underprediction of particular classes. They do not distinguish limited numerical resolution, learned class priors or other distribution-shift effects.

Table 23: Error structure for decimal text and the staged adapter on 14,000 benchmark questions (1,000 per type). Null-class rows compare label prevalence with prediction frequency. Bold highlights the two largest staged-adapter underprediction gaps, not rank. Neighbour rows give the share of errors on adjacent ordered classes, for which prevalence is inapplicable. Percentages.

Question type	Null class	True	Text	Adapter
How often the no-event class is predicted *against how often it is the label*				
Speed peak half	no_peak	27.8	24.4	**10.8**
Contrastive sequence	similar	33.3	31.7	**17.0**
Turn direction	straight	34.4	34.1	33.0
Braking intensity	none	32.1	32.8	28.4
Speed trend	steady	33.5	35.1	35.9
Motion axis	none	24.3	22.5	22.9
Share of errors landing on a neighbouring class *ordered scales only*				
Turn direction	neighbour	n/a	37	58
Braking intensity	neighbour	n/a	100	100
Speed regime	neighbour	n/a	100	100
Speed trend	neighbour	n/a	100	100

Table 24: Three staged training runs on the same benchmark (14,000 questions) and identified low-speed subgroup (175 clips with gold *no peak* and maximum speed below 4 m/s). The subgroup was selected post hoc, and 172 of its clips have maximum speed below 0.5 m/s. All checkpoints retain their original validation selection, and seed 42 remains the reference.

Staged seed	Benchmark (%)	Subgroup correct	Recall (%)
42 (reference)	88.57	14/175	8.0
43	89.14	94/175	53.7
44	90.20	172/175	98.3

H.2 What the diagnostics can localize

This subsection supports Section 4.4's low-speed comparison across recipes and training runs, class concentration and continuation results.

Low-speed flat-profile errors. Table 25 conditions on the gold *no_peak* class and bins different clips by maximum speed. The adapter's recall ranges from 8.0 percent below 4 m/s to 92.3 percent in the 8 to 14 m/s bin. Of the 175 clips in the lowest bin, 172 have maximum speed below 0.5 m/s. Furthermore, 58 enter the oracle's separate near-zero branch: when absolute mean speed is at most 10^{-6}, it uses an absolute peak check rather than the relative 5 percent prominence rule. The bins therefore do not vary only the absolute size of one fixed relative distinction.

For the seed-42 reference, the low-bin deficit persists within both sources: recall is 10.9 percent on 119 CARLA clips and 1.8 percent on 56 nuScenes clips, against text at 95.0 and 98.2. The CADE-style adapter correctly labels 114/119 CARLA and 53/56 nuScenes clips, for pooled recall of 167/175 (95.4%, scene-bootstrap interval $[90.2, 99.3]$). Architecture, preparation and downstream trainability differ together in this comparison. Table 28 includes source-specific intervals.

The same subgroup across staged runs. Seeds 42, 43 and 44 correctly label 14, 94 and 172 of these same 175 clips (Table 24). Thus a 1.63-point aggregate accuracy range conceals a 90.29-point recall range. Seed 44 demonstrates that the staged architecture and sixteen-position budget can avoid the severe seed-42 failure. The three complete training runs vary preparation and downstream training, so these counts do not locate the source of the variation or estimate its frequency in future runs. The subgroup was identified post hoc. This comparison reuses Table 25's existing speed bins and retains all validation-selected checkpoints.

Joint continuation. Stage 4 updates 122.4M parameters for 25,131 additional optimizer steps and closes about a fifth of the staged adapter's benchmark deficit to text (Section 4.3).

Table 25: Recall on the benchmark speed-peak-half `no_peak` class, stratified by maximum speed: 278 clips, with bin counts shown. The oracle uses relative prominence away from zero mean and a separate near-zero branch. Bins differ in source, trajectory shape and training support. Percentages.

Clip max speed (m/s)	n	Text	LLMTime	xVal	Adapter	Frozen
< 4	175	96.0	88.6	99.4	8.0	0.0
4 to 8	38	71.1	78.9	100.0	76.3	55.3
8 to 14	39	76.9	74.4	97.4	92.3	59.0
> 14	26	61.5	76.9	69.2	80.8	80.8

Selected classes and calibration. Beyond the low-speed subgroup, we examine class-level error concentration. The diagnostic group contains six pairs of question type and label: contrastive sequence/similar, speed peak half/no peak, braking intensity/low and moderate, driving smoothness/moderate, and motion axis/lateral. The comparison group contains eight pairs: contrastive sequence/first and second half, speed peak half/first and second half, braking intensity/emergency, driving smoothness/smooth and aggressive, and motion axis/longitudinal. Recalls average classes equally within each group. The subsets were motivated by answer-vocabulary distinctions between intermediate and extreme labels. They do not form a consistent or exhaustive taxonomy and are treated as post-hoc descriptive subsets. In particular, lateral dominance is an upper ratio region rather than an interior band. The six diagnostic classes occupy 10.5 percent of benchmark items and account for 43.7 percent of the adapter's excess errors against text. Eliminating their excess errors would raise adapter accuracy to 91.81, still below text at 96.00. Tables 26 and 27 report all classes defined by question type and label, their source counts and five models' recalls, marking these subsets. Table 28 additionally compares the same low-speed subgroup after continuation, with the CADE-style adapter and with the state-only MLP. Continuation raises recall from 14/175 to 60/175 correct, while the CADE-style adapter gets 167/175 and the MLP 171/175. This comparison connects training and recipe differences to the same low-speed failure.

Label-shift correction. To assess how much calibration can recover, we apply BBSE and EM. They recover 0.85 and 0.90 staged-adapter points, versus 0.19 and 0.18 for text. Fitting uses source labels and unlabelled target predictions, with target labels used only for scoring. The text BBSE result is unchanged when its source confusion matrix replaces the adapter's. Results remain separate from the uncorrected comparison, and the largest adapter EM improvement occurs on stop-and-go.

Stop-and-go priors. The training "yes" rate is approximately 0.5%, versus 32.0% on the benchmark. On development, the earlier staged checkpoint scores 99.4167% intact, 99.5000% with mean-state replacement and 99.4167% after shuffling, against a 99.0% majority floor (1,188/1,200). This imbalance limits what aggregate accuracy reveals about temporal dependence (Table 31). Excluding stop-and-go leaves benchmark adapter/text accuracy at 89.11/96.05 (gap 6.94) and development accuracy at 94.00/97.24 (gap 3.24). The primary comparison includes all fourteen types.

I Input interventions

This section supports Section 4.4's distinction between task accuracy and temporal-input dependence. The Qwen Transformer suite tests 33 conditions on 16,800 development questions without optimization: channel zero, training-mean and noise replacements, temporal shuffle, reversal and averaging, and visual or joint visual/state removal. It uses the earlier staged checkpoint (93.82 development), while the four-condition recurrent comparison uses its selected checkpoint. These are inference-time interventions, not independently trained baselines.

I.1 Channel dependence

Table 29 supports the channel-dependence analysis with paired interventions. Zero and training-mean replacement give complementary checks, while the no-video rows additionally change visual inputs.

Acceleration removal has the largest single-channel effect, reducing accuracy from 93.8 to 78.3 with zero replacement and 76.2 with mean replacement. Removing both velocity components leaves

Table 26: Benchmark class recall and source counts, part 1 of 2 (14,000 questions across both parts). n_C and n_N count CARLA and nuScenes items. D marks six diagnostic classes and C eight comparison classes. Blank cells mark neither subset. These post-hoc sets are not a difficulty taxonomy. Joint denotes joint continuation and MLP the single-model seed-0 baseline. Bold/underlined: best/second per row across five recipes, with ties. Ranks are descriptive. Percentages.

Question	Gold class	Set	n_C	n_N	Decimal	xVal	Staged	Joint	MLP
Speed regime	highway		183	0	**99.5**	94.0	92.9	90.2	90.7
	slow		31	288	**100.0**	83.7	96.6	96.6	92.8
	stopped		123	81	97.5	**98.5**	**98.5**	**98.5**	98.0
	urban		163	131	97.6	**98.3**	91.8	94.2	91.8
Low mean speed	no		333	120	**99.6**	97.8	97.6	96.9	94.7
	yes		167	380	99.6	99.5	99.3	99.1	**99.8**
Speed trend	accelerating		58	274	97.0	97.6	96.4	97.6	**98.5**
	decelerating		244	89	96.1	**100.0**	94.3	94.3	94.0
	steady		198	137	**97.9**	92.5	**97.9**	94.0	96.1
Speed peak half	first_half	C	230	118	96.6	**98.0**	96.8	96.8	96.3
	no_peak	D	191	87	86.7	**96.4**	36.0	47.1	92.4
	second_half	C	79	295	**98.9**	96.0	95.7	96.5	97.6
Stop-and-go	no		454	226	**100.0**	**100.0**	**100.0**	**100.0**	99.7
	yes		46	274	85.6	20.6	42.5	52.2	**89.1**
Contrastive sequence	first_half	C	179	155	89.5	89.8	82.3	88.6	**95.5**
	second_half	C	143	190	91.9	**94.0**	93.7	91.0	90.4
	similar	D	178	155	**77.2**	61.9	34.5	50.8	45.3
Braking intensity	emergency	C	257	88	99.1	**100.0**	98.3	98.6	97.4
	low	D	42	152	**95.9**	91.2	90.2	87.6	63.4
	moderate	D	55	85	**92.9**	57.9	61.4	62.9	63.6
	none		146	175	**99.7**	**99.7**	84.1	89.4	96.6

Table 27: Benchmark class recall and source counts, part 2 of 2 (14,000 questions across both parts). n_C and n_N count CARLA and nuScenes items. D marks six diagnostic classes and C eight comparison classes. Blank cells mark neither subset. These post-hoc sets are not a difficulty taxonomy. Joint denotes joint continuation and MLP the single-model seed-0 baseline. Bold/underlined: best/second per row across five recipes, with ties. Ranks are descriptive. Percentages.

Question	Gold class	Set	n_C	n_N	Decimal	xVal	Staged	Joint	MLP
Driving smoothness	aggressive	C	271	153	<u>93.9</u>	**99.8**	84.7	87.3	83.7
	moderate	D	49	239	**96.2**	62.2	78.8	<u>88.2</u>	67.7
	smooth	C	180	108	97.6	96.5	**99.3**	<u>98.6</u>	**99.3**
Heading change	no		223	324	**99.3**	<u>98.9</u>	<u>98.9</u>	98.0	95.6
	yes		277	176	**98.7**	82.1	80.6	82.8	<u>91.6</u>
Turn direction	left		184	144	**97.6**	<u>94.2</u>	85.1	82.0	87.5
	right		172	156	<u>94.2</u>	**95.4**	85.7	85.7	84.5
	straight		144	200	**97.7**	<u>94.5</u>	85.8	88.7	76.5
Lateral acceleration	no		192	441	93.0	**98.1**	<u>94.8</u>	94.0	91.6
	yes		308	59	**99.5**	91.0	85.6	88.8	<u>91.6</u>
Brake then turn	no		221	435	**99.5**	<u>97.9</u>	96.0	96.5	93.1
	yes		279	65	**93.3**	<u>91.0</u>	84.6	82.0	86.6
Motion axis	lateral	D	173	70	**88.1**	<u>79.0</u>	60.9	67.9	68.3
	longitudinal	C	170	344	96.5	96.7	<u>98.4</u>	96.3	**99.2**
	none		157	86	92.6	**97.5**	<u>94.2</u>	93.0	90.1
Extreme maneuver	no		249	431	**100.0**	**100.0**	**100.0**	**100.0**	<u>99.6</u>
	yes		251	69	91.2	**96.6**	90.6	<u>94.7</u>	91.2

Table 28: Speed-peak-half `no_peak` recall below 4 m/s: 175 benchmark clips (119 CARLA, 56 nuScenes), of which 172 are below 0.5 m/s. Brackets give paired parent-scene bootstrap 95% intervals. Pooled recall weights clips. Bold/underlined: best/second point estimate per column across six recipes, with ties. Ranks are descriptive. Percentages.

Recipe	CARLA ($n = 119$)	nuScenes ($n = 56$)	Pooled ($n = 175$)
Decimal	95.0 [89.8, 98.9]	**98.2** [93.0, 100.0]	96.0 [92.1, 98.9]
xVal	**100.0** [100.0, 100.0]	**98.2** [93.0, 100.0]	**99.4** [98.0, 100.0]
CADE-style	95.8 [90.7, 99.3]	94.6 [79.1, 100.0]	95.4 [90.2, 99.3]
Staged	10.9 [4.1, 19.0]	1.8 [0.0, 6.7]	8.0 [3.2, 14.1]
Joint	42.0 [30.8, 53.8]	17.9 [4.3, 36.7]	34.3 [24.8, 45.1]
Single MLP	<u>98.3</u> [95.5, 100.0]	<u>96.4</u> [86.0, 100.0]	<u>97.7</u> [94.4, 100.0]

Table 29: Input interventions on the earlier staged Qwen checkpoint (93.82 development accuracy): 16,800 questions, including 4,800 temporal questions. State changes at inference, with vision also removed in no-video rows. Bold/underlined: best/second per column across conditions, with ties. Ranks are descriptive. Shading shows degradation from intact input. Percentages.

Condition	Semantic			Temporal	
	Acc.↑	BAcc.↑	F1↑	Acc.↑	F1↑
Intact and input controls					
Full model (VD)	**93.8**	**90.8**	**91.2**	92.0	89.6
Dynamics only (no video)	93.5	90.5	91.0	**92.5**	**89.8**
Floor (no video, no dynamics)	55.6	32.3	20.2	53.6	28.2
Channel removal: zero *channel(s) set to* **0**					
Zero all	55.5	33.1	21.8	52.3	29.9
Zero x	90.0	86.0	86.3	90.9	88.2
Zero y	88.7	84.1	84.0	90.7	87.9
Zero v_x	93.2	89.7	90.4	91.6	89.2
Zero v_y	93.3	90.0	90.6	92.1	**89.8**
Zero ψ (heading)	88.0	81.6	81.3	91.0	85.0
Zero accel	78.3	66.1	63.2	61.0	43.6
Zero jerk	90.5	83.1	83.4	90.4	87.2
Zero position (x, y)	81.1	76.6	74.2	87.6	82.7
Zero velocity (v_x, v_y)	92.3	88.3	89.1	91.6	89.2
Zero accel+jerk	72.5	55.8	48.8	53.2	31.3
Channel removal: mean *channel(s) → train per-timestep mean*					
Mean all	58.2	34.3	21.7	57.1	24.3
Mean x	90.1	86.1	86.4	90.9	88.3
Mean y	88.8	84.3	84.2	90.7	87.9
Mean v_x	93.2	89.7	90.4	91.6	89.2
Mean v_y	93.3	90.2	90.7	92.1	**89.8**
Mean ψ (heading)	88.2	82.0	81.9	90.8	84.9
Mean accel	76.2	62.1	56.9	57.9	39.5
Mean jerk	90.4	83.2	83.7	89.9	86.4
Mean position (x, y)	81.2	76.9	74.6	87.6	82.8
Mean velocity (v_x, v_y)	92.4	88.5	89.2	91.6	89.3
Mean accel+jerk	71.8	53.9	45.8	53.3	28.2
Temporal Structure *whole-trajectory: shuffle / time-reverse / collapse T*					
Shuffle (permuted T)	87.3	81.0	80.8	82.4	80.3
Reverse (time-flipped)	81.8	76.1	75.1	75.5	75.6
Mean-token (collapse T)	82.0	70.8	69.6	79.4	75.4
Input Noise *additive Gaussian $x_c \mathrel{+}= \sigma \cdot \mathrm{std}_c \cdot \epsilon,\ \epsilon \sim \mathcal{N}(0,1)$*					
Noise σ=0.1	93.0	89.3	89.6	90.8	88.0
Noise σ=0.25	87.1	80.2	78.8	86.0	82.8
Noise σ=0.5	78.0	68.6	64.4	76.3	72.2
Noise σ=1.0	62.5	53.9	45.8	58.6	54.8
Noise σ=2.0	43.1	42.0	31.3	41.2	36.2

Degradation vs. full model [pp]



92.3 or 92.4. Related quantities remain available through other channels, so small changes need not mean that a physical quantity is unimportant. A rule-consistency check also collapses when all clip-specific state is replaced.

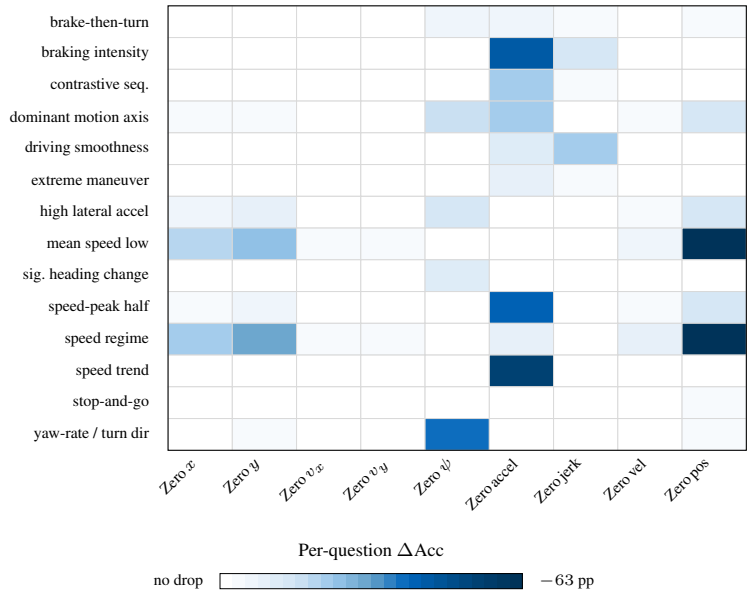


Figure 5: Per-question accuracy loss under channel removal. Rows are question types, columns are individual or paired state channels, and darker cells indicate larger drops. The selective pattern maps heading, acceleration, jerk and position to the questions they physically govern.

Figure 5 shows concentration by question: jerk removal has a modest aggregate effect but its largest effect is on driving smoothness.

I.2 Temporal dependence across encoders

This subsection supports Section 4.4's question-specific temporal-collapse result with Tables 30 and 31. Shuffling preserves values but changes arrangement, costing the Transformer 6.5 points. Reversal and averaging cost more. State removal leaves 55.54 versus a 55.64 no-input floor for the Transformer, and 56.70 versus 56.76 for the recurrent encoder. The columns compare each checkpoint against itself, with clean development accuracies 93.82 and 93.80, respectively. Temporal averaging costs the recurrent encoder nearly twice as much, despite similar clean accuracy.

Table 30: Input interventions on two Qwen encoder recipes over 16,800 development questions. Only the named input changes. The separately trained no-state control appears last. No such recipe identifies a missing trained counterpart. Percentages.

Condition	Transformer	xLSTM-inspired
Full model	93.8	93.8
Trajectory zeroed	55.5	56.7
Video zeroed	93.5	93.6
Mean-token (time collapsed)	82.0	74.1
No-input floor	55.6	56.8
Trajectory zeroed at training (interface recipe)	78.8	*no such recipe*

Where the order dependence sits. On speed peak half, zeroing state costs 56.2 points (37.2 versus 33.0 with video blanked and state zeroed), whereas replacing timesteps by the clip mean costs only 4.2 (89.2 remaining). The repeated means of position and derivative channels still provide aggregate motion cues, and video remains available. This establishes a sufficient route under changed inputs, without identifying the intact model's computation. Contrastive sequence instead loses 30.4 points under permutation and 51.8 under reversal. High question-level accuracy therefore need not imply dependence on the supplied temporal arrangement.

Table 31: Question-level input interventions on the earlier staged checkpoint: 16,800 development questions, 1,200 per type. Floor blanks video and state, while Zero removes state only. Reverse and Shuffle change temporal order. Mean-token repeats each clip mean over sixteen positions. Bold marks the intact-input comparison, not rank. Rows follow intact-minus-zero accuracy. Percentages.

Question type	n	**Floor**	**Zero**	**Reverse**	**Shuffle**	**Mean-token**	**Intact**
Speed regime	1200	9.7	13.2	51.6	79.8	71.8	**94.8**
Motion axis	1200	21.4	21.4	86.4	87.5	86.7	**91.4**
Low mean speed	1200	34.6	34.6	68.9	91.8	84.8	**96.8**
Speed peak half	1200	33.0	37.2	84.0	89.2	89.2	**93.4**
Braking intensity	1200	38.3	38.3	85.8	84.5	60.9	**93.0**
Speed trend	1200	41.8	41.8	93.7	93.2	96.1	**96.5**
Turn direction	1200	39.8	39.8	78.8	76.2	83.6	**87.5**
Contrastive sequence	1200	52.0	42.7	31.2	52.6	41.8	**83.0**
Driving smoothness	1200	69.2	69.2	93.2	90.3	69.2	**95.1**
Lateral acceleration	1200	78.8	78.8	88.2	89.7	89.2	**93.7**
Heading change	1200	82.9	82.9	93.7	95.2	94.1	**95.6**
Brake then turn	1200	87.4	87.4	93.2	94.6	90.6	**95.0**
Extreme maneuver	1200	91.1	91.1	97.9	98.1	91.2	**98.4**
Stop-and-go	1200	99.0	99.0	99.1	99.4	99.5	**99.4**
Overall	16,800	55.6	55.5	81.8	87.3	82.0	**93.8**

Stop-and-go imbalance. Zeroed and intact accuracies, 99.0 and 99.4, lie near the 99.0 development majority floor (Appendix H.2).

I.3 Projected-block rank

Table 32 supports the distinction between rank truncation and a smaller trained interface. It reduces the projected block's rank while preserving sixteen positions and sequence length. Rank eight costs the earlier Transformer checkpoint 6.9 points, while the recurrent checkpoint is close to its full-rank accuracy at rank four. These are inference-time truncations, not training at fewer state positions.

Table 32: Inference-time SVD rank sweep of the projected sixteen-position block: 16,800 development questions per cell, with sequence length fixed. Bold marks the full-rank comparison, not rank. The final column gives the smallest rank with loss at most 0.46 points. Percentages.

Encoder	r=1	r=2	r=4	r=8	r=16	**Smallest** r (loss $\leq$ 0.46 pp)
Transformer	70.7	75.0	79.1	86.9	**93.8**	none below 16
xLSTM-inspired	88.0	93.2	93.7	93.7	**93.8**	4
MLP	73.1	78.3	88.2	92.0	**92.0**	8
Conv1D	74.4	80.5	87.5	90.0	**90.1**	8

Loss is $\mathrm{Acc}_{16} - \mathrm{Acc}_r$. The 0.4583-point cutoff is the staged-adapter seed range on development, not an equivalence margin.

J Generation sanity check

This section complements Section 3.4's option-log-likelihood accuracy with generated answers.

Matched panel. Table 33 compares decimal text, xVal, four state-adapter encoder recipes and frozen random on the same 2,100 development questions, 150 per type. Caption metrics use the same 570 clips and holistic captions. The panel excludes LLMTime, joint continuation and the CADE-style adapter. Appendix D defines its metrics. No-state generation is omitted because its evaluation used a mismatched input pathway, separately from the multiple-choice control.

Numerical fidelity. Numeric Hit is 85.9 percent for text and 66.7 for the staged adapter: these are mean per-rationale tolerance-hit fractions, not fractions of entirely correct rationales. Text exposes

Table 33: Generation and caption diagnostics for seven recipes: 2,100 rationale records (150 per question type), 570 unique caption references, and 450 records for Meas. L1. Definitions are in Appendix D. Ans., Hit and Tok. are percentages, Fmt is a fraction, L1 values mix source units, and PPL is dimensionless.

Recipe	n	CoT					Caption	
		Ans.↑	Fmt	Num. L1↓	Hit↑	Meas. L1↓	PPL↓	Tok.↑
Decimal text	2100	98.3	1.00	0.54	85.9	0.05	1.41	89.8
xVal-style scalar embeddings	2100	96.2	1.00	0.64	72.4	0.21	1.40	88.6
Staged adapter, Transformer	2100	93.2	1.00	0.64	66.7	0.23	1.59	88.4
Staged adapter, xLSTM-inspired	2100	91.9	1.00	0.62	64.8	0.26	1.54	88.3
Staged adapter, MLP	2100	89.7	1.00	0.85	59.7	0.34	1.43	88.1
Staged adapter, Conv1D	2100	87.7	1.00	0.87	61.9	0.48	1.47	87.4
Frozen random pathway	2100	91.3	1.00	0.84	60.7	0.31	1.47	87.0

numerical strings that can be copied, but targets also include derived quantities and thresholds. Measured-value L1 covers 450 records from three question types and ranges from 0.23 to 0.48 across the four trained state-adapter encoders (0.05 for decimal text). A nearest match can be a threshold rather than the intended measurement. Caption columns measure teacher-forced likelihood and token accuracy, not generated-caption factuality.

K THRESHOLD SUPERVISION AUDIT

This section supports Section 3.4's supervision scope by quantifying decision rules stated in training targets. It does not report benchmark examples entering training. The concern is that explicit numerical thresholds might interact with an interface that exposes decimal values directly. The audit uses the earlier staged checkpoint, so its margins differ from the selected-checkpoint comparison.

The reasoning supervision states the thresholds. Every benchmark label is produced by a deterministic oracle that applies fixed thresholds to the kinematic state. The chain-of-thought targets used in Stage 3 verbalise those thresholds. Auditing all 56,723 reasoning targets, 63.8 percent state a numeric threshold explicitly, and the rate is above 94.8 percent on nine of the fourteen question types. On the remaining five it is at or below 7.9 percent, and on `stop_and_go` and `brake_then_turn` it is exactly zero. A trained model can therefore learn the decision boundaries themselves rather than only the mapping from state to label. This motivates checking whether explicit threshold supervision is associated with a larger text advantage.

Threshold disclosure and the text interface. Both main interfaces receive the same reasoning corpus with task-sampling probabilities 0.4 multiple choice, 0.3 caption and 0.3 reasoning. This matches exposure but does not exclude an interaction between rationale content and numerical representation. An exploratory comparison across question types finds a text advantage of 5.61 points on the nine types whose rationales usually state thresholds, versus 10.48 on five threshold-silent types, using the earlier adapter checkpoint. The groups differ in difficulty. After headroom normalization, the reported permutation test gives $p = 0.837$. This supplies no evidence of a simple across-type association but is not an ablation of threshold disclosure and cannot rule out that interaction.

What the multiple-choice metric does and does not see. The headline metric compares teacher-forced log-probabilities of the supplied options, so the model is never sampled from and no reasoning trace is produced or scored. The thresholds verbalised in training rationales are supervision, not additional information supplied at test time.

The limit that remains. The labels are deterministic functions of kinematic state and the task taxonomy is shared with training. This makes the benchmark useful for state-usability comparisons while limiting conclusions about new tasks, general driving reasoning or multimodal integration. The largest text advantages include contrastive-sequence and speed-peak-half questions, which compare timesteps. These are still deterministic state computations. Temporal interventions establish

Table 34: Unadapted Qwen3-VL-2B-Instruct with video and questions, scored by option log probability on 14,000 benchmark questions. No state values or trained adapter weights are supplied. Source and correct-answer counts are shown. This setting differs from both the trained no-state control and unadapted text conditioning. Percentages.

Evaluation population	Correct	Questions	Accuracy (%)
EgoDyn-Bench	6,159	14,000	43.99
CARLA	2,769	7,000	39.56
nuScenes	3,390	7,000	48.43

sensitivity to arrangement on the evaluated checkpoint, not general-purpose reasoning or causal correctness.

L UNADAPTED AND EXTERNAL REFERENCES

This section supports the introduction's comparison with zero-shot systems and reports an unadapted baseline.

Unadapted video-only. Qwen3-VL-2B-Instruct receives video, question and options without numerical state or active LoRA updates (all 196 LoRA-B tensors are zero). It scores 43.99% on 14,000 benchmark questions: 39.56% CARLA and 48.43% nuScenes (Table 34). Trained no-state instead includes a learned constant state prefix. Neither comparison isolates adaptation for decimal-text inputs.

External systems. Table 35 places the trained state adapter's 88.6 beside zero-shot frontier models receiving trajectory text (65.6 to 75.8) and a rule-based optical-flow system (70.3). Models, training and serialization differ. Geometric methods cover only six of fourteen types and are excluded from ranking.

On the same six types ($n = 6{,}000$), the trained no-state recipe scores 61.2 versus RAFT's 70.3 (9.1 points), while the staged adapter and decimal text score 89.5 and 96.8.

M PRECISION, RESAMPLING AND TOKEN BUDGET

This section supports Section 3.2's rounding statement and reports a supplementary eight-position variant.

M.1 PRECISION AND RESAMPLING SENSITIVITY

This subsection supports Section 3.2's rounding statement by separating interface errors from changes caused by rounding or resampling.

Sensitivity to decimal precision. The in7 rendering uses steps 0.1 for position and velocity, 0.001 for heading and 0.01 for acceleration and jerk. A diagnostic flags items whose recorded decision statistic lies within a proposed rounding-error bound of a decision threshold. This identifies potential sensitivity, not a proof that the rounded input cannot support the correct label. In particular, a possible worst-case perturbation does not imply that the actual rendering changes the label.

Elapsed time is computed from array timestamps. The five scalar-rule checks flag 58 items: 32 turn-direction, ten braking-intensity, nine lateral-acceleration, six speed-trend and one mean-speed item. Removing these 58 leaves text at 96.1555 and the staged adapter at 88.7391, a 7.4164-point gap versus 7.4286 on the full benchmark. These are sensitivity estimates under the implemented bounds, including a compound lateral-acceleration bound, rather than certified information ceilings.

The broader stop-and-go and speed-peak-half checks additionally flag 339 and 247 items. Removing all 644 flags leaves text at 96.5933 and the staged adapter at 90.9479, a 5.6454-point gap. Those checks are conservative approximations. The speed-peak-half oracle also has a near-zero branch, so

Table 35: *EgoDyn-Bench* accuracy: 14,000 questions for zero-shot and trained systems. External rows are from Schäfer et al. (2026). Geometric systems use only the six-type subset (6,000 questions) and are excluded from ranking. Bold/underlined: best/second across the full-benchmark rows. Ranks are descriptive. Percentages.

Model	Size	Trajectory input	Acc.↑
Geometric baselines	*optical-flow/visual-odometry kinematics + rules*		
RAFT-flow heuristic	*n/a*	video only	70.3
VO-proxy	*n/a*	video only	65.1
Zero-shot VLMs	*trajectory text in the prompt*		
Gemini 3 Pro (preview)	*n/a*	video + traj. as text	75.8
GPT-5.1	*n/a*	video + traj. as text	71.9
Claude Sonnet 4.5	*n/a*	video + traj. as text	71.5
Qwen3-VL-8B	8B	video + traj. as text	69.7
InternVL3.5-8B	8B	video + traj. as text	65.6
Trained text interfaces	*same Stage-3 objective, matched 20-epoch cap and patience*		
Qwen3-VL-2B + LoRA, decimal text	2B	Decimal text, 16×7	**96.0**
Qwen3-VL-2B + LoRA, LLMTime-style (Gruver et al., 2023)	2B	LLMTime-matched, 16×7	<u>94.8</u>
Trained scalar and adapter interfaces	*LoRA-adapted backbone, state delivered as embeddings rather than text*		
Qwen3-VL-2B + staged adapter (ours)	2B	K=16 positions	88.6
Qwen3-VL-2B + xVal-style (Golkar et al., 2024)	2B	scalar embeddings, 16×7	92.2
Qwen3-VL-2B + joint continuation (ours)	2B	K=16 positions, encoder/projector/LoRA joint	90.0

GPT-5.1 scored 13,734 of 14,000 questions. Other non-geometric rows cover all questions. The six-type geometric subset contains 6,000. Zero-shot text formats differ from the trained formats named in the table.

Table 36: Accuracy at two sampling resolutions and state-position budgets: 16,800 development and 14,000 benchmark questions. T and K change together, with separate preparation. Epoch 3 is the selected $K = 8$ checkpoint and epoch 7 is supplementary. Percentages.

Recipe / checkpoint	Epoch	T	K	Dev. (%)	Bench. (%)
K=16 staged adapter, selected	9	16	16	94.39	88.57
K=8, selected	3	8	8	91.79	84.43
K=8, supplementary	7	8	8	92.32	85.89

a relative-prominence bound alone is not a complete model of its label sensitivity. Seven of fourteen types are outside this bound's coverage.

Resampling fidelity. Applying the exact released rules to all benchmark raw arrays reproduces every speed-peak-half and stop-and-go gold label. Reapplying them to stride-two scalar speed samples agrees with gold on 99.4 and 99.7 percent, respectively. These plug-in checks do not give optimal information ceilings or cover every question, but they argue against attributing the large interface gap on these tasks to temporal decimation alone.

M.2 Eight-position budget

This subsection reports a supplementary smaller state budget (Table 36). Endpoint-aligned interpolation reduces the 31-sample history to eight points over the same three seconds, with $K = T$ after preprocessing. The selected epoch 3 scores 91.79% development and 84.43% benchmark, versus 94.39% and 88.57% at sixteen positions: deficits of 2.60 and 4.14 points. At $K = 8$, diagnostic-class mean recall falls 10.5 points versus 2.3 for comparison classes. Separate preparation and coarser sampling prevent attributing these changes solely to output length (Table 4).

Table 37: InternVL3.5-2B adapter accuracy: 16,800 development questions over 1,200 clips and 14,000 benchmark questions over 1,000 clips. Encoder sizes are stored parameter counts. The epoch-3 row is development-only. Bold: highest displayed accuracy per column, descriptive. Percentages.

InternVL recipe	Encoder	Development↑	EgoDyn↑
State adapter (K=16), xLSTM-inspired, selected (epoch 5)	11.03M	**93.64**	**87.44**
State adapter (K=16), xLSTM-inspired, epoch 3	11.03M	93.19	*n/a*
State adapter (K=16), Transformer, selected (epoch 6)	3.67M	90.04	81.89

Table 38: InternVL preparation and downstream metrics for two encoder recipes. Stage 1 uses a temporary reconstruction head and a common fixed validation scale. Stage 2 uses selected caption checkpoints. Downstream evaluation covers 16,800 development and 14,000 benchmark questions. Bold: best per column across two recipes, descriptive. ADE is in metres, HE in radians, CE in nats/token and accuracies in percent.

Encoder	Stage 1		Stage 2		Stage 3	
	ADE (m)↓	HE (rad)↓	CE↓	Tok. (%)↑	Dev. (%)↑	Bench (%)↑
Transformer	0.977	0.0130	0.7189	79.1	90.0	81.9
xLSTM-inspired	**0.556**	**0.0118**	**0.4850**	**83.8**	**93.6**	**87.4**

The supplementary epoch 7 scores 92.32% and 85.89%. Epoch 3 remains primary because epoch 7's sampled-selection improvement did not clear the 0.005 gate. The two evaluations used different RTX 5090 workstations. Larger budgets and longer histories are untested. At fixed sample rate, both the current adapter and text grow with the horizon.

N SECOND BACKBONE: INTERNVL3.5-2B

This section supports Sections 4.3 to 4.5 by testing state-adapter concept mapping and the low-speed failure on a second backbone.

Downstream accuracy. Table 37 reports the two selected InternVL3.5-2B-HF checkpoints: epoch 5 for xLSTM-inspired and epoch 6 for Transformer. The xLSTM-inspired recipe reaches 93.64% development and 87.44% benchmark accuracy, versus 90.04% and 81.89% for Transformer. Its advantages are 3.60 development points (paired parent-scene 95% interval $[2.93, 4.29]$) and 5.55 benchmark points ($[4.73, 6.41]$), using the procedure in Appendix D. Both benchmark sources agree in direction.

Preparation metrics. Table 38 pairs preparation and downstream metrics, as Table 21 does for Qwen. Transformer selects Stage-2 epoch 56 (caption CE 0.7189, token accuracy 79.1%), while xLSTM-inspired selects epoch 49 (0.4850, 83.8%). Checkpoint selection requires a relative loss improvement above 0.001, so the selected value need not be the global minimum. Reconstruction is evaluated with a common fixed development scale, rather than the changing Stage-1 EMA scale. These values are not comparable to Qwen's training-time values. The recurrent encoder has lower error on every channel group, including ADE 0.556 versus 0.977 m, and higher downstream accuracy. On Qwen, it reconstructs about four times more accurately than Transformer but has lower downstream accuracy: reconstruction quality has no fixed relationship to downstream accuracy across these two comparisons.

State removal. Zeroing the state at inference on the same 16,800 development questions reduces xLSTM-inspired accuracy to 56.87%, 7.32 points below the 64.19% per-type majority floor, whereas Transformer scores 69.17%, 4.98 points above it (Table 39). The more accurate intact recipe thus depends more strongly on the state pathway. Transformer's remaining accuracy is concentrated outside the four temporal types: on those types it scores 54.31%, below xLSTM-inspired's 57.65%. Video remains available, and zeroing changes the input distribution, so this measures input dependence without separating video cues from label and question priors.

Table 39: State removal from the selected InternVL checkpoints on 16,800 development questions, including 4,800 temporal questions. Each trained checkpoint receives intact or zeroed state with video present. Shading shows degradation from each full model. Percentages.

Condition	Semantic			Temporal	
	Acc.↑	BAcc.↑	F1↑	Acc.↑	F1↑
xLSTM-inspired (epoch 5)					
Full model (VD)	93.6	90.4	90.6	92.6	90.2
Zero all	56.9	30.5	19.7	57.6	26.0
Transformer (epoch 6)					
Full model (VD)	90.0	84.8	85.1	87.6	84.5
Zero all	69.2	51.7	45.2	54.3	34.8

Degradation vs. full model [pp]

no drop — larger degrad.

Table 40: InternVL benchmark accuracy: 14,000 questions, 1,000 per type and 7,000 per source. Types follow the recurrent encoder advantage. Floor is the majority share. CARLA/nuScenes accuracy is 88.9/86.0 for xLSTM-inspired and 84.3/79.5 for Transformer. The corresponding CARLA minus nuScenes gaps are 2.96 and 4.74 points. Encoder capacity also differs. Accuracies are percentages and Δ is percentage points.

Question type	xLSTM-inspired	Transformer	Δ	Floor
Driving smoothness	92.0	76.0	+16.0	42.4
Stop-and-go	88.1	74.6	+13.5	68.0
Contrastive sequence	69.7	58.7	+11.0	33.4
Speed regime	88.0	77.3	+10.7	31.9
Turn direction	81.8	73.8	+8.0	34.4
Low mean speed	95.7	90.5	+5.2	54.7
Braking intensity	85.1	80.0	+5.1	34.5
Heading change	88.8	85.1	+3.7	54.7
Lateral acceleration	90.2	87.7	+2.5	63.3
Motion axis	89.7	88.0	+1.7	51.4
Brake then turn	91.0	89.3	+1.7	65.6
Speed peak half	78.6	77.3	+1.3	37.4
Extreme maneuver	92.5	93.0	-0.5	68.0
Speed trend	92.9	95.1	-2.2	33.5

Question types and sources. Table 40 reports the question-level comparison and source accuracies. The recurrent encoder leads by 16.0 points on driving smoothness and 13.5 on stop-and-go. The recipes are within 2.2 points on extreme maneuver and speed trend. Both under-predict the no-event class (9.7 and 8.8%, versus 27.8% true), and 94 to 100% of errors on braking intensity, speed regime and speed trend fall on adjacent classes.

Low-speed flat profiles. Table 41 uses the same clips and speed bins as Table 25. Below 4 m/s, `no_peak` recall is 17.7% for xLSTM-inspired and 15.4% for Transformer, versus 64.1% and 61.5% at 8 to 14 m/s. These two InternVL checkpoints also exhibit low-speed failures, while the Qwen staged runs vary widely on this subgroup (Table 24). The bins contain different clips and do not hold trajectory shape or the oracle's relative decision rule fixed. The highest-speed bin contains only 26 clips.

Encoder capacity. InternVL's xLSTM-inspired encoder has 11.03M parameters versus Qwen's 2.90M, with widths 512 and 256 per direction. InternVL also clips input/forget gate preactivations above 15 (Appendix A.1). The Transformer encoders have 3.67M and 3.15M stored parameters, with different feed-forward widths, activations and output normalization (one output LayerNorm for InternVL, two for Qwen). Excluding 503,808 unused positional parameters from each Transformer

Table 41: InternVL recall on the benchmark speed-peak-half `no_peak` class: the same 278 clips and speed bins as Table 25, with bin counts shown. Bins contain different clips and do not isolate numerical resolution. Percentages.

Clip max speed (m/s)	n	xLSTM-inspired	Transformer
< 4	175	17.7	15.4
4 to 8	38	55.3	57.9
8 to 14	39	64.1	61.5
> 14	26	73.1	46.2

gives approximately 3.17M and 2.65M active parameters. The recurrent encoder is then larger on both backbones, so the accuracy reversal does not follow one capacity ordering.

Scope. The development-to-benchmark gaps are 6.21 points for xLSTM-inspired and 8.16 for Transformer, but capacity, gate clipping, reconstruction-head pooling, Stage-1 scaling and selection sampling differ across backbones. The trained-interface ranking is Qwen-specific because matched InternVL text and trained no-state comparisons are absent. Joint continuation, resource accounting, timing and generation are also Qwen-only. InternVL supports adapter accuracy, preparation metrics, state removal and the low-speed analysis.

O EVIDENCE COVERAGE AND SINGLE-RUN LIMITS

This section supports Section 4.5's scope of inference across training runs, backbones and evaluation modes.

Training runs. Three staged-adapter seeds and three state-only MLP seeds are reported. The staged runs vary substantially on the identified low-speed subgroup (Table 24). Other interface and encoder recipes generally have one training run, so their relative training-run stability is unmeasured. Paired intervals condition on checkpoints, not retraining variation. Joint continuation is one additional training run and combines unfreezing with extra optimization.

Coverage. The central interface ranking, continuation, resource measurements and seven-recipe development generation panel concern Qwen. InternVL supplies two adapter configurations and low-speed diagnostics, without a matched trained-text comparison. The unadapted video-only control changes both adaptation and numerical input relative to trained text. Fixed-shape timing does not measure serving throughput, and development generation does not establish benchmark generation performance.

Evidence access. The supplementary CPU replay reproduces saved-prediction statistics, not training or inference. `ARTIFACTS.md` documents provenance and excluded analyses.