DRIVEVLA-CACHE: TRAINING-FREE ACCELERATION OF VISION-LANGUAGE-ACTION MODELS FOR AUTONOMOUS DRIVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning vision–language–action (VLA) models show strong performance in autonomous driving, but inference latency makes real-time deployment challenging. The cost spans four stages: visual encoding repeatedly processes overlapping camera frames; multimodal prefill recomputes shared context; autoregressive decoding often regenerates identical reasoning text; and trajectory generation repeatedly evaluates the action network over successive integration steps. Reasoning decoding contributes the largest share in our profiling, but accelerating it alone leaves substantial computation in the remaining stages. We propose DRIVEVLA-CACHE, a training-free framework that integrates Reasoning Recontextualization with visual feature, prompt K/V, and velocity-field caching to accelerate the full VLA inference pipeline. Visual feature caching avoids re-encoding overlapping frames, while selective prompt key–value (K/V) reuse reduces repeated context computation. For reasoning, the central challenge is that reusable text has cached K/V states tied to earlier observations. Our Reasoning Recontextualization periodically re-encodes retained reasoning text under the current observations in one forward pass, updating the states used for action generation without repeating autoregressive decoding. Finally, velocity-field caching reuses or extrapolates recent fields between selected action-network evaluations, reducing computation while preserving all integration updates. Together, these mechanisms accelerate the pipeline while producing a new trajectory every planning cycle. On Alpamayo 1.5-10B across 913 closed-loop ALPASIM scenes, DRIVEVLA-CACHE achieves the highest mean driving score and lowest latency among evaluated methods. It delivers 3.5× driver-inference and 2.4× end-to-end speedups over the full-computation baseline, while increasing mean AlpaSim Score from 70.8 to 73.4.

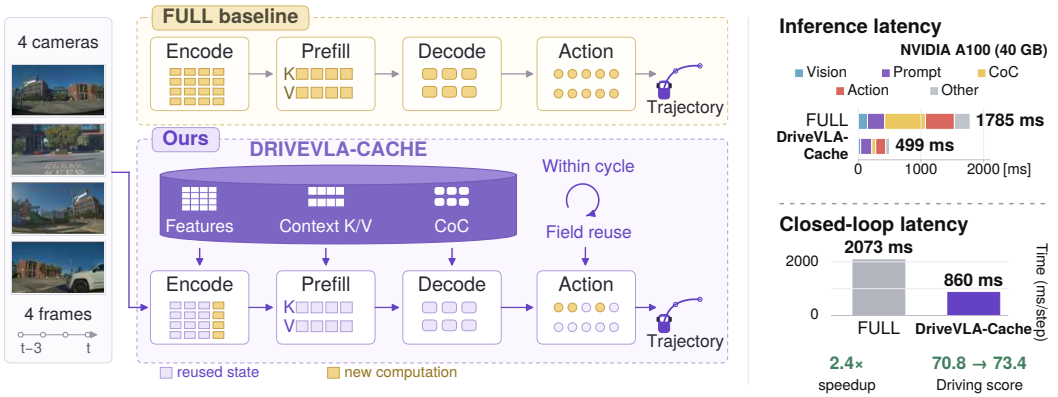


Figure 1: **DRIVEVLA-CACHE accelerates reasoning VLAs while preserving closed-loop driving quality. Left:** Slow–fast inference overview. **Top right:** Single-scene inference latency breakdown on Alpamayo 1.5. **Bottom right:** A 2.4× end-to-end latency speedup with improved driving score.

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

1 INTRODUCTION

Reasoning vision–language–action (VLA) models are emerging as a powerful paradigm for autonomous driving by unifying multi-camera perception, explicit reasoning, and trajectory generation to handle complex and long-tail scenarios (Fu et al., 2025; Kong et al., 2025; Zhou et al., 2025; NVIDIA et al., 2026). In models such as Alpamayo 1.5 (NVIDIA, 2026), autoregressively generated Chain-of-Causation (CoC) reasoning conditions the action expert through its contextual key–value (K/V) states. Their computational cost, however, makes real-time deployment challenging. For example, Alpamayo 1.5-10B takes an average of 1.8 s per driver inference in our evaluation (Table 1). This burden is shared by reasoning VLAs with the same architecture. Our profiling shows that CoC reasoning contributes the most latency, but visual encoding, multimodal prefill, and action generation also incur substantial costs, limiting the benefit of accelerating any single stage (Figure 7).

Closed-loop execution raises the bar further. Unlike in open-loop replay, approximation errors do not remain confined to a single prediction: changes in the predicted trajectory can affect the vehicle's subsequent state, observations, and planning decisions (Ross et al., 2011). For example, in the same scene, predicted-trajectory deviations from full computation reach approximately 0.8 m in open-loop replay and 4.4 m in closed-loop execution (Figure 3). The challenge is therefore to reduce inference latency while maintaining closed-loop driving quality.

Prior work has explored stage-specific acceleration. Visual-token pruning (Pei et al., 2026) and VLA-Cache (Xu et al., 2025) accelerate multimodal prefill, while BAC (Ji et al., 2026) accelerates flow-matching action generation through intermediate feature caching. However, reusing upstream representations can affect downstream action generation (Oi et al., 2026). For reasoning VLAs, combining these techniques must preserve how retained reasoning guides actions under changing observations.

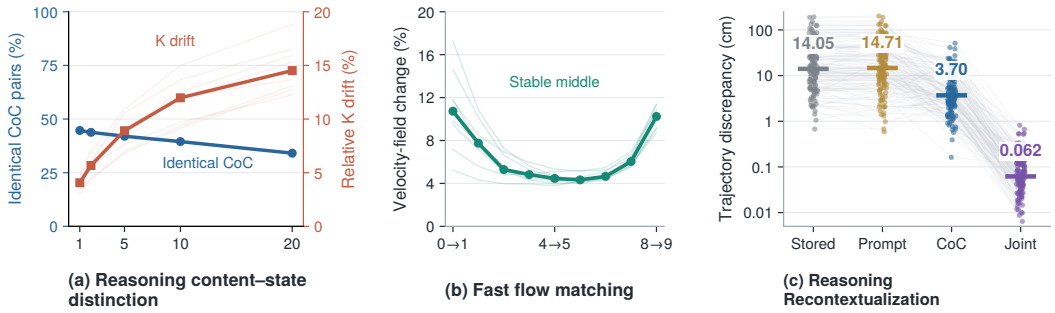
Can we jointly accelerate the reasoning-to-action pipeline while maintaining closed-loop driving quality? We propose **DRIVEVLA-CACHE**, a training-free inference framework built on a *slow–fast asymmetry around the action expert*. Its central insight is a **Reasoning Content–State Distinction**: reasoning content can persist across planning cycles even as its contextual K/V states need refreshing. **Reasoning Recontextualization** periodically rebuilds the current prompt and re-encodes retained CoC tokens, refreshing the state used for the current action prediction without autoregressive CoC decoding. Content renewal follows a separate schedule. The framework combines this mechanism with visual feature caching across planning cycles and velocity-field reuse within action generation, using four network evaluations for ten integration updates and producing a new trajectory every planning cycle.

On the slow reasoning side, we find that retained reasoning can be recontextualized for current action generation without regenerating its content. Across 125 comparisons from 19 scenes where retained and freshly generated CoC token sequences are identical, rebuilding the prompt alone leaves a median trajectory discrepancy of 14.71 cm; jointly rebuilding the prompt and re-encoding the same CoC tokens reduces it to 0.062 cm (Figure 2). With observations and action noise matched and all ten field evaluations retained, Joint yields lower trajectory discrepancy than both Prompt-only and CoC-only in all 125 comparisons. Its practical benefit is demonstrated in closed-loop evaluation: adding Reasoning Recontextualization to context/CoC reuse improves mean Score_{AF} by 1.7 points with 65 ms of additional end-to-end latency (Table 2).

On the fast action generation side, velocity-field caching provides a complementary source of acceleration, and the placement of fresh evaluations affects closed-loop driving quality. In our schedule ablation, distributing four fresh evaluations across early and late integration steps increases the mean driving score from 67.4 to 75.1 compared with concentrating four evaluations at the beginning, at comparable latency (Table 3b). These results motivate distributing the available network evaluations across the integration process. Our implementation reuses or extrapolates the intervening fields while retaining all integration updates.

Our contributions are:

- **Reasoning Content–State Distinction.** We show that reasoning-content regeneration and contextual-state refresh can be separated for inference reuse. Controlled comparisons with identical CoC token sequences demonstrate the importance of refreshing the action-



(a) Reasoning content–state distinction **(b) Fast flow matching** **(c) Reasoning Recontextualization**

Figure 2: **Motivation for slow–fast reuse. (Left)** CoC content often remains unchanged across planning cycles, while its contextual key states drift. **(Middle)** Velocity-field changes vary more across scenes at early integration steps than at later steps. **(Right)** For identical retained and freshly generated CoC sequences, Joint reduces median trajectory discrepancy from 14.05 to 0.062 cm compared with Stored, measured against the fresh-context reference.

conditioning state of retained reasoning, motivating Reasoning Recontextualization without repeated autoregressive decoding.

- **DRIVEVLA-CACHE.** We develop a training-free inference framework that reuses visual features and CoC content across planning cycles, refreshes the retained reasoning's contextual K/V through Reasoning Recontextualization, and reduces velocity-field evaluations within action generation while retaining all integration updates.
- **Closed-loop evaluation.** On 913 ALPASIM scenes, DRIVEVLA-CACHE achieves a $2.4\times$ end-to-end speedup and a $3.5\times$ driver-inference speedup on Alpamayo 1.5-10B, while increasing ALPASIM Score from 70.8 to 73.4 (Table 1). On Alpamayo 1, DRIVEVLA-CACHE achieves a $2.1\times$ end-to-end speedup while retaining 98.9% of the full-computation driving score.

2 RELATED WORK

VLA Models for Autonomous Driving. Early work connects language to driving through graph-structured reasoning and instruction-following control (Sima et al., 2025; Shao et al., 2024). Recent VLA models integrate perception, language understanding, and trajectory prediction (Renz et al., 2025; Kong et al., 2025; Fu et al., 2025). Reasoning adds latency to closed-loop control. Slow–fast systems coordinate a slower reasoner with a faster planner (Tian et al., 2025; Chen et al., 2024; Zhang et al., 2025), while adaptive reasoning learns when to skip intermediate reasoning (Zhou et al., 2025). CogDriver promotes temporal coherence through sparse temporal memory and distillation (Liu et al., 2026a). Alpamayo 1.5 couples Chain-of-Causation reasoning with a flow-matching action expert in a unified pipeline (NVIDIA et al., 2026; NVIDIA, 2026). We reduce repeated computation in this pipeline with fixed pretrained weights.

Efficient VLA Inference. VLA acceleration targets per-prediction costs and temporal redundancy. Adaptive depth and quantization (Yue et al., 2024; Xu et al., 2026), visual-token pruning (Pei et al., 2026), and compact action tokenization with blockwise decoding (Pertsch et al., 2025; Liu et al., 2026b) reduce per-prediction computation. Across control cycles, VLA-Cache reuses visual K/V (Xu et al., 2025), while SP-VLA predicts actions from recent history (Li et al., 2026). Within diffusion and flow-matching generation, feature and velocity caching reduce repeated network evaluations (Lyu et al., 2025; Liu et al., 2025a; Ji et al., 2026; Gao et al., 2026). TaylorSeer forecasts intermediate features using finite-difference Taylor expansions (Liu et al., 2025b). EfficientVLA combines language and visual pruning with diffusion-feature caching, integrating acceleration across modules (Yang et al., 2025). Context dependence poses a further challenge to K/V reuse: CacheBlend selectively recomputes cached K/V to restore cross-chunk attention in LLM serving (Yao et al., 2025). Our framework makes generated reasoning reusable across planning cycles by separating CoC renewal from periodic refresh of its action-conditioning K/V. This retains the savings from skipping autoregressive decoding while adapting the reasoning's contextual state

to current observations. Combined with visual and velocity-field caching, this mechanism enables training-free acceleration of closed-loop driving.

3 Problem Formulation

Reasoning VLA inference. At planning cycle t, input x_t is encoded into prompt K/V P_t; autoregressive CoC decoding produces tokens c_t and contextual K/V K_t, giving the action-conditioning context $z_t = (P_t, K_t)$. Here P_t and K_t denote the prompt and CoC partitions of the causal VLM K/V context consumed by the action expert. For initial action noise ξ_t, native flow-matching action generation (Lipman et al., 2023) computes

$$\begin{aligned} u_{t,0} &= \xi_t, \\ u_{t,k+1} &= u_{t,k} + \Delta\tau_k \, v_\theta(u_{t,k}, \tau_k; z_t), \qquad k = 0, \dots, N-1, \\ y_t &= D(u_{t,N}; x_t), \end{aligned} \tag{1}$$

where $u_{t,k}$ is the complete normalized action sequence at integration step k, N is the number of integration steps, τ_k is flow time, $\Delta\tau_k = \tau_{k+1} - \tau_k$, v_θ is the velocity-field predictor with frozen parameters θ, and D is the trajectory decoder. FULL denotes native inference that recomputes the current visual and prompt states, generates fresh c_t, K_t, and evaluates v_θ at all N integration steps.

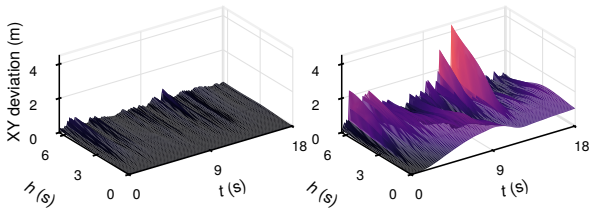


Figure 3: **Closed-loop execution can accumulate approximation errors across planning cycles.** Unlike open-loop replay, closed-loop predictions depend on states and observations shaped by earlier executed trajectories, allowing approximation errors to accumulate across planning cycles.

Closed-loop acceleration. Let ρ denote a computation-reuse policy and p_ρ the closed-loop rollout distribution induced by executing it. In open-loop replay, x_t is prescribed independently of ρ; in closed loop, the executed trajectory y_t affects subsequent inputs. For a T-cycle rollout ζ, let ℓ_t^ρ denote wall-clock inference latency at cycle t and $Q(\zeta)$ its driving-quality score. We seek

$$\min_\rho \; \mathbb{E}_{\zeta \sim p_\rho} \left[\frac{1}{T} \sum_{t=0}^{T-1} \ell_t^\rho \right] \qquad \text{s.t.} \qquad \mathbb{E}_{\zeta \sim p_\rho}[Q(\zeta)] \geq Q_{\mathrm{FULL}} - \epsilon, \tag{2}$$

where

$$Q_{\mathrm{FULL}} = \mathbb{E}_{\zeta \sim p_{\mathrm{FULL}}}[Q(\zeta)]$$

is FULL's closed-loop quality and $\epsilon \geq 0$ is the allowed degradation. The reuse policy ρ leaves the pretrained model and planning schedule unchanged. DRIVEVLA-CACHE constructs ρ through cross-cycle context reuse and velocity-field reuse within each planning cycle.

4 Method

DriveVLA-Cache constructs the reuse policy ρ in Eq. (2) along the native reasoning-to-action computation flow at two timescales. Across planning cycles, it reuses visual features and CoC content while refreshing the contextual states that condition the action expert. During flow-matching action generation, it reduces velocity-field evaluations while retaining the native integration grid and producing a new trajectory at every planning cycle. Figure 4 summarizes the computation reuse within one planning cycle.

At each planning cycle, the action expert uses either the partially updated prompt with stored CoC K/V or, on recontextualization cycles, the rebuilt prompt with re-encoded CoC K/V. Recontextualization supplies the context for the current prediction while retaining the stored reasoning snapshot. When CoC renewal occurs, valid fresh reasoning and its K/V replace that snapshot and condition the current action prediction; trajectory agreement determines the subsequent reuse interval.

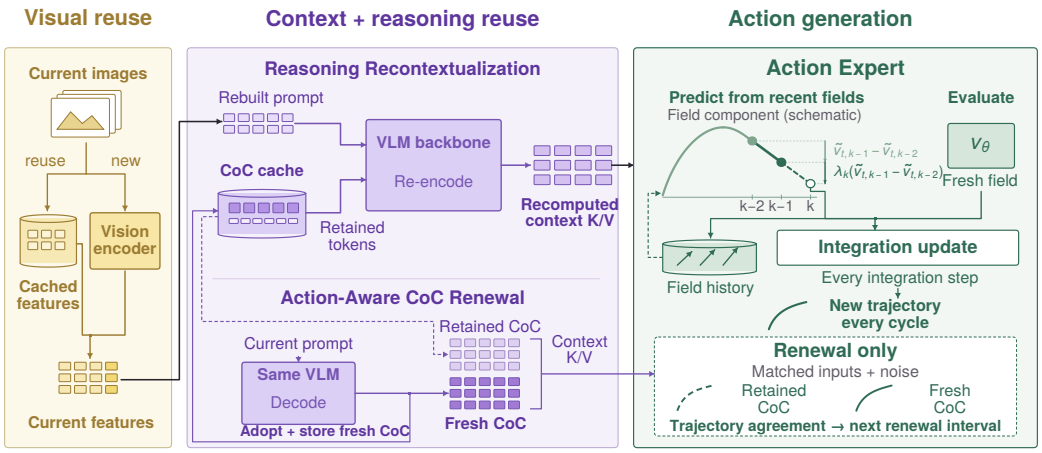


Figure 4: **DRIVEVLA-CACHE at one planning cycle. Slow computation reuse (left and middle):** reuse visual features and CoC content across planning cycles. Reasoning Recontextualization refreshes the contextual K/V of retained CoC tokens, while Action-Aware CoC Renewal updates the content separately. **Fast action generation (right):** generate a new trajectory using sparse velocity-field evaluations while retaining all integration updates.

4.1 SLOW CONTEXT CACHING ACROSS PLANNING CYCLES: VISUAL ENCODING AND COC REASONING

Source-identical images admit exact feature reuse before multimodal contextualization. CoC K/V, in contrast, depends on both the reasoning tokens and the prompt context under which they are causally processed. Unchanged content therefore does not guarantee that its contextual state remains suitable for the current input. With identical reasoning tokens, substituting retained CoC K/V changes the action expert's local attention readout under a fixed current query and attention background. The median readout difference grows with cycle gap in all eight examined scenes (Appendix B, Figure 9). We separate visual reuse, Reasoning Recontextualization, and Action-Aware CoC Renewal accordingly.

Visual Feature Caching. We memoize image-wise encoder outputs by source identity. Retained images reuse their cached representations, while only newly arriving images are encoded; cached and new features are assembled in the current window order before multimodal prefill. Exact reuse at the image-encoder boundary does not imply exact reuse of downstream prompt K/V: as the observation window advances, an unchanged image may occupy a different position and interact with updated prompt inputs.

CoC Caching and Reasoning Recontextualization. Let a be the latest successful CoC-renewal cycle, with CoC tokens c_a, the prompt state P_a used for that renewal, and stored contextual K/V

$$K_a = \mathcal{K}_\theta(c_a \mid P_a),$$

where $\mathcal{K}_\theta(c \mid P)$ denotes the contextual K/V obtained by causally processing known tokens c after prompt P.

On carry cycles, we construct P_t^{carry} by recomputing K/V for all non-visual prompt tokens and visual tokens from the newest frames and selected historical frames, while reusing the remaining overlapping visual K/V (Appendix C). The action expert uses P_t^{carry} with stored K_a, avoiding both fresh CoC decoding and re-encoding of the retained reasoning. However, K_a was computed under the earlier prompt P_a and need not remain suitable for the current input.

To refresh this contextual state without regenerating the reasoning content, **Reasoning Recontextualization** rebuilds the current prompt and re-encodes the retained CoC:

$$K_{a\to t} = \mathcal{K}_\theta\big(c_a \mid P_t^{\text{full}}\big), \tag{3}$$

where P_t^{full} is the prompt rebuilt from the current input. Because c_a is known, Eq. (3) requires one causal forward pass rather than autoregressive decoding.

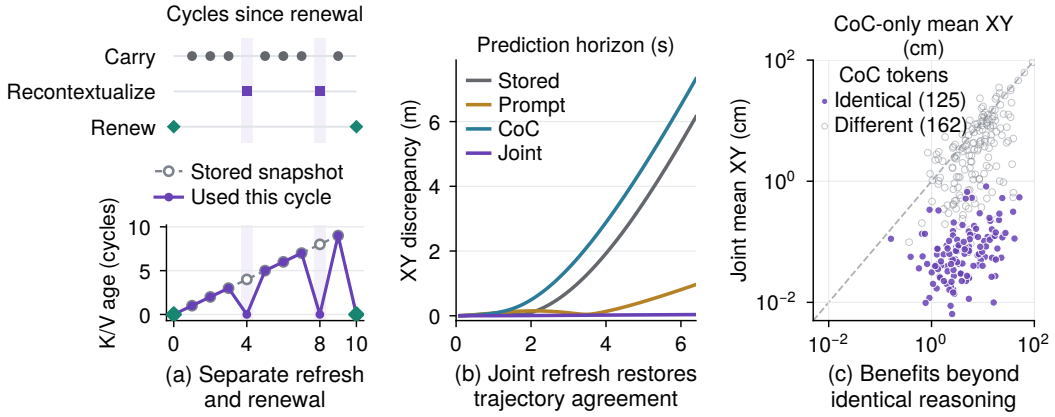


Figure 5: **Reasoning reuse through recontextualization.** **(a)** Context refresh and reasoning renewal follow separate schedules. **(b)** Joint refresh brings the predicted trajectory closer to the fresh-context reference than either partial refresh. **(c)** Across 287 comparisons from 24 scenes, Joint reduces discrepancy relative to CoC-only for both identical and different reasoning sequences, with most points below the equality line.

Every p planning cycles, where p is the recontextualization period, the action expert uses P_t^{full} with $K_{a\to t}$. Recontextualization refreshes the contextual state for the current query without replacing the stored reasoning snapshot (c_a, K_a) (Figure 5a). The benefit also extends to cases where retained and freshly generated CoC token sequences differ: Joint reduces 2-s mean XY discrepancy from the fresh-context reference relative to Prompt-only in 134 of 162 comparisons and CoC-only in 129 of 162 comparisons (Figure 5c; Appendix B).

To update the reasoning content separately from its contextual state, we use **Action-Aware CoC Renewal**. At renewal, we compare trajectories conditioned on retained and freshly generated CoC to determine the next reuse interval. Fresh CoC is generated under the prompt selected for the current cycle. Holding the current input, prompt selection, velocity-field evaluation schedule, and initial action noise fixed, let $y_{t,m}^C, y_{t,m}^F \in \mathbb{R}^2$ denote the m-th XY waypoint from the retained and fresh branches, respectively, for $m = 1, \ldots, M$. Here M is the comparison horizon in waypoints, and superscripts C and F denote retained/cached and fresh reasoning. Their mean and endpoint disagreement are

$$d_t^{\text{ADE}} = \frac{1}{M}\sum_{m=1}^{M} \left\| y_{t,m}^C - y_{t,m}^F \right\|_2, \qquad d_t^{\text{FDE}} = \left\| y_{t,M}^C - y_{t,M}^F \right\|_2. \tag{4}$$

The next reuse interval is

$$H_t^{\text{next}} = \begin{cases} H_{\text{long}}, & d_t^{\text{ADE}} \le \epsilon_A \wedge d_t^{\text{FDE}} \le \epsilon_F, \\ H_{\text{short}}, & \text{otherwise}, \end{cases} \tag{5}$$

where $H_{\text{long}} > H_{\text{short}}$ are the long and short renewal intervals, and ϵ_A, ϵ_F are trajectory-agreement thresholds. A valid fresh CoC and its contextual K/V replace the stored reasoning snapshot, and the current cycle becomes the new renewal anchor. Equation (5) controls its subsequent reuse duration; invalid cases are handled as described in Appendix C. The distances compare two predictions rather than either prediction with ground truth. On carry cycles, the retained branch uses stored K_a, so disagreement may reflect both reasoning-content change and contextual-state staleness.

4.2 Fast Action Generation with Velocity-Field Caching

Alpamayo's action expert repeatedly evaluates velocity fields to generate the current trajectory through flow matching. Let $\widehat{z}_t$ denote the conditioning context used for this generation: $(P_t^{\text{carry}}, K_a)$ on carry cycles and $(P_t^{\text{full}}, K_{a\to t})$ on recontextualization cycles without CoC renewal. When valid fresh CoC is adopted, it comprises the selected current prompt K/V and the fresh CoC K/V. During action generation, $\widehat{z}_t$ remains fixed while the intermediate action sequence and flow time evolve.

Figure 2b shows smaller velocity-field changes at intermediate steps, suggesting opportunities to reuse field information within the current trajectory generation.

Selected integration steps obtain a fresh velocity field from the network; the remaining steps predict fields from recent history on the native integration grid.

Velocity-Field Caching. Let

$$\{0, 1\} \subseteq \mathcal{E} \subseteq \{0, \dots, N-1\}$$

denote the flow-matching steps with fresh field evaluation, and let $\widetilde{v}_{t,k}$ denote the field used at step k. For $k \notin \mathcal{E}$, we model the next field change as a scaled continuation of the previous one, with coefficient λ_k. Let $\widehat{u}_{t,k}$ denote the intermediate action sequence at step k. The field is then

$$\widetilde{v}_{t,k} = \begin{cases} v_\theta(\widehat{u}_{t,k}, \tau_k; \widehat{z}_t)\,, & k \in \mathcal{E}, \\ \widetilde{v}_{t,k-1} + \lambda_k \left(\widetilde{v}_{t,k-1} - \widetilde{v}_{t,k-2}\right), & k \notin \mathcal{E}. \end{cases} \tag{6}$$

Direct reuse is recovered by $\lambda_k = 0$.

The native integration grid remains unchanged:

$$\widehat{u}_{t,0} = \xi_t, \qquad \widehat{u}_{t,k+1} = \widehat{u}_{t,k} + \Delta\tau_k \widetilde{v}_{t,k}, \quad k = 0, \dots, N-1, \qquad \widehat{y}_t = D(\widehat{u}_{t,N}; x_t), \tag{7}$$

where $\widehat{y}_t$ is the resulting trajectory. Thus, only steps in $\mathcal{E}$ invoke the velocity-field network, while all N native integration updates are retained.

Both evaluated and predicted fields enter the field history, so a later fresh evaluation is performed at the intermediate state actually reached by Eq. (7). Field history is reset for each new trajectory; during CoC renewal, the retained and fresh branches maintain separate histories while sharing the same initial action noise.

The recontextualization period p, renewal parameters $(M, H_{\text{short}}, H_{\text{long}}, \epsilon_A, \epsilon_F)$, fresh-evaluation set $\mathcal{E}$, and field-prediction coefficients $\{\lambda_k\}$ instantiate the reuse policy ρ and are specified in Section 5.

5 Experiments

Model. We evaluate DriveVLA-Cache on Alpamayo 1.5-10B, a reasoning state-of-the-art VLA model for autonomous driving (NVIDIA, 2026). DriveVLA-Cache is training-free and applicable to VLA models that combine autoregressive reasoning with a flow-matching action expert. Results on Alpamayo 1 (formerly Alpamayo-R1) are reported in Appendix A.2.

Dataset. We use the open-source NVIDIA Physical AI Autonomous Vehicles NuRec Dataset (NVIDIA, 2025) for closed-loop evaluation in AlpaSim (NVIDIA et al., 2025). We evaluate on 913 scenes from the AlpaSim `public_2601` suite.

Metrics. Following the official AlpaSim evaluation, we use Score_{AF} as the primary closed-loop metric and additionally report at-fault Collision, OffRoad, relative progress P_{rel}, and Plan Deviation. Score_{AF} combines clipped route progress with safety, assigning zero score after an official at-fault collision or off-road event (NVIDIA et al., 2025). Scores are displayed on a 100-point scale; retention is computed from unrounded values. For efficiency, we report driver-inference latency, end-to-end latency per simulator step (including model inference, control, simulation, and rendering), and speedup over FULL. Metric definitions and aggregation rules are provided in Appendix A.3.

Baselines. Given the absence of caching-based acceleration methods specifically designed for reasoning VLAs in autonomous driving, we compare against three representative caching methods from VLA inference: VLA-Cache (Xu et al., 2025), BAC (Ji et al., 2026), and TeaCache (Liu et al., 2025a). We use full-computation Alpamayo 1.5 (FULL) as the reference policy. The three caching baselines target individual stages of our inference pipeline. VLA-Cache (Xu et al., 2025) targets multimodal prefill by reusing selected visual K/V states. BAC (Ji et al., 2026) and TeaCache (Liu et al., 2025a) target flow-matching action generation: BAC reuses block residuals with a uniform cache schedule, while TeaCache caches action-expert evaluations using an Alpamayo-calibrated mapping.

Table 1: **Closed-loop driving quality and efficiency on 913 AlpaSim `public_2601` scenes.** DriveVLA-Cache has the highest mean score and lowest driver-inference and end-to-end latencies among evaluated methods. Scores are scaled by 100; parentheses give speedups over FULL.

Method	Closed-loop					Efficiency	
	Score_{AF} ↑ (×100)	$\text{Collision}_{\text{AF}}$ ↓	OffRoad ↓	P_{rel} ↑	Plan Dev. ↓	Driver (ms/inference) ↓	Latency (ms) ↓
Alpamayo 1.5 (FULL)	70.8	0.04	0.12	0.82	0.32	1832	2073 (1.0×)
BAC	67.2	0.04	0.13	0.81	0.19	1545	1809 (1.1×)
TeaCache	69.9	0.05	0.12	0.82	0.23	1605	1900 (1.1×)
VLA-Cache	70.6	0.04	0.12	0.82	0.31	2241	2444 (0.8×)
DriveVLA-Cache	**73.4**	**0.04**	**0.09**	**0.82**	**0.14**	**526**	**860 (2.4×)**

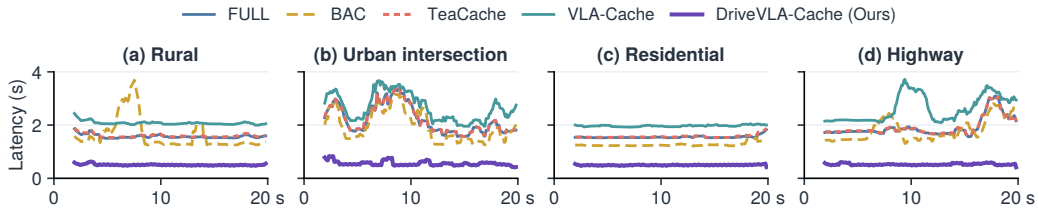


Figure 6: **Lower inference latency across driving scenes.** DriveVLA-Cache keeps mean latency below 1 s, while FULL and the caching baselines peak around 3.5 s.

Implementation Details. We implement DriveVLA-Cache on Alpamayo 1.5-10B using NVIDIA A100 (40 GB) GPUs. For hyperparameter settings, we set the recontextualization period $p = 4$, the trajectory-comparison horizon $M = 20$ waypoints (2 s), and the agreement thresholds $\epsilon_A = 0.30$ m and $\epsilon_F = 1.0$ m. For Action-Aware CoC Renewal, the intervals are $H_{\text{short}} = 5$ and $H_{\text{long}} = 10$ planning cycles, with the short interval used initially. For velocity-field caching, we retain $N = 10$ integration updates and set $\mathcal{E} = \{0, 1, 3, 8\}$ for fresh evaluations. At skipped steps, we use $\lambda_2 = 0$ and $\lambda_k = 1/4$ for $k \in \{4, 5, 6, 7, 9\}$.

5.1 Main Results

Closed-loop evaluation. Table 1 compares DriveVLA-Cache with FULL and three stage-specific caching baselines on 913 AlpaSim scenes. DriveVLA-Cache achieves the highest mean driving score and the lowest driver-inference and end-to-end latencies among all evaluated methods, without modifying the pre-trained model. It reduces driver-inference latency from 1832 to 526 ms (3.5×) and end-to-end latency, which also includes simulation and rendering, from 2073 to 860 ms (2.4×), while increasing mean Score_{AF} from 70.8 to 73.4. In Figure 6, DriveVLA-Cache maintains mean latency below 1 s, while baseline latencies fluctuate and peak around 3.5 s. Appendix D shows sustained sub-second mean latency across 24 scenes covering 12 NVIDIA scene-label combinations. Appendix E breaks down driving quality and latency across scene categories, showing the lowest mean driver latency in all 15 groups and the highest mean driving score in 11 groups. Relative progress and the at-fault collision rate remain unchanged at the displayed precision. OffRoad decreases from 0.12 to 0.09, and Plan Deviation is more than halved, from 0.32 to 0.14, indicating more temporally consistent successive plans. Together, these results demonstrate that substantial acceleration can be achieved alongside a higher mean driving score and more consistent successive plans. Appendix A.4 presents a qualitative example of closed-loop driving in Figure 8.

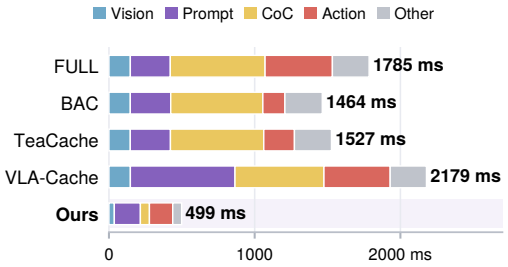


Figure 7: **Stage-wise inference latency.** Five methods on the same development scene using an A100 (40 GB). Bars average 179 calls after excluding the first, with synchronization at stage boundaries.

BAC and TeaCache achieve only approximately $1.1\times$ end-to-end speedups, while their mean driving scores fall 3.6 and 0.9 points below FULL, respectively. Both methods accelerate action generation but leave visual encoding, multimodal prefill, and CoC decoding on the inference path. On the profiled scene (Figure 7), DRIVEVLA-CACHE reduces CoC latency from 650 to 62 ms, and its complete driver inference takes 499 ms—less than CoC decoding alone in every baseline (611–650 ms). In contrast, VLA-Cache closely preserves FULL's score but is incompatible with FlashAttention (Dao et al., 2022) due to its requirement for explicit attention weights, which incurs extra overhead and yields no net speedup ($0.8\times$). Compared with BAC, the fastest caching baseline, DRIVEVLA-CACHE achieves a further $2.1\times$ end-to-end speedup and a 6.2-point higher mean driving score. These results demonstrate lower inference cost and higher mean closed-loop driving quality with the pretrained model unchanged, while generating a new trajectory at every planning cycle.

5.2 ABLATION STUDIES

Table 2: **Component ablation on 128 randomly sampled scenes.** The full framework gains 7.1 mean Score_{AF} points over FULL with a $2.1\times$ end-to-end speedup; Reasoning Recontextualization adds 1.7 of these points for 65 ms. Components are added cumulatively; checkmarks denote enabled components and parentheses give speedups over FULL.

Variant	Components				Closed-loop					Efficiency
	Visual cache	Context / CoC reuse	Reasoning Recontextualization	Sparse action	$\text{Score}_{\text{AF}}\uparrow$ ($\times 100$)	$\text{Collision}_{\text{AF}}\downarrow$	OffRoad $\downarrow$	$P_{\text{rel}}\uparrow$	Plan Dev. $\downarrow$	Latency (ms) $\downarrow$
FULL	×	×	×	×	66.1	0.08	0.13	0.83	0.32	1932 (1.0×)
+ Visual cache	✓	×	×	×	66.1	0.08	0.13	0.83	0.32	1705 (1.1×)
+ Sparse action	✓	×	×	✓	68.8	0.05	0.12	0.83	0.27	1494 (1.3×)
+ Context / CoC reuse	✓	✓	×	✓	71.5	0.04	0.11	0.83	0.23	863 (2.2×)
DRIVEVLA-CACHE	✓	✓	✓	✓	**73.2**	**0.05**	**0.09**	**0.83**	**0.26**	**928 (2.1×)**

Table 2 adds the components cumulatively on the same 128 scenes. Visual caching is exact: it removes 227 ms (1932 $\rightarrow$ 1705 ms) with every closed-loop metric unchanged. Adding velocity-field caching saves a further 211 ms and improves mean Score_{AF} by 2.7 points, while lowering the at-fault collision rate from 0.08 to 0.05 and Plan Deviation from 0.32 to 0.27. Context/CoC reuse delivers the largest incremental latency reduction in this sequence, saving 631 ms (1494 $\rightarrow$ 863 ms) while improving mean Score_{AF} by another 2.7 points. The latency saving comes from reducing repeated prompt computation and avoiding per-cycle autoregressive CoC decoding. Reasoning Recontextualization adds a further 1.7 score points (71.5 $\rightarrow$ 73.2) for 65 ms of additional computation, achieving the highest mean score and lowest OffRoad rate (0.09) among the tested configurations. These results demonstrate the practical value of the Reasoning Content–State Distinction: retaining CoC content saves repeated decoding, while refreshing its contextual K/V improves the mean driving score. Together, the components yield a $2.1\times$ end-to-end speedup and a 7.1-point higher mean driving score than FULL on the same scenes. Appendix A.1 examines the recontextualization period p and field-evaluation schedule in Table 3, and the CoC renewal horizon H in Table 3c. All tested renewal horizons outperform per-cycle CoC regeneration in both mean score and latency.

6 LIMITATIONS AND FUTURE WORK

Our current implementation uses predefined trajectory-agreement thresholds and a predefined field-evaluation schedule. Future work may explore adaptive threshold selection and field-evaluation schedules conditioned on the current driving context. Combining computation reuse with quantization, pruning, and lightweight adaptation is another promising direction.

7 CONCLUSION

We presented DRIVEVLA-CACHE, a training-free inference acceleration framework exploiting slow–fast asymmetry around the action expert. The Reasoning Content–State Distinction motivates Reasoning Recontextualization, which refreshes the contextual K/V of retained CoC tokens without repeated autoregressive decoding. Combined with visual feature caching and velocity-field caching, the framework achieves a $2.4\times$ end-to-end speedup while increasing mean Score from 70.8 to 73.4 on 913 closed-loop scenes.

AI USE STATEMENT

We used generative AI tools to assist with manuscript preparation, including drafting, language polishing, and figure preparation. The authors take responsibility for the final content of this paper.

REFERENCES

Yuan Chen, Zi-han Ding, Ziqin Wang, Yan Wang, Lijun Zhang, and Si Liu. Asynchronous large language model enhanced planner for autonomous driving. In *Computer Vision – ECCV 2024*, pp. 22–38, 2024. doi: 10.1007/978-3-031-72764-1_2.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Dingkang Liang, Chong Zhang, Dingyuan Zhang, Hongwei Xie, Bing Wang, and Xiang Bai. ORION: A holistic end-to-end autonomous driving framework by vision-language instructed action generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 24823–24834, 2025.

Huanlin Gao, Ping Chen, Fuyuan Shi, Ruijia Wu, Yantao Li, Qiang Hui, Yuren You, Ting Lu, Chao Tan, Shaoan Zhao, Zhaoxiang Liu, Fang Zhao, Kai Wang, and Shiguo Lian. MeanCache: From instantaneous to average velocity for accelerating flow matching inference. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/62868cc2fc1eb5cdf321d05b4b88510c-Abstract-Conference.html.

Kangye Ji, Yuan Meng, Hanyun Cui, Ye Li, Jianbo Zhou, Shengjia Hua, Lei Chen, and Zhi Wang. Block-wise adaptive caching for accelerating diffusion policy. In *International Conference on Learning Representations*, volume 2026, pp. 22805–22858, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/2712b17bb58ea5b2b65c45857b024744-Abstract-Conference.html.

Fanjie Kong, Yitong Li, Weihuang Chen, Chen Min, Yizhe Li, Zhiqiang Gao, Haoyang Li, Zhongyu Guo, and Hongbin Sun. VLR-Driver: Large vision-language-reasoning models for embodied autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 26966–26976, 2025.

Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shu-Tao Xia, Zhi Wang, and Wenwu Zhu. SP-VLA: A joint model scheduling and token pruning approach for VLA model acceleration. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RwdGIIjPlC.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7353–7363, 2025a. URL https://openaccess.thecvf.com/content/CVPR2025/html/Liu_Timestep_Embedding_Tells_Its_Time_to_Cache_for_Video_Diffusion_CVPR_2025_paper.html.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with TaylorSeers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15853–15863, 2025b. URL https://openaccess.thecvf.com/content/ICCV2025/html/Liu_From_Reusing_to_Forecasting_Accelerating_Diffusion_Models_with_TaylorSeers_ICCV_2025_paper.html.

Pei Liu, Qingtian Ning, Xinyan Lu, Haipeng Liu, Weiliang Ma, Dangen She, Xianpeng Lang, and Jun Ma. CogDriver: Integrating cognitive inertia for temporally coherent planning in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18150–18160, 2026a. URL https://openaccess.thecvf.com/content/CVPR2026/html/Liu_CogDriver_Integrating_Cognitive_Inertia_for_Temporally_Coherent_Planning_in_Autonomous_CVPR_2026_paper.html.

Yicheng Liu, Shiduo Zhang, Zibin Dong, Baijun Ye, Tianyuan Yuan, Xiaopeng Yu, Linqi Yin, Chenhao Lu, Junhao Shi, Luca Jiang-Tao Yu, Liangtao Zheng, Tao Jiang, Jingjing Gong, Xipeng Qiu, and Hang Zhao. FASTer: Toward efficient autoregressive vision language action modeling via neural action tokenization. In *International Conference on Learning Representations*, 2026b. URL https://openreview.net/pdf?id=k6nTUFoqeT.

Zhengyao Lyu, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K. Wong. FasterCache: Training-free video diffusion model acceleration with high quality. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/518046d86bbc41a0707727c38301ad8e-Abstract-Conference.html.

NVIDIA. Physical AI Autonomous Vehicles NuRec Dataset. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/nvidia/PhysicalAI-Autonomous-Vehicles-NuRec.

NVIDIA. Alpamayo 1.5-10B: Model card and inference architecture. Official model documentation, 2026. URL https://huggingface.co/nvidia/Alpamayo-1.5-10B.

NVIDIA, Yulong Cao, Riccardo de Lutio, Sanja Fidler, Guillermo Garcia Cobo, Zan Gojcic, Maximilian Igl, Boris Ivanovic, Peter Karkus, Janick Martinez Esturo, Marco Pavone, Aaron Smith, Ellie Tanimura, Michal Tyszkiewicz, Michael Watson, Qi Wu, and Le Zhang. AlpaSim: A modular, lightweight, and data-driven research simulator for autonomous driving. Software, October 2025. URL https://github.com/NVlabs/alpasim.

NVIDIA, Yan Wang, Wenjie Luo, Junjie Bai, Yulong Cao, Tong Che, Ke Chen, Yuxiao Chen, Jenna Diamond, Yifan Ding, Wenhao Ding, et al. Alpamayo 1: Bridging reasoning and action prediction for generalizable autonomous driving in the long tail. Technical report, NVIDIA Research, 2026. URL https://research.nvidia.com/publication/2025-10_alpamayo-r1.

Ryuji Oi, Hikari Otsuka, Kosuke Matsushima, Yuki Ichikawa, Masato Motomura, Tatsuya Kaneko, and Daichi Fujiki. Actioncache: Training-free acceleration for vision-language-action models with action caching and refinement. *arXiv preprint arXiv:2607.06370*, 2026. URL https://arxiv.org/abs/2607.06370.

Xiaohuan Pei, Yuxing Chen, Yuheng Shi, Siyu Xu, Yunke Wang, and Chang Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1223ca2b14513106ad55db6a537337f9-Abstract-Conference.html.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.012. URL https://www.roboticsproceedings.org/rss21/p012.html.

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

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Hao Shao, Yuxuan Hu, Letian Wang, Guanglu Song, Steven L. Waslander, Yu Liu, and Hongsheng Li. LMDrive: Closed-loop end-to-end driving with large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15120–15130, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Shao_LMDrive_Closed-Loop_End-to-End_Driving_with_Large_Language_Models_CVPR_2024_paper.html.

Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. DriveLM: Driving with graph visual question answering. In *Computer Vision – ECCV 2024*, volume 15110 of *Lecture Notes in Computer Science*, pp. 256–274, 2025. doi: 10.1007/978-3-031-72943-0_15. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/html/6870_ECCV_2024_paper.php.

Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. DriveVLM: The convergence of autonomous driving and large vision-language models. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4698–4726. PMLR, 2025. URL https://proceedings.mlr.press/v270/tian25c.html.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-Cache: Efficient vision-language-action manipulation via adaptive token caching. In *Advances in Neural Information Processing Systems*, volume 38, pp. 164448–164473, 2025. doi: 10.52202/085713-5484.

Yuhao Xu, Yantai Yang, Zhenyang Fan, Yufan Liu, Yuming Li, Bing Li, and Zhipeng Zhang. QVLA: Not all channels are equal in vision-language-action model's quantization. In *International Conference on Learning Representations*, 2026.

Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. EfficientVLA: Training-free acceleration and compression for vision-language-action models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 40891–40914, 2025. doi: 10.52202/085713-1365.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*, 2025. doi: 10.1145/3689031.3696098. URL https://doi.org/10.1145/3689031.3696098.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. DeeR-VLA: Dynamic inference of multimodal large language models for efficient robot execution. In *Advances in Neural Information Processing Systems*, volume 37, pp. 56619–56643, 2024. doi: 10.52202/079017-1803.

Ruifei Zhang, Junlin Xie, Wei Zhang, Weikai Chen, Xiao Tan, Xiang Wan, and Guanbin Li. AdaDrive: Self-adaptive slow-fast system for language-grounded autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5112–5121, 2025.

Zewei Zhou, Tianhui Cai, Seth Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. AutoVLA: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 27920–27956, 2025. doi: 10.52202/085713-0942.

A Additional Experimental Results

A.1 Ablation

Ablation. Table 3 quantifies the quality–latency trade-offs across the three reuse mechanisms. All four tested recontextualization periods improve mean Score_{AF} over disabling recontextualization (65.7–67.2 versus 63.3). In (a), increasing the recontextualization period from $p = 1$ to $p = 4$ saves 152 ms while improving mean Score_{AF} by 1.5 points; increasing it further to $p = 8$ saves another 30 ms with a 0.4-point score reduction. In (b), placing four velocity-field evaluations at $\{0, 1, 3, 8\}$ instead of $\{0, 1, 2, 3\}$ raises the mean score from 67.4 to 75.1 with a latency increase of 5 ms (876 to 881 ms), showing the benefit of distributing evaluations across early and late integration steps. In (c), all tested CoC renewal horizons outperform per-cycle CoC regeneration in both mean score and latency. Extending H from 5 to 20 saves 132 ms (1168 to 1036 ms), while the mean score decreases by 1.6 points (72.9 to 71.3).

Table 3: **Ablation of Reasoning Recontextualization, velocity-field evaluation, and CoC renewal.** In (c), **Off** disables CoC reuse; all variants use $p = 4$ and dense field evaluation.

Setting	$\text{Score}_{\text{AF}}\uparrow$ ($\times 100$)	$\text{Collision}_{\text{AF}}\downarrow$	OffRoad$\downarrow$	$P_{\text{rel}}\uparrow$	Plan Dev.$\downarrow$	Latency (ms)$\downarrow$
(a) Reasoning Recontextualization						
Off	63.3	0.070	0.109	0.82	0.09	779
$p = 1$	65.7	0.062	0.094	0.81	0.17	994
$p = 2$	66.6	0.070	0.102	0.82	0.23	902
$p = 4$	67.2	0.062	0.102	0.83	0.15	842
$p = 8$	66.8	0.055	0.117	0.83	0.12	812
(b) Velocity-field evaluation						
$\{0, 1, 2\}$	66.6	0.078	0.117	0.83	0.15	837
$\{0, 1, 3\}$	67.6	0.062	0.102	0.83	0.15	826
$\{0, 1, 2, 3\}$	67.4	0.055	0.109	0.83	0.15	876
$\{0, 1, 3, 8\}$	75.1	0.039	0.078	0.83	0.14	881
Dense 0–9	69.8	0.039	0.109	0.82	0.16	1089
(c) CoC renewal horizon						
CoC reuse Off	69.0	0.047	0.125	0.83	0.33	1612
$H = 5$	72.9	0.055	0.102	0.83	0.28	1168
$H = 10$ (reference)	71.9	0.070	0.094	0.82	0.29	1073
$H = 20$	71.3	0.070	0.094	0.83	0.30	1036

A.2 DriveVLA-Cache on Alpamayo 1

We further validate DriveVLA-Cache on Alpamayo 1 (R1) (NVIDIA et al., 2026). As shown in Table 4, DriveVLA-Cache achieves a 2.1× end-to-end speedup while retaining 98.9% of FULL's mean driving score. These results extend the closed-loop acceleration benefits of DriveVLA-Cache to a second model in the Alpamayo family.

Table 4: **Closed-loop evaluation on Alpamayo 1 across 913 scenes.** DriveVLA-Cache delivers a 2.1× end-to-end speedup on Alpamayo 1 while retaining 98.9% of FULL's mean driving score, demonstrating acceleration benefits across model generations.

Method	$\text{Score}_{\text{AF}}\uparrow$ ($\times 100$)	$\text{Collision}_{\text{AF}}\downarrow$	OffRoad$\downarrow$	$P_{\text{rel}}\uparrow$	Plan Dev.$\downarrow$	Driver (ms/inference)$\downarrow$	Latency (ms)$\downarrow$	Speedup$\uparrow$
Alpamayo 1 (FULL)	71.1	0.05	0.12	0.81	0.34	1853	2097	1.0×
DriveVLA-Cache	**70.4**	**0.05**	**0.12**	**0.82**	**0.30**	**698**	**1023**	**2.1×**

A.3 ALPASIM METRICS

We use the official ALPASIM evaluation outputs (NVIDIA et al., 2025). Table 5 defines the closed-loop and efficiency metrics reported in our experiments and their aggregation rules. Closed-loop metrics are first reduced within each rollout and then averaged across the evaluated scenes. For binary events, this yields the fraction of affected scenes. Plan Deviation averages only valid scene values; missing values are not treated as zero.

Table 5: **Closed-loop and efficiency metrics.** Temporal reductions follow the official evaluator; efficiency uses our measured driver-inference and rollout-session times.

Metric	Definition	Aggregation
$\text{Score}_{\text{AF}} \uparrow$	Official ALPASIM at-fault scene score, displayed on a 100-point scale.	Mean over scenes
$\text{Collision}_{\text{AF}} \downarrow$	Official at-fault collision indicator within the scoring window.	Max over time
OffRoad↓	Official indicator that the ego vehicle leaves the drivable region within the scoring window.	Max over time
$P_{\text{rel}} \uparrow$	Normalized projection of the ego centroid onto the recorded reference path up to the current timestamp.	Min over time
Plan Dev. (m)↓	L2 discrepancy between consecutive selected plans at shared future timestamps, with exponential weighting favoring nearer waypoints.	Mean over time
Driver (ms)↓	Measured driver inference time, excluding the first driver inference of each scene.	Pooled inferences
Latency (ms)↓	Rollout-session wall time, including setup and warmup, divided by the requested simulator steps.	Per requested step
Speedup↑	End-to-end latency of the corresponding campaign's FULL baseline divided by the method's end-to-end latency.	Ratio of latencies

Latency includes the driver, controller, and rendering work within the rollout session; external service launch and offline scoring are excluded. Driver latency and end-to-end latency use different denominators and warmup conventions.

The stage breakdown in Figure 7 and Table 6 profiles all five methods on the same development scene, using one trajectory sample per generation branch. We exclude the first driver call and average the remaining 179 calls, including recontextualization and renewal. CUDA synchronization at stage boundaries makes these diagnostic timings distinct from the unprofiled 913-scene averages in Table 1.

Table 6: **Stage-wise inference latency with the final configuration.** DRIVEVLA-CACHE achieves a $3.6\times$ speedup over FULL on the profiled scene. All methods run on an NVIDIA A100 (40 GB).

	Latency (with 1 trajectory) (ms)						
Method	**Vision**	**Prompt**	**CoC**	**Action**	**Other**	**Total**	**Speedup**
Alpamayo 1.5 (FULL)	147	274	650	462	252	1785	$1.0\times$
BAC	147	277	632	151	256	1464	$1.2\times$
TeaCache	146	276	641	210	254	1527	$1.2\times$
VLA-Cache	147	718	611	454	249	2179	$0.8\times$
DRIVEVLA-CACHE	**37**	**179**	**62**	**160**	**61**	**499**	$\mathbf{3.6\times}$

A.4 VISUALIZATION

A recorded closed-loop driving episode. Figure 8 shows one illustrative right turn from the 913-scene DRIVEVLA-CACHE evaluation (scene `39f3ab94`). The policy approaches the junction, turns, and exits onto the next road. This episode receives a Score_{AF} of 100 on the reported scale, with no recorded collision or off-road event. We selected this case for visual clarity; aggregate performance is reported in Table 1.

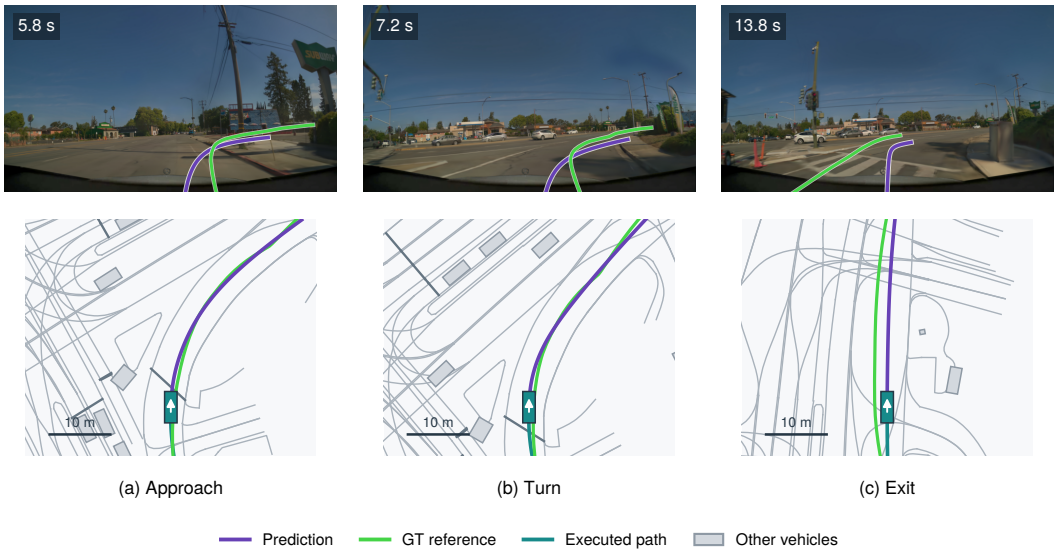


Figure 8: **A closed-loop driving episode of DRIVEVLA-CACHE in ALPASIM.** Three synchronized views from one DRIVEVLA-CACHE episode: approach, turn, and exit. Top: logged front-camera observations with the current predicted trajectory (purple) and the original recorded GT path (green), projected using the camera calibration. Bottom: corresponding ego-oriented map views, with the executed ego path in teal and other vehicles in gray; all panels use the same metric scale. GT is a spatial reference from the original recording, not the accelerated policy's executed path. Time is measured from the first policy query.

B ADDITIONAL MECHANISM EVIDENCE

Effect of Reasoning Recontextualization. To isolate contextual K/V effects, we compare strategies in 125 cases from 19 scenes with identical retained and freshly generated CoC token sequences. Joint reduces 2-s XY discrepancy relative to both Prompt-only and CoC-only in every case. Median discrepancies from the fresh-context reference are 14.05, 14.71, 3.70, and 0.062 cm for Stored, Prompt-only, CoC-only, and Joint, respectively. When the sequences differ, Joint improves over Prompt-only in 134 of 162 comparisons and CoC-only in 129.

We preselect 12 query times in each of 24 development scenes, excluding the original four mechanism scenes. One native fresh-anchor query lacks an eligible retained branch, leaving 287 comparisons. Same-state branches fix recorded observations, share physical action noise, and use ten field evaluations. They start from cloned memory without committing updates; predictions are not executed as closed-loop interventions. These comparisons supply Figures 2c and 5c; native profiles and the Scene 053 example use separately reported samples.

B.1 REUSE ACROSS PLANNING CYCLES

We first examine the native policy before changing its inference path. Eight selected Alpamayo 1.5 FULL rollouts provide 1,446 planning cycles and 6,926 pairs at gaps of 1, 2, 5, 10, and 20 planning cycles. The panel contains four scenes with historical cache regressions and four without a historical score loss; it is a development panel, not an estimate of the 913-scene population. The vehicle always executes native FULL.

Stable tokens do not imply unchanged contextual state. Among the 6,926 pairs, 2,835 have identical complete CoC suffixes. Even for these pairs, median K drift and local K+V readout difference increase with the planning-cycle gap in all eight scenes (Figure 9). The readout diagnostic holds the current Action query and attention background fixed while substituting retained CoC K/V at matched token positions, measuring the local effect of reusing contextual states.

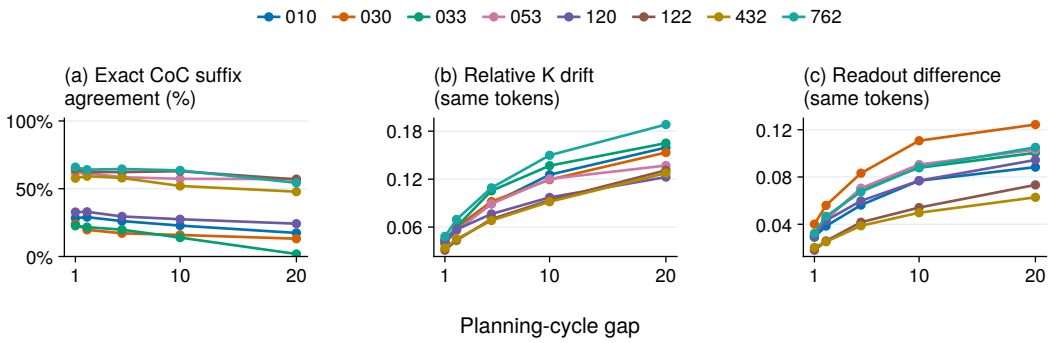


Figure 9: **Reasoning Content–State Distinction.** Colors identify eight FULL scenes. (a) Exact CoC suffix agreement (%). (b,c) Median relative pre-RoPE K drift and local K+V readout difference, conditional on identical complete CoC suffixes. Readout substitutions are unexecuted diagnostics.

C Implementation Details

Partial prompt update. For the four-camera, four-frame window, we index frame slots from 0 to 15 in camera-major, chronological order. The newest frames occupy slots $\{3, 7, 11, 15\}$, and the default historical recomputation set is $\mathcal{R} = \{0, 2, 4, 5, 6, 12, 13, 14\}$. On eligible carry cycles, all overlapping historical visual K/V can be reused across consecutive cycles. Frames leaving the current window are discarded. Other partial updates recompute the historical slots in $\mathcal{R}$. If an attempt to reuse all historical K/V fails validation, prefill is repeated with this default set. Reasoning Recontextualization instead recomputes the complete current prompt. For reused visual K/V, cached pre-RoPE keys receive M-RoPE at their current window positions, while their values are retained. The attention mask follows the current logical token order.

CoC renewal. At each CoC renewal, Algorithm 1 generates fresh CoC and, if valid, adopts it and determines its subsequent reuse interval. The two action branches use the field-evaluation schedule in Section 5.

Algorithm 1 Action-Aware CoC Renewal

Require: Current input x_t, selected prompt P_t^{sel}, retained context $\widehat{z}_t$, shared noise ξ_t
$y^C \leftarrow \textsc{GenerateAction}(x_t, \widehat{z}_t, \xi_t)$
Generate fresh CoC c^F and its K/V K^F under P_t^{sel}
if fresh generation or its context is invalid **then**
    return native fallback
end if
$y^F \leftarrow \textsc{GenerateAction}(x_t, (P_t^{\text{sel}}, K^F), \xi_t)$
$H \leftarrow H_{\text{short}}$
if the pair is valid and $\text{ADE}_M(y^C, y^F) \le \epsilon_A$ and $\text{FDE}_M(y^C, y^F) \le \epsilon_F$ **then**
    $H \leftarrow H_{\text{long}}$
end if
Commit the fresh CoC snapshot (c^F, K^F, t)
Schedule the next CoC renewal after H planning cycles
return y^F

D INFERENCE LATENCY ACROSS DRIVING SCENES

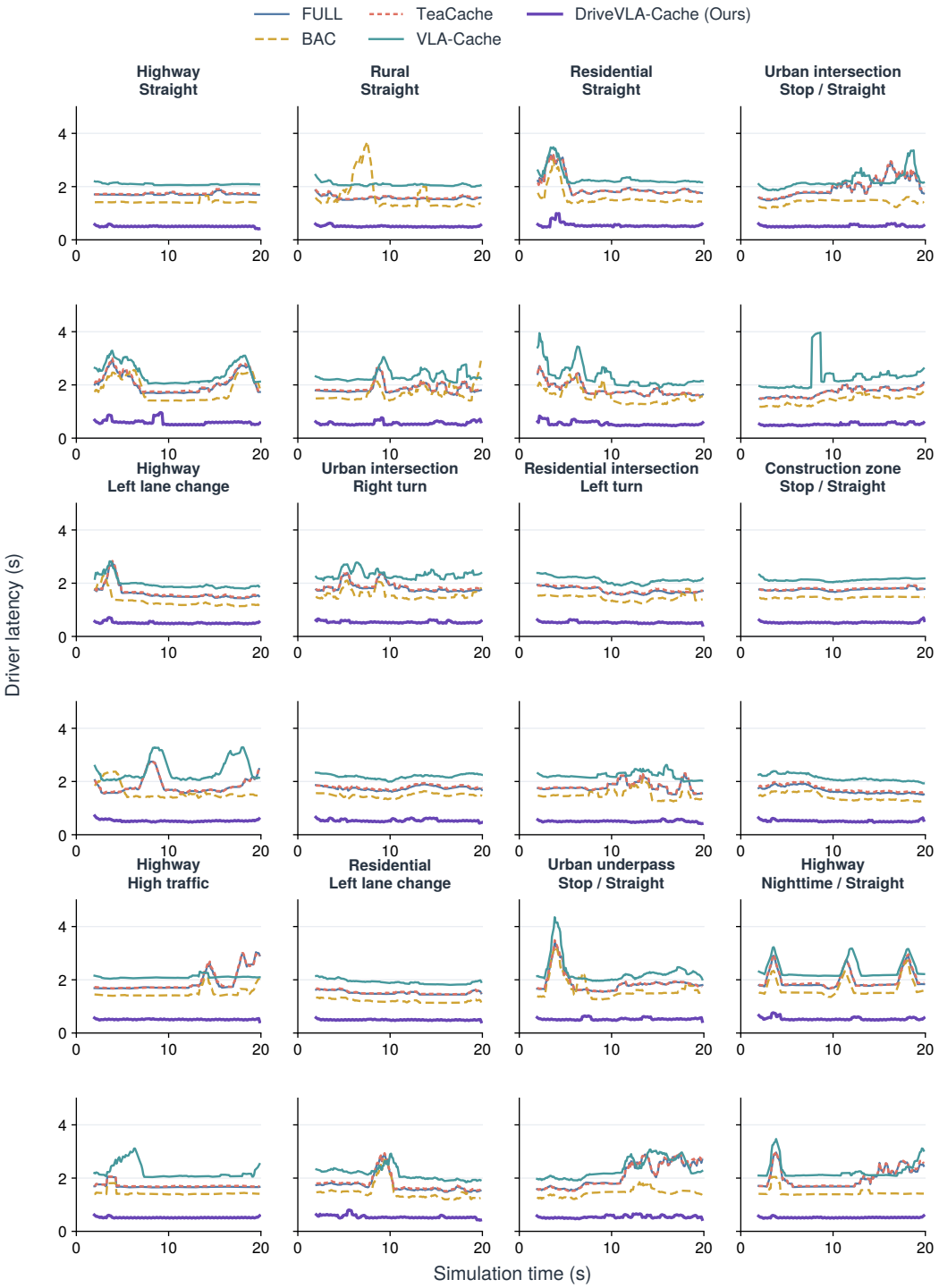


Figure 10: **Sustained inference acceleration across 24 driving scenes.** DRIVEVLA-CACHE keeps 1-s mean driver latency below 1 s throughout all 24 scenes, with 3.1–3.9× scene-average speedups over FULL and lower mean latency than each caching baseline. Two scenes per NVIDIA label combination are paired vertically, extending Figure 6. Curves exclude the first driver call and share axes; time is measured from each simulation start.

E Driving Quality and Latency across Scene Categories

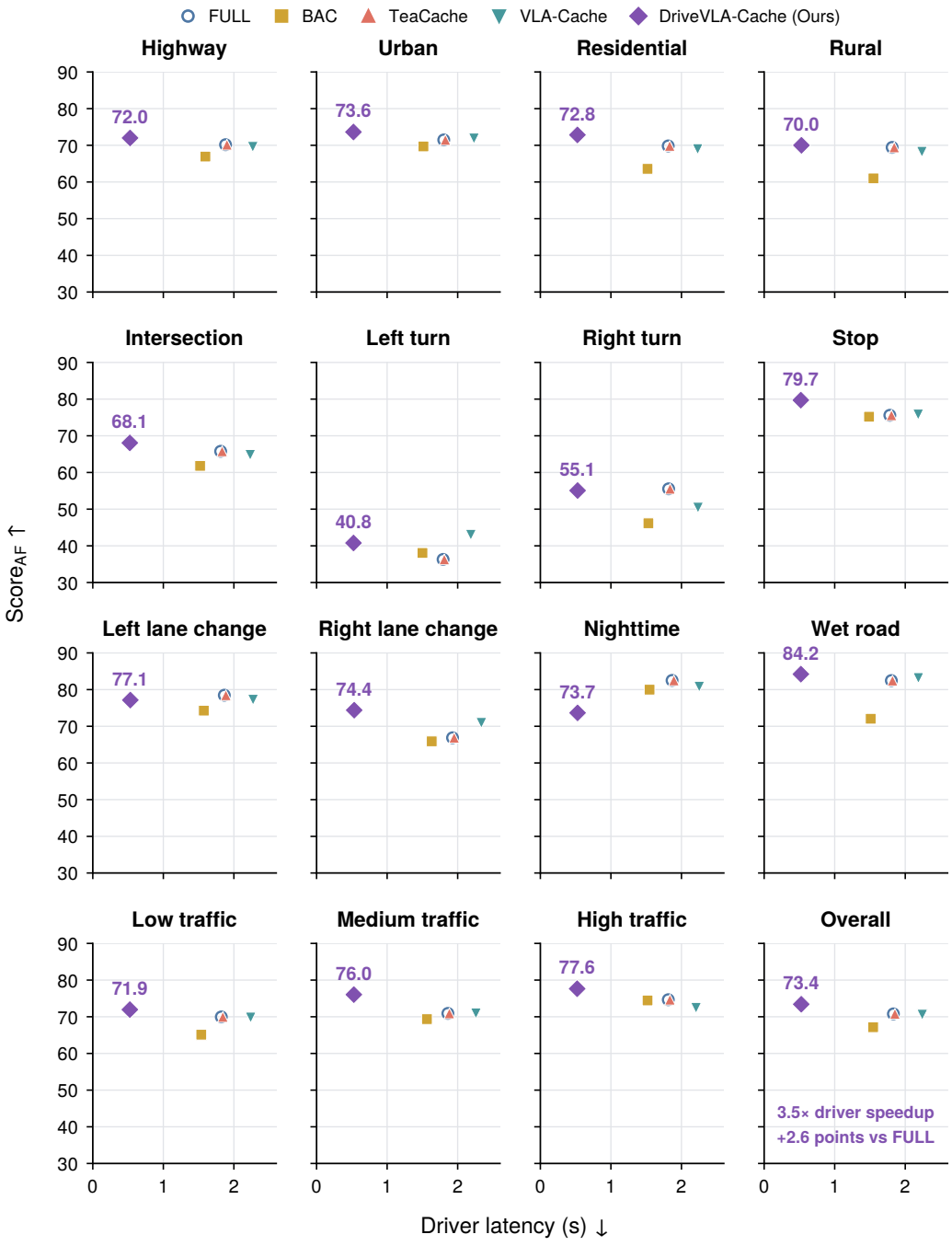


Figure 11: **Faster inference across scene categories with higher overall driving quality.** DriveVLA-Cache achieves the lowest mean driver latency in all 15 NVIDIA tag groups and the highest mean Score_{AF} in 11 groups. Across all 913 scenes, it improves the score by 2.6 points (70.8 to 73.4) with a 3.5× driver speedup over FULL (1.832 to 0.526 s). Category results use the 821 scenes with available labels; groups may overlap, while Overall includes all 913 scenes.