VISION-LANGUAGE-ACTION AUTONOMOUS DRIVING AGENT WITH LANGUAGE-BASED MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) foundation models have recently emerged as one of the prevailing solutions for autonomous driving, as they can utilize knowledge acquired during vision-language pretraining for accurate and interpretable driving. However, VLAs can take only a limited number of frames as visual input due to the high token cost of an image, which is problematic for memory-dependent tasks such as determining the arrival order at all-way stops and long-horizon driving scene understanding. Existing solutions use latent vector memories accessed through cross-attention, which are neither interpretable nor portable. In this paper, we propose AD-Memo, a general-purpose VLA driving agent with language-based memory. The agent outputs memory as an extension of its Chain-of-Thought (CoT) to record surrounding objects critical to driving; this memory becomes part of the agent's future input. We curate memory-based datasets and train VLAs with a two-stage recipe: Supervised Fine-Tuning (SFT) and *Da Capo*, a novel semi-closed-loop Reinforcement Learning (RL) algorithm which uses trajectory-level advantage for memory and step-level advantage for driving, leading to better credit assignment. Across scenarios such as all-way stops and general driving, AD-Memo improves driving quality, enables better question answering on driving scenes, and produces plug-and-play memory for other models.

1 INTRODUCTION

In recent years autonomous driving has successfully transitioned from the lab to real-world commercial deployment at scale (Kusano et al., 2025; Mawakana & Dolgov, 2026). During this period, Vision-Language-Action (VLA) models (Zitkovich et al., 2023; O'Neill et al., 2024) have rapidly emerged as one of the prevailing solutions (Wang et al., 2025; Li et al., 2026e; Zhou et al., 2026) for autonomous driving, as they acquire knowledge through large-scale vision-language pretraining and apply it after fine-tuning on driving data. Through this architectural synergy, VLA driving agents produce accurate and interpretable driving trajectories in an end-to-end manner.

However, the high token cost of multiple images and the quadratic computational cost of attention (Dao et al., 2022) severely limit the time horizon of VLA inputs (Shi et al., 2026a). For example, for a 4-camera VLA receiving visual input at 10Hz with 200 tokens per image, a 0.4s history requires $0.4/0.1 \times 200 \times 4 = 3200$ tokens; expanding this history to 4 seconds increases the input length to 32K tokens and the quadratic attention cost by a factor of 100, which is prohibitively expensive for time-sensitive autonomous driving tasks (Liu et al., 2022). While one can simply shorten the input horizon for faster inference, the lack of historical context becomes problematic for memory-dependent driving tasks, such as determining the arrival order at all-way stops (Liang et al., 2026), identifying occluded road objects (Kumar et al., 2025; Xie et al., 2025), and long-horizon driving scene understanding (Zeng et al., 2025a;b) — all of which are important for driving.

To address this issue, prior works have introduced memory mechanisms into VLA driving agents using two prevalent approaches: *latent vectors* (Fu et al., 2025; Huang et al., 2026c) and *language in-context learning* (Wen et al., 2024; Mei et al., 2024). The former stores latent vectors extracted from past inputs and accesses them via cross-attention (Vaswani et al., 2017), while the latter stores structured examples of prior driving experiences. However, both approaches have shortcomings. Latent vector memory is not interpretable by humans or portable to other agents; it does not make full use of the VLA's reasoning ability inherited from its backbone, which motivates the use of

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.

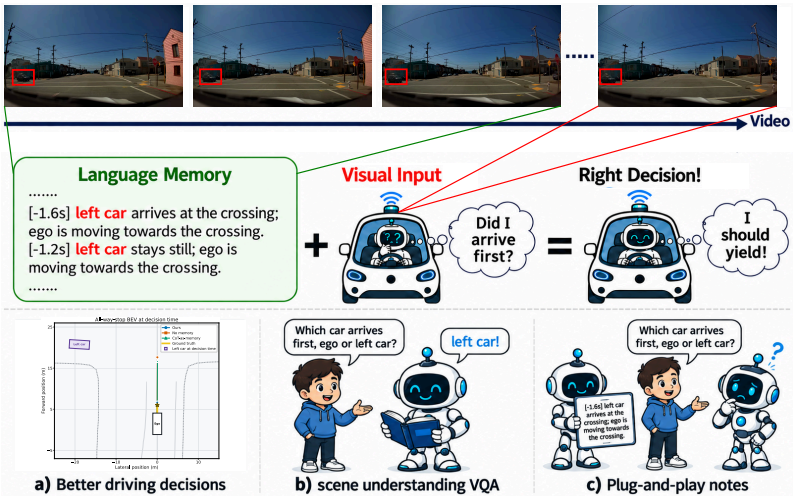


Figure 1: An illustration of AD-Memo. AD-Memo generates language-based memory periodically, which is beneficial for a diverse set of tasks illustrated in panel (a), (b) and (c), including driving, scene understanding question-answering tasks, and producing portable memory for other models.

VLAs in the first place. In turn, the language in-context learning approaches do not provide *in-episode* memory, i.e., they do not help retain information about critical objects or events that affect current driving. Thus, in this work we ask: *Can we fully exploit the knowledge in a VLA's backbone by using language-based, in-episode memory that is brief, explainable, and portable?*

To achieve this, we propose AD-Memo, a general-purpose **A**utonomous **D**riving VLA agent with language-based, in-episode **memo**ry (see Fig. 1 for an overview). The agent outputs memory as an extension of its Chain-of-Thought (CoT) (Wei et al., 2022), which then becomes part of the agent's future input. AD-Memo acquires this ability through a two-stage training recipe: Supervised Fine-Tuning (SFT) and *Da Capo*, a novel semi-closed-loop RL algorithm. *Da Capo* has two features: (1) it uses open-loop RL (i.e., ground-truth trajectory replay) for driving states to avoid the high cost of environment simulation, while using closed-loop memory (i.e., the agent's own memory generated at past steps) to overcome exposure bias (Bengio et al., 2015); (2) inspired by stochastic computation graph theory (Schulman et al., 2015), it assigns different reward components to driving tokens, which do not affect future steps under open-loop control, and memory tokens which do affect future steps, improving credit assignment. To apply this recipe, we curate a task-specific (all-way stop) dataset and a task-agnostic general driving dataset using rule-based and agentic frameworks, respectively. We then evaluate our model on the corresponding test sets, showing that AD-Memo not only improves driving quality but also better understands driving scenes, answers related questions, and generates plug-and-play memories that help other models with driving-related tasks.

Our contributions are as follows: (1) We introduce AD-Memo, the first general-purpose VLA driving agent that uses language-based, in-episode memory, which is scalable, explainable, and portable; (2) we propose *Da Capo*, a novel semi-closed-loop RL algorithm that combines the fine-grained, per-step driving reward signal with the future-dependent, per-episode memory reward signal, overcoming exposure bias in the agent's memory while avoiding the expensive environment simulation required for fully closed-loop RL training; (3) we build novel agentic data pipelines and curate challenging, memory-dependent datasets; (4) we evaluate AD-Memo across multiple driving scenarios and empirically demonstrate that AD-Memo can not only improve driving performance, but also aid scene understanding and serve as a plug-and-play module for driving-related reasoning tasks.

2 Preliminaries

Driving Vision-Language-Action (VLA) Models. In this paper, we consider VLA driving agents with three input modalities: (1) *past trajectory* $\tau^{\text{past}} = (\tau^1, \tau^2, \ldots, \tau^{M_1})$, a special token sequence describing the waypoints of the past ego trajectory; (2) *visual input* $I = (I^1, I^2, \ldots, I^{M_2})$,

a sequence of past camera images ordered chronologically; and (3) *language input* $L = (L^1, L^2, \ldots, L^{M_3})$, a sequence of language tokens. The agent π_θ first outputs a Chain-of-Thought (CoT) (Wei et al., 2022) and then a future trajectory τ^{future}. We let y denote the combination of CoT and τ^{future}. The primary goal of the agent is to minimize the difference between the human expert's future trajectory τ^{gold} and its prediction τ^{future} (see Appendix B.1 for detailed metrics) on video *clips* $V = \{(\tau_1^{\text{past}}, I_1, L_1, y_1), (\tau_2^{\text{past}}, I_2, L_2, y_2), \ldots, (\tau_n^{\text{past}}, I_n, L_n, y_n)\}$, each with n steps corresponding to temporally ordered frames. Besides driving, we also expect the model to develop a good understanding of the scene and support multiple tasks. Thus, we consider two other types of tasks in this paper: (1) *memory writing*, where the model takes a single image I and past memory L as input and outputs plug-and-play language memory y that helps other Vision-Language Models (VLMs) understand the driving scene; (2) *Visual Question Answering (VQA)*, where the model takes the same visual input I as in driving, optional language input L, and a natural-language question Q, then generates a language answer y^{VQA} about the driving history.

Open-Loop and Closed-Loop RL for VLAs. Open-loop RL (Zhou et al., 2025b; Li et al., 2026d) is an RL training paradigm in which the action at the current step does not affect future environmental states in the trajectory, i.e., ground-truth inputs from expert demonstrations are provided at every step. In contrast, closed-loop RL (Shi et al., 2026b) resembles MuJoCo-based RL tasks (Todorov et al., 2012; Yan et al., 2024), in which the future state depends on the current action. While the latter can better address exposure bias (Bengio et al., 2015) and enable more effective exploration (Li et al., 2026a), the former remains a prevalent choice in autonomous driving (Zhou et al., 2025b; Sheng et al., 2026) and robotics (Huang et al., 2025; 2026b) due to the difficulty and high cost of simulating real-world environments (Wang et al., 2024; Choi et al., 2026). Our work strikes a balance between the two with semi-closed-loop RL, which combines open-loop replay of visual input I and past trajectory τ^{past} with closed-loop memory y generated from previous steps.

Group Relative Policy Optimization (GRPO). GRPO (Shao et al., 2024) is a prevalent Reinforcement Learning with Verifiable Rewards (RLVR) method for building Large Language Model (LLM) agents. Given an input x ($x = (\tau^{\text{past}}, I, L)$ for driving, (I, L, Q) for question-answering, and (I, L) for memory writing) and a group of model responses $y_1, y_2, \ldots, y_n$, GRPO optimizes the policy π_θ by minimizing

$$-\frac{1}{n}\sum_{i=1}^{n}\frac{1}{|y_i|}\sum_{j=1}^{|y_i|}\min\big[\rho_{i,j}A_i,\ \text{clip}(\rho_{i,j}, 1-\epsilon_1, 1+\epsilon_2)A_i\big] + \beta\,\text{KL}(\pi_\theta\|\pi_{\text{old}}), \tag{1}$$

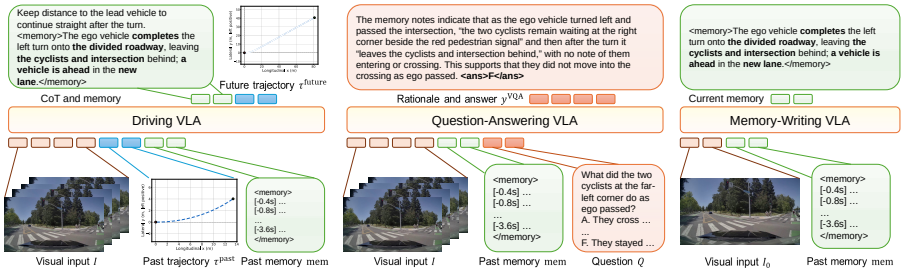
where $\rho_{i,j} = \frac{\pi_\theta(y_{i,j}|x, y_{i,<j})}{\pi_{\theta_{\text{old}}}(y_{i,j}|x, y_{i,<j})}$, $\beta \geq 0$ and $\epsilon_2 > \epsilon_1 > 0$ following DAPO (Yu et al., 2025a). Here, $\pi_{\theta_{\text{old}}}$ is the old policy used to generate responses in the current sampling-training iteration, and π_{old} is the fixed reference policy before GRPO training. The advantage A_i is calculated relative to the group mean as $A_i = \frac{r(x,y_i) - \text{avg}(r(x,y_k))}{\text{std}(r(x,y_k))}$, where avg and std are the mean and standard deviation over $k \in \{1, 2, \ldots, n\}$.

3 METHODOLOGY

The proposed AD-Memo, a general-purpose VLA driving agent, learns driving, question-answering and memory-writing jointly. Sec. 3.1 describes how the language-based memory works, while Sec. 3.2 describes our training recipe.

3.1 LANGUAGE-BASED MEMORY

At a high level, AD-Memo is a *streaming video agent* that uses the current visual input to summarize objects of interest that could affect future driving. This design differs fundamentally from existing VLA approaches with language-based memory in robotics (Haresh et al., 2026; Torne et al., 2026), where notes mainly record the agent's state and subgoals. This is because *in robotics, objects are often relatively stable but tasks are heterogeneous:* For example, making a cup of tea requires sequentially completing very different tasks, such as grabbing the cup, boiling water, and pouring from the kettle, but these objects do not move by themselves. In contrast, *in autonomous driving, objects are often transient but tasks are homogeneous:* pedestrians ahead may disappear from view within seconds, but the task remains to output a safe and efficient trajectory — regardless of



(a) Driving (b) Question-Answering (c) Memory-Writing

Figure 2: An illustration of supervised training data of AD-Memo; the three tasks listed in panel (a), (b) and (c) are jointly trained in the SFT stage (all-way stop dataset only has (a) and (b)). Driving and memory-writing data are one entry per step, while question-answering data is one entry per clip.

whether the agent passed an intersection a minute ago. Thus, in this work, we focus on a memory horizon of around 10 seconds, which, as shown in Appendix C.1 largely suffices for many scenarios of interest, but remains expensive to cover with visual input alone, requiring at least 100 images at 10Hz. We discuss the potential need for even longer-term memory, e.g., remembering speed limits, in Appendix F.

To expand AD-Memo's memory to the horizon discussed above, we divide the memory horizon by the visual input horizon t, and generate language-based memory for each interval such that the content of the memory covers the video without overlap. More specifically, suppose we have a driving clip V with $n+1$ keyframes spaced t seconds apart.[1] On those keyframes, AD-Memo switches between two modes: driving and VQA. At the i-th keyframe ($i \in \{1, 2, \ldots, n\}$), AD-Memo operates in driving mode, in which it generates Chain-of-Thought (CoT) (Wei et al., 2022) and a future trajectory τ_i^{future} for driving control. Within CoT, AD-Memo outputs *memory* mem_i, which is a formatted substring appended to the input of future keyframes and can be also used by other models as input. At the $(n+1)$-th keyframe, i.e., the end of the clip, AD-Memo switches to VQA mode, in which it takes the current visual input, memories from a window of up to T previous steps $\{\text{mem}_{\max(1,n+1-T)}, \ldots, \text{mem}_n\}$, and a question Q, then generates an answer. The question concerns a crucial object of interest that affects driving in the video (e.g., a pedestrian or traffic light) and is presented as a Multiple-Choice Question (MCQ) for verifiability. As AD-Memo does not have access to Q during driving, it must identify and record all important objects on the road to answer the question, which also helps its VLM backbone to reason about and improve driving. When deployed on the road, AD-Memo remains in driving mode but continues recording important objects on the road, as it does when preparing for questions during training.

As no public autonomous driving dataset with language-based memory annotations exists, we curate two datasets from NVIDIA's Physical-AV dataset (NVIDIA Corporation, 2026): the all-way stop dataset (Sec. 4.1) and the general driving dataset (Sec. 4.2), where the former provides a task-specific setting in which memory is beneficial, while the latter serves to evaluate generalizability. Each dataset is divided into three splits: the SFT training set $\mathcal{D}_{\text{SFT}}$, the RL training set $\mathcal{D}_{\text{RL}}$, and the test set $\mathcal{D}_{\text{test}}$, in a $3:1:1$ ratio. Each dataset consists of triplets (V, Q, y^{VQA}), which are the clip itself $V = \{(\tau_1^{\text{past}}, I_1, L_1, y_1), \ldots, (\tau_n^{\text{past}}, I_n, L_n, y_n)\}$, the accompanying question Q, and the answer to the question y^{VQA}. See Appendix C for details on dataset curation and statistics.

3.2 Two-Stage Training Recipe

AD-Memo uses a two-stage training recipe: Supervised Fine-Tuning (SFT) on the SFT training set $\mathcal{D}_{\text{SFT}}$, and semi-closed loop Reinforcement Learning (RL) on the RL training set $\mathcal{D}_{\text{RL}}$.

Supervised Fine-Tuning (SFT). This is the main stage for equipping the VLA model with memory-writing and memory-dependent reasoning abilities, during which the model jointly learns three

[1] In this paper, we focus on performance at keyframes; for intermediate frames, we can simply reuse the memory input used at the previous keyframe and discard memory generated at non-keyframes.

tasks: driving while writing memory, answering VQA questions based on its memory, and standalone memory writing, as described in Sec. 3.1. The training loss for this stage is

$$-\mathbb{E}_{(V,Q,y^{\text{VQA}})\sim\mathcal{D}_{\text{SFT}}}\left[\sum_{i=1}^{n}(\ell_{1,i}+c_1\ell_{2,i})+c_2\ell_3\right], \quad \text{where}$$
$$\ell_{1,i}=\log\pi_\theta(y_i^{\text{drive}}|x_i),\ \ell_{2,i}=\log\pi_\theta(\text{mem}_i|I_i^0,L_i),\ \ell_3=\log\pi_\theta(y^{\text{VQA}}|I_{n+1},L_{n+1},Q). \tag{2}$$

In Equation 2, I_i contains the past few frames at step i, I_i^0 is the current single frame ($I_i^0 \in I_i$), $L_i = \{\text{mem}_{i-T},\ldots,\text{mem}_{i-1}\}$ is the memory from prior steps, $y_i^{\text{drive}} = (\text{CoT}_i, \text{mem}_i, \tau^{\text{future}})$ is the driving task output, mem_i is the current memory, Q is the question, and y^{VQA} is the response for the VQA task. $c_1, c_2 \geq 0$ are constants. See examples of driving, question-answering and memory-writing data in Fig. 2. For both datasets, we train 2 epochs as discussed in Appendix D.10.

Semi-Closed-Loop RL. While our SFT stage effectively equips the VLA agent with memory-related abilities, one key shortcoming remains: exposure bias. When driving and answering driving-related questions, the model relies on its own memory from prior steps, which it does not encounter during SFT and which may be out of distribution at inference time. Closed-loop RL using the agent's own memory is a natural way to address this mismatch. However, fully closed-loop RL requires a simulator to reproduce changes in environmental inputs and the reactions of other road users. Simulators like CARLA (Dosovitskiy et al., 2017) exhibit a substantial sim-to-real appearance gap (Pasios & Nikolaidis, 2025), while 3D reconstruction-based simulators such as NuRec (Huang et al., 2026a) can be costly and still inaccurate.

To address this challenge, we propose a novel training paradigm: *semi-closed-loop RL*, which is open-loop in control and closed-loop in memory. More specifically, for a given clip with $|V| = n+1$ keyframes, we start from the first keyframe and sample K independent rollouts, each using memory generated at previous steps of the same rollout; the other inputs, such as visual observations and past trajectories, are replayed from ground truth and are therefore identical at the same step across all K rollouts. We then apply GRPO to these rollouts.

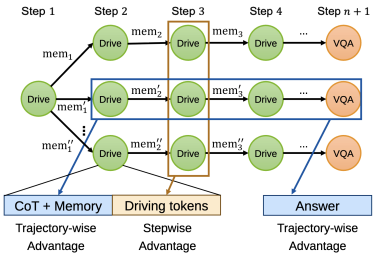


Figure 3: An illustration of semi-closed loop RL, where we keep K parallel memories; the memory and driving tokens require heterogeneous handling, with the former being trajectory-wise and the latter being stepwise.

However, the key difficulty of semi-closed-loop RL lies in computing advantages for two heterogeneous components: the driving trajectory τ^{future} and memory/VQA answers. The former does not affect future decisions in our open-loop control process, so its advantage is computed across rollouts at the same step to provide denser feedback and reduce variance; memory, in contrast, affects subsequent driving and question answering, so its advantage must account for downstream rewards, while terminal VQA answers are evaluated using the final MCQ reward (see Fig. 3 for an illustration). Considering these factors, and inspired by factored policy gradients (Spooner et al., 2021), we propose a novel RL algorithm named **D**eviation-**A**djusted **C**ausal **A**dvantage **P**olicy **O**ptimization (*Da Capo*). More specifically, for rollout $i \in \{1,\ldots,K\}$ at step $j \in \{1,\ldots,n\}$, let $r_{i,j}^{\text{Driving}}$ denote the driving reward and r_i^{VQA} the terminal MCQ reward. We define the following returns for driving trajectories, memory, and VQA answers, respectively: $G_{i,j}^{\text{Driving}} = \frac{r_{i,j}^{\text{Driving}}}{n}$, $G_{i,j}^{\text{Memory}} = \frac{1}{n}\sum_{k=j}^{n} r_{i,k}^{\text{Driving}} + \lambda^{\text{VQA}} r_i^{\text{VQA}}$, and $G_i^{\text{VQA}} = \lambda^{\text{VQA}} r_i^{\text{VQA}}$, where $\lambda^{\text{VQA}} > 0$ is a constant. The corresponding advantage $A_{i,j}^c$ for component $c \in \{\text{Driving}, \text{Memory}, \text{VQA}\}$ in rollout i at step j is:

$$A_{i,j}^c = (G_{i,j}^c - \frac{1}{K}\sum_{k=1}^{K} G_{k,j}^c)/\text{std}(G_{*,j}^c), \tag{3}$$

with the step index j omitted for VQA (i.e., $\forall i,j,\ G_{i,j}^{\text{VQA}} = G_i^{\text{VQA}}$). The name "causal advantage" reflects the fact that, without standard-deviation normalization, the resulting on-policy gradient estimator has the same expectation as one based on the trajectory-level reward, while improving credit assignment by removing conditionally independent terms from the applied advantage; for example,

as the driving tokens in step 1 do not affect the memory in step 10 with our open-loop control, reward from driving at step 1 should not be considered on memory advantage at step 10. Formally:

Claim 1. *The advantage in Equation 3, without division by std($G^c_{*,j}$), yields the same expected on-policy score-function gradient as using the group-centered trajectory-level reward* $\frac{1}{n}\sum_{s=1}^{n} r_{i,s}^{Driving} + \lambda^{VQA} r_i^{VQA}$*, also without standard-deviation normalization.*

See Appendix A for the proof. In our implementation, we set $\lambda^{\text{VQA}} = 0.5$, with r_i^{VQA} equal to 1 if the parsed answer is correct and 0 otherwise. The driving reward is $r^{\text{Driving}} = \max(-\frac{0.2}{64}\sum_{i=1}^{64} \|p_i - p_i^{\text{GT}}\|_2, -1)$, where $p_i \in \mathbb{R}^2$, $i = \{1, 2, \ldots, 64\}$, denotes a predicted 2D waypoint, with waypoints sampled every 0.1 seconds over the 6.4-second future trajectory, and p_i^{GT} is the corresponding ground-truth waypoint. We train our model for 2 epochs on the all-way stop dataset and 1 epoch on the general driving dataset. See Appendix B.3 for detailed hyperparameter choices. In contrast to "Dr. GRPO" (Liu et al., 2025), standard-deviation normalization of the advantage is vital in our setting, as it naturally accounts for differences in reward scales, e.g., 0.1m vs. 10m ADE difference between rollouts; see Appendix A for detailed rationale and Appendix D.2 for ablations.

4 Experiments

This section is organized as follows. Sec. 4.1 studies whether AD-Memo can drive well on the task-specific all-way stop scenes where memory is particularly important, as the model must remember the arrival order at the crossing when multiple cars are stopping simultaneously; Sec. 4.2 studies whether AD-Memo can handle general and diverse driving scenes. In both sections, we study whether AD-Memo better understands the driving scene by better answering related VQA questions, and whether our semi-closed loop RL algorithm, *Da Capo* can further improve our model. Finally, in Sec. 4.3, we study whether the memory produced by AD-Memo can be portable to other models.

Baseline Settings. For a fair comparison, unless otherwise specified, all models are based on the VLA backbone of a 2B, single camera-focused checkpoint of Alpamayo 2 (NVIDIA, 2026) (i.e., no action experts; see Appendix F for details). We mainly test four baselines: 1) the *base model* mentioned above that has not been further finetuned; 2) *no memory*, which goes through the same training recipe but without memory; 3) *CoT-as-memory*, which goes through the same training recipe but uses the original past CoT text as memory; 4) To provide a standard for the performance, we also test NVIDIA's recently published *flagship driving model, Alpamayo 2 Super 34B*.

4.1 All-Way Stop

Evaluation Settings. For each of the 2479 clips in the test set, following prior work (Tan et al., 2026), we generate $K = 6$ parallel trajectories with different memories. To expand our evaluation basis, we calculate and report the average performance in each keyframe before the ground truth stop, which is a total of 50533 keyframes. For those keyframes, we follow existing evaluation protocols (Liang et al., 2026) and report the following metrics: (1) *minADE, Average (Avg.) ADE, and Most Likely (ML.) ADE* (Pellegrini et al., 2009), where ADE stands for Average Displacement Error. This metric measures the difference between the predicted trajectory and the ground truth human expert trajectory; (2) *Task-specific metric*, such as go Success Rate (go SR), stop Success Rate (stop SR), roll-through rate (Roll), position error (Δ_{pos}) and stop duration error (Δ_{dur}), which measures the performance of prediction stopping and moving within a particular threshold of the ground truth, the chance of not stopping at all when it should stop, and the spatial and temporal difference related to ground truth stop; (3) *MCQ Accuracy (Acc.)*, which measures the scene understanding ability of the VLA as a streaming video agent. See Appendix B.1 for details on the metrics.

Results. Tab. 1 illustrates the result, which clearly shows that AD-Memo, both after SFT and RL, outperforms all other baselines (even Alpamayo2 Super 34B) under the same training recipe, which shows the effectiveness of AD-Memo. Notably, with the reference memory, AD-Memo works much better than its SFT counterpart with self-generated memory, which shows the potential of improvement from accurate language memory, and also necessitates RL training. CoT-as-memory works similar or slightly better in driving than no memory and largely helps question answering, but shows much lower potential as the MCQ accuracy of reference and self-generated memory is almost the same for CoT-as-memory. This is because CoT focuses more on ego action and misses most objects

Table 1: The main result for all-way stop dataset, with the best results bolded. "ref. mem" stands for using "reference memory" as input; they are generated in the same way as SFT ground truth labels. ↓ means the lower the better, and vice versa. The result shows that AD-Memo works the best on both driving and question-answering tasks, and even far outperforms the flagship model, Alpamayo2 Super 34B (which has action expert and does not output logits, thus has no most likely ADE).

	minADE ↓	Avg. ADE ↓	ML. ADE ↓	Stop SR ↑	Go SR ↑	Δ_{pos} ↓	Δ_{dur} ↓	Roll ↓	MCQ Acc. ↑
Base Model	1.386	2.351	2.393	82.06%	33.74%	1.725	1.871	11.79%	0%
Alpamayo 2 Super 34B	0.993	2.293	N/A	78.75%	34.30%	2.182	1.488	15.94%	33.19%
No mem. (SFT only)	1.102	2.185	2.318	87.04%	38.59%	1.358	1.564	6.62%	33.77%
CoT mem. (SFT only)	1.096	2.171	2.296	86.89%	39.37%	1.377	1.531	6.69%	45.41%
AD-Memo (SFT only)	1.049	2.121	2.256	87.48%	40.66%	1.257	1.469	6.05%	45.99%
CoT mem. (SFT+ref. mem)	1.098	2.169	2.286	86.95%	39.45%	1.367	1.521	6.61%	45.24%
AD-Memo (SFT+ref. mem)	0.963	1.963	2.086	87.70%	44.66%	1.185	1.296	5.84%	**89.53%**
No mem. (SFT+*Da Capo*)	**0.944**	1.952	2.005	88.92%	43.76%	1.125	1.248	**5.21%**	35.32%
CoT mem. (SFT+*Da Capo*)	0.968	1.906	1.942	89.20%	44.88%	1.157	1.194	5.31%	45.87%
AD-Memo (Ours)	0.951	**1.866**	**1.915**	**89.26%**	**45.01%**	**1.070**	**1.183**	5.26%	51.66%



Figure 4: An example of how memory aids driving. The ego and the car on the left (circled in red) are both stopping. Without memory, the agent does not know whether it should move first.

on the road (Appendix C). Also, all three types of memory after RL significantly improved, which proves the effectiveness of *Da Capo*. See Appendix D.2 for more ablation on *Da Capo* components.

Qualitative example. Fig. 4 shows an example of how memory aids driving. In this case, both ego and the left car stay still at the intersection. Without memory, the agent cannot decide which car to go first. In fact, the car on the left arrives prior to ego car, which is recorded in the memory as "left car arrives at the crossing". This information does not exist in the original CoT without memory. Thus, as Fig. 5 shows, our method achieves more precise control that better fits ground truth.

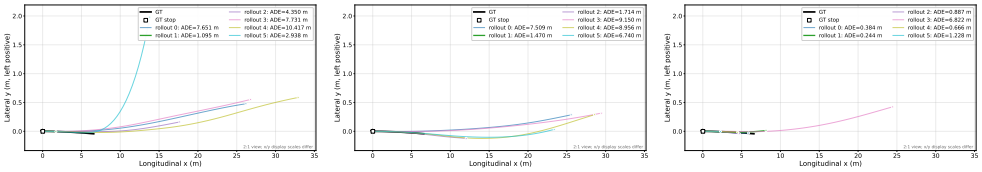
4.2 General Driving Scenes

Evaluation Settings. For general driving scenes, we use the same ADE metrics and MCQ accuracy as those in Sec. 4.1. We additionally report: (1) *minFDE, average FDE and most likely FDE* (Final Displacement Error), which measures the difference between the car's final position between the predicted and ground truth trajectory; (2) *corner distance* (See Appendix B.1 for details), which describes the discrepancy of the car's bounding box between ground truth and prediction. In total, we report the average result of 9112 clips over 145462 keyframes.

Results. Tab. 2 illustrates the result on the general driving dataset. Our method outperforms all other baselines. Notably, different from the all-way stop dataset (Sec. 4.1), RL improves average and most likely discrepancy to ground truth but not the minimum of rollouts. This is because with RL, the policy distribution is more concentrated and thus leads to higher ADE with inaccurate intention prediction. For example, the model may concentrate on predictions of turning left at the crossing, while the ground truth is turning right. In contrast, at an all-way stop crossing, the intention of passing is clear. See Appendix D.1 for an example of how our memory benefits general driving.

4.3 Memory Portability

One crucial advantage of AD-Memo is its *portability*: it can improve the model's own performance and serve as a plug-and-play enhancement for other models. To evaluate this portability, we use our SFT-trained checkpoint to process each clip sequentially, generating memory from memories



(a) No memory (b) CoT as memory (c) Ours

Figure 5: The trajectory of different methods with the input from Fig. 4. The result shows that our method can better recognize that ego should stop, which yields lower ADE results.

Table 2: The main result for the general driving dataset, where the best results are bolded; the result shows that our method performs the best. ADE, FDE and corner distance are all the lower the better; the MCQ accuracy is the higher the better.

	minADE ↓	Avg. ADE ↓	ML. ADE ↓	minFDE ↓	Avg. FDE ↓	ML. FDE ↓	Corner ↓	MCQ Acc. ↑
Base Model	1.032	1.981	2.049	2.853	5.937	6.194	1.001	0
Alpamayo 2 Super 34B	0.954	2.081	N/A	2.556	6.141	N/A	0.902	25.71%
No mem. (SFT only)	1.007	2.033	2.075	2.716	6.108	6.279	0.968	58.02%
CoT mem. (SFT only)	1.031	2.059	2.105	2.787	6.192	6.375	0.993	57.51%
AD-Memo (SFT only)	**1.005**	2.011	2.049	**2.705**	6.042	6.196	**0.967**	**65.63%**
CoT mem. (SFT+ref. mem)	1.034	2.064	2.105	2.795	6.207	6.373	0.995	57.36%
AD-Memo (SFT+ref. mem)	1.012	2.014	2.051	2.734	6.053	6.208	0.974	67.65%
No mem. (SFT+*Da Capo*)	1.090	1.883	1.891	3.042	5.583	5.648	1.062	57.43%
CoT mem. (SFT+*Da Capo*)	1.105	1.907	1.918	3.071	5.631	5.707	1.079	56.94%
AD-Memo (Ours)	1.091	**1.864**	**1.869**	3.027	**5.512**	**5.576**	1.066	65.51%

generated at previous frames and the current frame. We then ask GPT-5.6 Luna to answer questions from LingoQA (Marcu et al., 2024) and WaymoQA (Yu et al., 2025b) (see Appendix B.2), using only the final frame and the generated memory as context. This setup allows the answering model to access information from earlier frames through memory generated by our model. The results in Tab. 3 show that this memory improves another model's VQA performance in a zero-shot setting.

What matters for language memories? When we curate "ground truth / reference memory" as SFT training labels, we ask the model to review the memories generated for each frame, and *merge* them to keep track of the same object (See Appendix C.2 for details). To verify the importance of this merging step, we train our checkpoint on the memory-writing task with merged memory and unmerged memory respectively, and test the portability of the memory generated by such checkpoint. The result is illustrated in Tab. 3, which clearly shows that unmerged memory performs significantly worse.



Figure 6: An image paired with the question "What is the color of traffic lights?" Repeated statements in the unmerged memory that "the ego vehicle is now stopped" mislead GPT-5.6 Luna into incorrectly inferring that the traffic light was red.

A closer inspection in the wording of the two versions of memory suggests that the performance gap can be attributed to three reasons. First, the lack of consistent references (e.g., the lead car, the pedestrian on the left) in standalone description of individual frames caused the memory to repeatedly identify the same item, increasing more than twice in length (16.57 words vs. 51.62 words on average) than the merged memory, making it not only undesirable for memory, but also distracting; second, standalone memory contains 97-98% fewer temporal words such as "continue" and "remain", reducing the temporal information of the objects; finally, as shown in Fig. 6, when the standalone memory repeatedly states "ego vehicle is stopped", the model can be misled and determine that the color of traffic lights were mostly red. Thus, *temporal information in memory is a key to success.*

Table 3: The accuracy of GPT-5.6 Luna on LingoQA and WaymoQA, which shows that our memory can enhance the scene understanding of other models in a plug-and-play manner.

	Our memory + last frame	Unmerged memory + last frame	Only last frame	Full video
WaymoQA	71.65%	68.3%	66.63%	75%
LingoQA	67%	60.4%	62.6%	70%

5 Related Works

VLA driving agents. Empowered by the recent success of LLMs (Singh et al., 2025; OpenAI, 2026) and Vision-Language Models (VLMs) (Li et al., 2025), VLA models (Zitkovich et al., 2023; O'Neill et al., 2024) are an emerging solution to the long-standing challenge of fully autonomous driving (Wang et al., 2025; Li et al., 2026e; Zhou et al., 2026). Compared to the conventional "perception-decision-action" modular pipeline (Urmson et al., 2009; Kato et al., 2018), Vision-Action (VA) models that map sensor input to action (Hu et al., 2023; Liao et al., 2025), and driving VLMs that only predict high-level meta-actions (Huang et al., 2024; Cui et al., 2025b), VLA driving agents utilize the reasoning abilities of their VLM backbones while producing end-to-end trajectories for control. This idea has attracted considerable attention recently (Tan et al., 2026; Fu et al., 2026; Peng et al., 2026; Li et al., 2026c). Following this idea, we make full use of VLM backbone knowledge with language memory and build AD-Memo on Alpamayo (Wang et al., 2025).

Driving agents with memory. Memory-dependent tasks pose a long-standing and important challenge in autonomous driving (Liang et al., 2026). Prior solutions mainly fall into two groups: *latent vector* (Leng et al., 2025; Song et al., 2025) and *in-context learning* (Wen et al., 2024; Mei et al., 2024; Zhang et al., 2026). The former maintains a bank of dense representation vectors extracted from past inputs. While some methods keep KV-memory bank (Shao et al., 2023) or perform test-time training (Kim & Park, 2026), the prevailing approach is to store compact tokens produced by Q-Former (Li et al., 2023; Fu et al., 2025; Zhang et al., 2025; Huang et al., 2026c), ConvLSTM (Shi et al., 2015; Gerasyov et al., 2024), or other encoders (Salazar-Gomez et al., 2024); such memory is usually accessed via cross-attention. The latter often stores structured records of prior driving experiences (e.g., scene descriptions and responses), which are added to the agent's input as prompts in a Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) manner (Cui et al., 2025a; Yao et al., 2025). While other memory methods also exist, such as directly storing past frames for reasoning agents (Zheng et al., 2026; Baimbetova et al., 2026) and maintaining knowledge graphs (Li et al., 2026b), AD-Memo is the first VLA driving agent with in-episode language memory.

VLA agents with in-episode language memory. Several works in the robotics community explore in-episode language memory in VLA agents. Notes-to-Self (Haresh et al., 2026) records grounding information, plans, and the agent's actions, and updates memory when the model determines that a subtask has been completed; MEM (Torne et al., 2026) uses language memory to record the agent's plans and actions for a high-level VLM, which in turn instructs a low-level VLA; Goal2Skill (Liu et al., 2026) and τ_0-VLA (Cai et al., 2026) are similar but incorporate additional mechanisms for verbal self-reflection and verification. AD-Memo differs from these works in two ways: 1) None of these works studies autonomous driving, where language memory remains surprisingly under-explored. This domain presents new challenges regarding memory content (Sec. 3.1); 2) None of the above works uses RL for training, whereas our work proposes a novel semi-closed-loop RL algorithm, representing a methodological advance.

6 Conclusion

In this paper, we propose AD-Memo, the first VLA driving agent with in-episode language-based memory that is not only effective but also brief, explainable, and portable. The agent outputs memory alongside CoT when driving, and the memory is then appended to the future input. To train this model, we curate multiple datasets, and propose *Da Capo*, a novel semi-closed-loop RL algorithm that replays ground-truth driving states while using the agent's self-generated memory. We empirically demonstrate that AD-Memo improves driving performance, performs better on VQA tasks for scene understanding, and produces memory portable to other models. We believe that AD-Memo offers a timely and effective solution for memory-dependent tasks in autonomous driving.

AI USE STATEMENT

In this work, we used generative AI tools for synthesizing / formatting datasets (especially memory and VQA questions), implementing methods, and supporting data analysis / interpreting results by running AI-generated scripts / drafting and verifying mathematical claims. We have not used generative AI tools for translation assistance or developing theoretical models / conceptual frameworks. Additionally, we used generative AI tools for creating scientific figures, editing paper to improve readability, identifying / summarizing related literature, and early brainstorming. We have reviewed all AI-assisted work. For datasets and experiment results, we check via detailed visualization; for the correctness of the code, we check by reading the code; for mathematical results, we checked carefully and rewrote it into a form easier to understand; for literature, we check via searching each paper online manually. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This paper fully conforms to the ICLR 2027 Code of Ethics statement, and does not raise question regarding it.

REPRODUCIBILITY STATEMENT

We describe the detail of how we curate our dataset in Appendix C. The experiment metric, external benchmarks and hyperparameters are described in Appendix B.1, Appendix B.2, and Appendix B.3 respectively. We also report our computational resource consumption in Appendix E.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Aidana Baimbetova, Haruki Yonekura, Hamada Rizk, and Hirozumi Yamaguchi. Pedestrian-aware llm-driven behavioral planning for autonomous vehicles. In *ITSC*, 2026.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, 2015.

Xiaowei Cai, Yunuo Cai, Bingao Chen, Jingxiao Chen, Zhi Chen, Siyuan Feng, Tengyu Hou, Jingshun Huang, Han Jiang, Runkun Ju, et al. τ_0-vla: a hierarchical robot foundation model with world-model-guided test-time computation. *arXiv preprint arXiv:2608.16885*, 2026.

Hyeonbeom Choi, Daechul Ahn, Youhan Lee, Taewook Kang, Seongwon Cho, and Jonghyun Choi. Scale: Self-uncertainty conditioned adaptive looking and execution for vision-language-action models. In *ICML*, 2026.

Can Cui, Zichong Yang, Yupeng Zhou, Juntong Peng, Sung-Yeon Park, Cong Zhang, Yunsheng Ma, Xu Cao, Wenqian Ye, Yiheng Feng, Jitesh Panchal, Lingxi Li, Yaobin Chen, and Ziran Wang. Onboard vision-language models (vlms) for personalized motion control of autonomous vehicles. In *IROS*, 2025a.

Erfei Cui, Wenhai Wang, Zhiqi Li, Jiangwei Xie, Haoming Zou, Hanming Deng, Gen Luo, Lewei Lu, Xizhou Zhu, and Jifeng Dai. Drivemlm: aligning multi-modal large language models with behavioral planning states for autonomous driving. *Visual Intelligence*, 2025b.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *CoRL*, 2017.

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 *ICCV*, 2025.

Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Hongwei Xie, Bing Wang, Guang Chen, Hangjun Ye, Dingkang Liang, and Xiang Bai. Minddrive: A vision-language-action model for autonomous driving via online reinforcement learning. In *ECCV*, 2026.

Matvey Gerasyov, Andrey V. Savchenko, and Ilya Makarov. Enhancing autonomous driving with spatial memory and attention in reinforcement learning. *IEEE Access*, 2024.

Sanjay Haresh, Daniel Dijkman, Apratim Bhattacharyya, and Roland Memisevic. Notes-to-self: Scratchpad augmented vlas for memory dependent manipulation tasks. In *ICRA*, 2026.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *CVPR*, 2023.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Frank Wang, and Fu-En Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning. In *NeurIPS*, 2025.

Jiahui Huang, Jiawei Ren, Michal Tyszkiewicz, Bjoern Haefner, Michael Shelley, Xin Kang, Seung Wook Kim, Ning Xu, Qi Wu, Janick Martinez Esturo, et al. Instant nurec: Feed-forward 3d gaussian reconstruction for driving scene simulation. *arXiv preprint arXiv:2607.14203*, 2026a.

Ting Huang, Yue Huang, Zeyu Zhang, Shuicheng Yan, and Hao Tang. Mobilevla-r1 2.0: Rl-enhanced reasoning for mobile robot control. *arXiv preprint arXiv:2609.06251*, 2026b.

Yidong Huang, Jacob Sansom, Ziqiao Ma, Felix Gervits, and Joyce Chai. Drivlme: Enhancing llm-based autonomous driving agents with embodied and social experiences. In *IROS*, 2024.

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

Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *ICCPS*, 2018.

Donghyun Kim and Jaehyoung Park. Think as needed: Geometry-driven adaptive perception for autonomous driving. *arXiv preprint arXiv:2605.10117*, 2026.

Sanjay Kumar, Tim Brophy, Reenu Mohandas, Eoin Martino Grua, Ganesh Sistu, Valentina Donzella, and Ciaran Eising. Occluded nuscenes: A multi-sensor dataset for evaluating perception robustness in automated driving. *arXiv preprint arXiv:2510.18552*, 2025.

Kristofer D Kusano, John M Scanlon, Yin-Hsiu Chen, Timothy L McMurry, Tilia Gode, and Trent Victor. Comparison of waymo rider-only crash rates by crash type to human benchmarks at 56.7 million miles. *Traffic Injury Prevention*, 2025.

Ziyang Leng, Jiawei Yang, Wenlong Yi, and Bolei Zhou. Occupancy learning with spatiotemporal memory. In *ICCV*, 2025.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*, 2020.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. In *ICLR*, 2026a.

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 *ICML*, 2023.

Yijian Li, Xiangru Mu, Changze Li, Hantian Shi, Jiyuan Cai, Jia Cai, Xiaoxue Liu, Yajing Sun, Ming Yang, and Tong Qin. Vln-avp: Zero-shot vision-language navigation with hybrid long-short-term memory for autonomous valet parking. *arXiv preprint arXiv:2607.17767*, 2026b.

Yingyan Li, Shuyao Shang, Weisong Liu, Bing Zhan, Haochen Wang, Yuqi Wang, Yuntao Chen, Xiaoman Wang, Yasong An, Chufeng Tang, et al. Drivevla-w0: World models amplify data scaling law in autonomous driving. In *ICLR*, 2026c.

Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, Kun Ma, et al. Recogdrive: A reinforced cognitive framework for end-to-end autonomous driving. In *ICLR*, 2026d.

Yongkang Li, Lijun Zhou, Sixu Yan, Bencheng Liao, Tianyi Yan, Kaixin Xiong, Long Chen, Hongwei Xie, Bing Wang, Guang Chen, et al. Unidrivevla: Unifying understanding, perception, and action planning for autonomous driving. *arXiv preprint arXiv:2604.02190*, 2026e.

Zongxia Li, Xiyang Wu, Hongyang Du, Fuxiao Liu, Huy Nghiem, and Guangyao Shi. A survey of state of the art large vision language models: Alignment, benchmark, evaluations and challenges. In *CVPRW*, 2025.

Zhixuan Liang, Yuxiao Chen, Yurong You, Peter Karkus, Wenhao Ding, Boyi Li, Alexander Popov, Yan Wang, Maximilian Igl, Yiming Li, et al. Planning-aligned token compression for long-context autonomous driving. *IEEE Robotics and Automation Letters*, 2026.

Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, et al. Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. In *CVPR*, 2025.

Liangkai Liu, Yanzhi Wang, and Weisong Shi. Understanding time variations of dnn inference in autonomous driving. *arXiv preprint arXiv:2209.05487*, 2022.

Zhen Liu, Xinyu Ning, Zhe Hu, Xinxin Xie, Weize Li, Zhipeng Tang, Chongyu Wang, Zejun Yang, Hanlin Wang, Yitong Liu, et al. Goal2skill: Long-horizon manipulation with adaptive planning and reflection. *arXiv preprint arXiv:2604.13942*, 2026.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *COLM*, 2025.

Ana-Maria Marcu, Long Chen, Jan Hünermann, Alice Karnsund, Benoit Hanotte, Prajwal Chidananda, Saurabh Nair, Vijay Badrinarayanan, Alex Kendall, Jamie Shotton, et al. Lingoqa: Visual question answering for autonomous driving. In *ECCV*, 2024.

Tekedra Mawakana and Dmitri Dolgov. Accelerating our global growth: Waymo raises $16 billion investment round, 2026. URL https://waymo.com/blog/2026/02/waymo-raises-usd16-billion-investment-round/.

Jianbiao Mei, Yukai Ma, Xuemeng Yang, Licheng Wen, Xinyu Cai, Xin Li, Daocheng Fu, Bo Zhang, Pinlong Cai, Min Dou, et al. Continuously learning, adapting, and improving: A dual-process approach to autonomous driving. In *NeurIPS*, 2024.

NVIDIA. Alpamayo 2 Super. Hugging Face, 2026. URL https://huggingface.co/nvidia/Alpamayo2-Super. Model card.

NVIDIA Corporation. Physicalai-autonomous-vehicles dataset. https://huggingface.co/datasets/nvidia/PhysicalAI-Autonomous-Vehicles, 2026. NVIDIA Physical AI Autonomous Vehicles Dataset, version 26.03.

OpenAI. Ten advances in mathematics and theoretical computer science. https://cdn.openai.com/pdf/ten-proofs-oai.pdf, 2026.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration. In *ICRA*, 2024.

Stefanos Pasios and Nikos Nikolaidis. Carla2real: A tool for reducing the sim2real appearance gap in carla simulator. *IEEE Transactions on Intelligent Transportation Systems*, 2025.

Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *ICCV*, 2009.

Zhenghao Peng, Wenhao Ding, Yurong You, Yuxiao Chen, Wenjie Luo, Thomas Tian, Yulong Cao, Apoorva Sharma, Danfei Xu, Boris Ivanovic, et al. Counterfactual vla: Self-reflective vision-language-action model with adaptive reasoning. In *CVPR*, 2026.

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 *AAAI*, 2024.

Gustavo Salazar-Gomez, Wenqian Liu, Manuel Diaz-Zapata, David Sierra-Gonzalez, and Christian Laugier. Tlcfuse: Temporal multi-modality fusion towards occlusion-aware semantic segmentation. In *IEEE Intelligent Vehicles Symposium*, 2024.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *NIPS*, 2015.

Hao Shao, Letian Wang, Ruobing Chen, Steven L Waslander, Hongsheng Li, and Yu Liu. Reasonnet: End-to-end driving with temporal and global reasoning. In *CVPR*, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Zihao Sheng, Xin Ye, Jingru Luo, Sikai Chen, and Liu Ren. Explorevla: Dense world modeling and exploration for end-to-end autonomous driving. In *ECCV*, 2026.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *ICLR*, 2026a.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, 2015.

Yunxiao Shi, Hong Cai, Mohammad Ghavamzadeh, and Fatih Porikli. Clear: Closed-loop reinforcement learning at scale for end-to-end autonomous driving. *arXiv preprint arXiv:2607.02841*, 2026b.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Ziying Song, Caiyan Jia, Lin Liu, Hongyu Pan, Yongchang Zhang, Junming Wang, Xingyu Zhang, Shaoqing Xu, Lei Yang, and Yadan Luo. Don't shake the wheel: Momentum-aware planning in end-to-end autonomous driving. In *CVPR*, 2025.

Thomas Spooner, Nelson Vadori, and Sumitra Ganesh. Factored policy gradients: Leveraging structure for efficient learning in momdps. In *NeurIPS*, 2021.

Shuhan Tan, Kashyap Chitta, Yuxiao Chen, Ran Tian, Yurong You, Yan Wang, Wenjie Luo, Yulong Cao, Philipp Krähenbühl, Marco Pavone, et al. Latent chain-of-thought world modeling for end-to-end autonomous driving. In *CVPR*, 2026.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, et al. Mem: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026.

Chris Urmson, Chris Baker, John Dolan, Paul Rybski, Bryan Salesky, William Whittaker, Dave Ferguson, and Michael Darms. Autonomous driving in traffic: Boss and the urban challenge. *AI Magazine*, 2009.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models. In *ICLR*, 2024.

Yan Wang, Wenjie Luo, Junjie Bai, Yulong Cao, Tong Che, Ke Chen, Yuxiao Chen, Jenna Diamond, Yifan Ding, Wenhao Ding, et al. Alpamayo-r1: Bridging reasoning and action prediction for generalizable autonomous driving in the long tail. *arXiv preprint arXiv:2511.00088*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

Licheng Wen, Daocheng Fu, Xin Li, Xinyu Cai, Tao Ma, Pinlong Cai, Min Dou, Botian Shi, Liang He, and Yu Qiao. Dilu: A knowledge-driven approach to autonomous driving with large language models. In *ICLR*, 2024.

Shaoyuan Xie, Lingdong Kong, Yuhao Dong, Chonghao Sima, Wenwei Zhang, Qi Alfred Chen, Ziwei Liu, and Liang Pan. Are vlms ready for autonomous driving? an empirical study from the reliability, data and metric perspectives. In *ICCV*, 2025.

Runsheng Xu, Hubert Lin, Wonseok Jeon, Hao Feng, Yuliang Zou, Liting Sun, John Gorman, Kate Tolstaya, Sarah Tang, Brandyn White, et al. Wod-e2e: Waymo open dataset for end-to-end driving in challenging long-tail scenarios. In *CVPR*, 2026.

Kai Yan, Alexander G Schwing, and Yu-Xiong Wang. Reinforcement learning gradients as vitamin for online finetuning decision transformers. In *NeurIPS*, 2024.

Huaiyuan Yao, Pengfei Li, Bu Jin, Yupeng Zheng, An Liu, Lisen Mu, Qing Su, Qian Zhang, Yilun Chen, and Peng Li. Lilodriver: A lifelong learning framework for closed-loop motion planning in long-tail autonomous driving scenarios. *arXiv preprint arXiv:2505.17209*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. In *NeurIPS*, 2025a.

Seungjun Yu, Seonho Lee, Namho Kim, Jaeyo Shin, Junsung Park, Wonjeong Ryu, Raehyuk Jung, and Hyunjung Shim. Waymoqa: A multi-view visual question answering dataset for safety-critical reasoning in autonomous driving. *arXiv preprint arXiv:2511.20022*, 2025b.

Tong Zeng, Longfeng Wu, Liang Shi, Dawei Zhou, and Feng Guo. Are vision llms road-ready? a comprehensive benchmark for safety-critical driving video understanding. In *SIGKDD*, 2025a.

Xiangyu Zeng, Kefan Qiu, Qingyu Zhang, Xinhao Li, Jing Wang, Jiaxin Li, Ziang Yan, Kun Tian, Meng Tian, Xinhai Zhao, et al. Streamforest: Efficient online video understanding with persistent event memory. In *NeurIPS*, 2025b.

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 *ICCV*, 2025.

Yi Zhang, Xian Zhang, Saisi Zhao, Yinglei Song, Chengdong Wu, Nenad Petrovic, and Alois Knoll. Pram-r: A perception-reasoning-action-memory framework with llm-guided modality routing for adaptive autonomous driving. *arXiv preprint arXiv:2603.04222*, 2026.

Weicheng Zheng, Xiaofei Mao, Nanfei Ye, Pengxiang Li, Kun Zhan, Xianpeng Lang, and Hang Zhao. Driveagent-r1: Advancing vlm-based autonomous driving with active perception and hybrid thinking. In *ICLR*, 2026.

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

Xingcheng Zhou, Konstantinos Larintzakis, Hao Guo, Walter Zimmer, Mingyu Liu, Hu Cao, Jiajie Zhang, Venkatnarayanan Lakshminarasimhan, Leah Strand, and Alois C Knoll. Tumtraffic-videoqa: A benchmark for unified spatio-temporal video understanding in traffic scenes. *arXiv preprint arXiv:2502.02449*, 2025a.

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 *NeurIPS*, 2025b.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, 2023.

APPENDIX: VISION-LANGUAGE-ACTION AUTONOMOUS DRIVING AGENT WITH LANGUAGE-BASED MEMORY

The appendix is organized as follows: in Sec. A, we provide discussion and proofs for our novel semi-closed loop algorithm, *Da Capo*; in Sec. B, we discuss the details on our evaluation metrics (Sec. B.1), benchmark adopted (Sec. B.2), and hyperparameters (Sec. B.3); in Sec. C, we introduce our dataset curation pipeline, statistics and examples of memory and VQA questions in our dataset; in Sec. D, we provide a broad range of qualitative and quantitative ablations; in Sec. E, we describe the computational resource for this work; finally, in Sec. F, we discuss the limitation and potential future work of our AD-Memo.

A MATHEMATICAL PROOFS

To see the equivalence between the expected gradient of *Da Capo* without standard deviation and the gradient of GRPO with trajectory-level reward, we start from an intuitive example. Suppose we have a 3-step trajectory with memory $\text{mem}_1, \text{mem}_2, \text{mem}_3$, predicted trajectories τ_1, τ_2, τ_3, driving rewards $r_1^{\text{Driving}}, r_2^{\text{Driving}}, r_3^{\text{Driving}}$, and a subsequent VQA reward r^{VQA} with $\lambda^{\text{VQA}} = 1$. With a trajectory-level return, without baseline subtraction or advantage normalization, the contribution of mem_2 to the policy gradient is

$$\mathbb{E}_{\pi_\theta}\left[\left(\frac{1}{3}\left(r_1^{\text{Driving}} + r_2^{\text{Driving}} + r_3^{\text{Driving}}\right) + r^{\text{VQA}}\right)\nabla_\theta \log \pi_\theta(\text{mem}_2 \mid c_2)\right], \tag{4}$$

where c_2 consists of the second-step input, including mem_1, and the second-step CoT output.

Let H denote the common rollout prefix containing the first-step input, CoT output, and mem_1, before generating τ_1 and the second-step continuation. The latter uses mem_1 and the fixed second-step environmental input, but not τ_1. With independent sampling randomness for the two branches, τ_1 and (c_2, mem_2) are therefore *conditionally independent* given H. Since r_1^{Driving} is determined by τ_1 and the fixed reward references, $r_1^{\text{Driving}} \perp (c_2, \text{mem}_2) \mid H$.

By the score-function identity and the law of total expectation,

$$\begin{aligned}
&\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(\text{mem}_2 \mid c_2) \mid H\right] \\
&\quad = \mathbb{E}_{c_2|H}\left[\sum_m \pi_\theta(m \mid c_2)\nabla_\theta \log \pi_\theta(m \mid c_2)\right] \\
&\quad = \mathbb{E}_{c_2|H}\left[\nabla_\theta \sum_m \pi_\theta(m \mid c_2)\right] = 0,
\end{aligned} \tag{5}$$

where c_2 is held fixed in the inner derivative. Consequently, conditional independence gives

$$\begin{aligned}
&\mathbb{E}_{\pi_\theta}\left[\frac{1}{3} r_1^{\text{Driving}} \nabla_\theta \log \pi_\theta(\text{mem}_2 \mid c_2)\right] \\
=&\frac{1}{3}\mathbb{E}_{\pi_\theta}\left[\mathbb{E}_{\pi_\theta}\left[r_1^{\text{Driving}} \mid H\right]\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(\text{mem}_2 \mid c_2) \mid H\right]\right] = 0.
\end{aligned} \tag{6}$$

Thus, removing $\frac{1}{3} r_1^{\text{Driving}}$ from the return weighting the score-function term for mem_2 preserves the expected policy gradient, consistent with Schulman et al. (2015).

More generally, fix the replayed inputs and terminal question X, and sample K independent on-policy rollouts with $n+1$ keyframes (n driving frames + 1 VQA frame). All expectations below are conditional on X; the conclusion also holds after averaging over X. Assume differentiable policy probabilities with fixed support and sufficient integrability to interchange differentiation, summation, and expectation.

For rollout i at step j, let $S_{i,j}^{\text{Memory}}$, $S_{i,j}^{\text{Driving}}$, and S_i^{VQA} denote the scores of the CoT/memory block, predicted trajectory, and terminal answer, respectively. Each score is the gradient of the corresponding conditional log-probability, summed over the block's tokens.

Define the trajectory-level return

$$R_i = \frac{1}{n}\sum_{k=1}^{n} r_{i,k}^{\text{Driving}} + \lambda^{\text{VQA}} r_i^{\text{VQA}}, \tag{7}$$

and recall the causal returns from Sec. 3.2:

$$\begin{aligned} G_{i,j}^{\text{Driving}} &= \frac{1}{n} r_{i,j}^{\text{Driving}}, \\ G_{i,j}^{\text{Memory}} &= \frac{1}{n}\sum_{k=j}^{n} r_{i,k}^{\text{Driving}} + \lambda^{\text{VQA}} r_i^{\text{VQA}}, \\ G_i^{\text{VQA}} &= \lambda^{\text{VQA}} r_i^{\text{VQA}}. \end{aligned} \tag{8}$$

Without standard-deviation normalization, the corresponding group-centered advantages are

$$A_{i,j}^{c} = G_{i,j}^{c} - \frac{1}{K}\sum_{\ell=1}^{K} G_{\ell,j}^{c}, \qquad A_i^{\text{traj}} = R_i - \frac{1}{K}\sum_{\ell=1}^{K} R_\ell, \tag{9}$$

where $c \in \{\text{Driving}, \text{Memory}, \text{VQA}\}$, with the step index j omitted for VQA.

Claim 1 (Expected-gradient equivalence)**.** *For every output block,*

$$\mathbb{E}\left[S_{i,j}^{c} A_{i,j}^{c}\right] = \mathbb{E}\left[S_{i,j}^{c} A_i^{traj}\right]. \tag{10}$$

Consequently, without standard deviation, our method and trajectory-level group centering yield the same expected on-policy score-function gradient (but different credit assignment).

Proof. For any generated block Y with conditioning input and prefix H, its score $S = \nabla_\theta \log \pi_\theta(Y \mid H)$ satisfies

$$\mathbb{E}[S \mid H] = \sum_y \pi_\theta(y \mid H)\nabla_\theta \log \pi_\theta(y \mid H) = \nabla_\theta \sum_y \pi_\theta(y \mid H) = 0. \tag{11}$$

Let $\Delta_{i,j}^{c} = R_i - G_{i,j}^{c}$ denote the discarded reward. For driving, it contains the other steps' driving rewards and the VQA reward; these are conditionally independent of the current predicted trajectory because predicted trajectories are not reused as later driving or VQA inputs. For Memory and VQA, it contains only driving rewards determined before the corresponding block is generated. Thus, in each case, $\Delta_{i,j}^{c}$ is conditionally independent of the generated block given its conditioning context H, and

$$\mathbb{E}\left[S_{i,j}^{c}\Delta_{i,j}^{c}\right] = \mathbb{E}\left[\mathbb{E}[\Delta_{i,j}^{c} \mid H]\,\mathbb{E}[S_{i,j}^{c} \mid H]\right] = 0. \tag{12}$$

For $\ell \neq i$, independence between the rollouts and $\mathbb{E}[S_{i,j}^{c}] = 0$ also give $\mathbb{E}[S_{i,j}^{c}\Delta_{\ell,j}^{c}] = 0$. Therefore,

$$\mathbb{E}\left[S_{i,j}^{c}\left(A_i^{\text{traj}} - A_{i,j}^{c}\right)\right] = \mathbb{E}\left[S_{i,j}^{c}\left(\Delta_{i,j}^{c} - \frac{1}{K}\sum_{\ell=1}^{K}\Delta_{\ell,j}^{c}\right)\right] = 0. \tag{13}$$

Summing over output blocks and averaging over rollouts completes the proof. □

Remark 1. Adding standard deviation to the advantage (i.e. the Deviation-Adjustment (DA) in *Da Capo*) generally breaks the expected-gradient equivalence in two ways. First, as standard deviation is computed individually for every step, the introduction of such denominator adds a stepwise scaling to the total reward. Second, estimating the denominator from the same rollout group can invalidate the zero-mean cancellation of conditionally independent reward terms. To see this, decompose the trajectory-level return associated with a driving output as $r_1 + r_2$, where r_2 is conditionally independent of the current trajectory tokens given their conditioning context. Although r_2 can be removed without changing the expected unnormalized gradient, its contribution after division by $\text{std}(r_1 + r_2)$ does not necessarily vanish, because the denominator also depends on the sampled output through r_1.

Despite this loss of exact equivalence, we find standard-deviation normalization empirically beneficial in our autonomous-driving setting, unlike the findings of "Dr. GRPO" (Liu et al., 2025) for mathematical reasoning. This choice is motivated by the heterogeneous reward scales across scenes and task components. For example, driving-reward differences within a rollout group can range from approximately 0.02 for a $0.1\,\mathrm{m}$ ADE difference to 1 for an ADE difference exceeding $5\,\mathrm{m}$. Normalization within each step and output component puts these heterogeneous advantage signals on a comparable scale, allowing small but meaningful driving improvements to contribute alongside larger reward differences. Our ablations show that this adaptive normalization outperforms RL with no standard deviation or a fixed normalization constant; see Sec. D.2 for ablations.

B Experiment Details

B.1 Evaluation Metrics

We list the details on the evaluation metrics used in Sec. 4 here:

- **minADE (lower is better).** Minimum Average Displacement Error (ADE) is one of the most commonly used metric in autonomous driving. More specifically, given K predicted trajectory $\tau_1 = (p_1^1, p_2^1, \ldots, p_M^1), \ldots, \tau_K = (p_1^K, p_2^K, \ldots, p_M^K)$ and ground truth trajectory $\tau = (p_1^{\text{GT}}, p_2^{\text{GT}}, \ldots, p_M^{\text{GT}})$, minADE is calculated as

$$\min_{i \in \{1,2,\ldots,K\}} \frac{1}{M} \sum_{j=1}^{M} \|p_j^i - p_j^{\text{GT}}\|_2, \tag{14}$$

  where $p_j^i, p_j^{\text{GT}} \in \mathbb{R}^2$ are waypoints on the 2D coordinate, each represents the location for every 0.1 second. Following prior works, unless otherwise specified, we have $K = 6$ and $M = 64$ (i.e. prediction of 6.4s) in our evaluation. minADE describes the behavior of the best prediction, which compensates for potential mode error (e.g. due to human intention unavailable to the agent, the car turns right when the prediction is turning left).

- **Average ADE (lower is better).** Similar to minADE, average ADE can be calculated as

$$\frac{1}{MK} \sum_{i=1}^{K} \sum_{j=1}^{M} \|p_j^i - p_j^{\text{GT}}\|_2, \tag{15}$$

  which describes the average driving performance of the VLA.

- **Most likely ADE (lower is better).** Most likely ADE describes the performance of the agent if the car is under its control. With K rollouts, we calculate the log probability of each rollout, and report the ADE $\frac{1}{M} \sum_{j=1}^{M} \|p_j^i - p_j^{\text{GT}}\|_2$ for trajectory i with the highest log probability.

- **minFDE (lower is better).** Minimum Final Displacement Error (FDE) describes the discrepancy between the final position of the predicted and ground truth trajectory. More specifically, it is calculated as

$$\min_{i \in \{1,2,\ldots,K\}} \|p_M^i - p_M^{\text{GT}}\|_2, \tag{16}$$

  i.e., only take the last waypoint into account.

- **Average FDE (lower is better).** Similar to minFDE and average ADE, average FDE is calculated as

$$\sum_{j=1}^{K} \frac{1}{K} \|p_M^j - p_M^{\text{GT}}\|_2, \tag{17}$$

- **Most likely FDE (lower is better).** Similar to most likely ADE, most likely FDE is the FDE for the trajectory with the highest log probability.

- **Corner Distance (lower is better).** Corner distance jointly measures the translation and orientation errors of the predicted ego-vehicle trajectory. Let $c^i_{j,l}$ and $c^{\mathrm{GT}}_{j,l}$ denote the l-th corner of the predicted and ground-truth ego-vehicle bounding boxes, respectively, at timestep j, where $l \in \{1, \ldots, 8\}$ for a 3D bounding box. Corner distance is calculated as

$$\frac{1}{8M} \sum_{j=1}^{M} \sum_{\ell=1}^{8} \min_{i \in \{1,\ldots,K\}} \left\| c^i_{j,\ell} - c^{\mathrm{GT}}_{j,\ell} \right\|_2 . \tag{18}$$

  The corners are obtained from the predicted vehicle position and orientation using a fixed-size ego-vehicle bounding box.

- **Stop Success rate (SR, higher is better).** We first detect a stop as a contiguous interval during which the vehicle speed is below a threshold v_{stop}. Only examples containing a valid ground-truth stop are included in the evaluation. A prediction is considered a successful stop if it contains a valid predicted stop and its stop-start time $t^{\mathrm{pred}}_{\mathrm{stop}}$ satisfies

$$t^{\mathrm{GT}}_{\mathrm{stop}} - 2\epsilon_{\mathrm{stop}} \le t^{\mathrm{pred}}_{\mathrm{stop}} \le t^{\mathrm{GT}}_{\mathrm{stop}} + \epsilon_{\mathrm{stop}}. \tag{19}$$

  Stop SR is the fraction of eligible predictions satisfying this condition. Unless otherwise specified, following prior work (Liang et al., 2026), we use $v_{\mathrm{stop}} = 0.5\,\mathrm{m/s}$, a minimum stop duration of $0.2\,\mathrm{s}$, and $\epsilon_{\mathrm{stop}} = 0.7\,\mathrm{s}$. Notably, only when a keyframe is before the ground truth stop time will we consider it as eligible for metric computation as reported in Sec. 4.1.

- **Go Success rate (SR, higher is better).** For a ground-truth trajectory that departs after stopping, a prediction is considered successful if it contains a valid stop and its stop-end time is within the specified tolerance of the ground-truth stop-end time:

$$t^{\mathrm{GT}}_{\mathrm{go}} - \epsilon_{\mathrm{early}} \le t^{\mathrm{pred}}_{\mathrm{go}} \le t^{\mathrm{GT}}_{\mathrm{go}} + \epsilon_{\mathrm{late}}. \tag{20}$$

  If the ground-truth vehicle remains stopped until the end of the prediction horizon, the prediction is successful only if it also remains stopped. Go SR is the fraction of eligible predictions satisfying the corresponding condition. We use $\epsilon_{\mathrm{early}} = \epsilon_{\mathrm{late}} = 0.7\,\mathrm{s}$ following prior work (Liang et al., 2026).

- **Roll-through Rate (Roll, lower is better).** A prediction is classified as a roll-through if the ground-truth trajectory contains a valid stop but the predicted trajectory contains no valid stop interval. Roll-through rate is the fraction of eligible predictions classified as roll-throughs.

- **Position Error (Δ_{pos}, lower is better).** Position error measures the spatial discrepancy between the predicted and ground-truth stopping locations. Let $p^{\mathrm{GT}}_{\mathrm{stop}}$ denote the ground-truth stopping position and $p^{\mathrm{pred}}_{\mathrm{stop}}$ the position at which the predicted stop begins. It is calculated as

$$\Delta_{\mathrm{pos}} = \left\| p^{\mathrm{pred}}_{\mathrm{stop}} - p^{\mathrm{GT}}_{\mathrm{stop}} \right\|_2 . \tag{21}$$

  If no valid predicted stop is detected, the final predicted waypoint is used as $p^{\mathrm{pred}}_{\mathrm{stop}}$. We report the mean position error in meters over eligible examples.

- **Stop Duration Error (Δ_{dur}, lower is better).** Let $d^{\mathrm{pred}}_{\mathrm{stop}}$ and $d^{\mathrm{GT}}_{\mathrm{stop}}$ denote the durations of the predicted and ground-truth stop intervals, respectively. Stop duration error is calculated as

$$\Delta_{\mathrm{dur}} = \left| d^{\mathrm{pred}}_{\mathrm{stop}} - d^{\mathrm{GT}}_{\mathrm{stop}} \right| . \tag{22}$$

  The predicted stop duration is set to zero when no valid predicted stop is detected. We report the mean duration error in seconds over eligible examples.

- **MCQ Accuracy (higher is better).** MCQ accuracy is the accuracy of the model answering the question at the end of the clip, and is an indicator for the general scene understanding.

B.2 LingoQA and WaymoQA

LingoQA. LingoQA (Marcu et al., 2024) is a video question-answering benchmark for autonomous driving that covers both fine-grained scene perception and higher-level driving reasoning. Each sample contains a 4-second driving clip sampled at 1Hz (i.e. a total of 5 frames), together with free-form questions and answers concerning nine competencies, including action recognition, action justification, object to pay attention to, object identification, localization, description, counting, anticipation, and counterfactual reasoning. We test our memory on its evaluation set, which comprises 500 questions from 100 held-out scenarios, with two independently written reference answers per question, yielding 1,000 reference answers. Predictions are evaluated primarily using Lingo-Judge, a learned truthfulness classifier that determines whether a generated answer is semantically consistent with either reference answer; the resulting correctness scores are averaged over the evaluation set.

WaymoQA. WaymoQA (Yu et al., 2025b) is a visual question-answering benchmark built from long-tail scenarios in the Waymo End-to-End Driving Dataset (Xu et al., 2026) and designed to assess safety-critical driving reasoning. It includes both key-frame image questions and temporal video questions spanning perception, behavior prediction, planning, uncertainty, counterfactual reasoning, spatial and temporal relationships, traffic signals, and two-stage reasoning about immediate hazards and risks induced by an initial evasive action. In our experiment, we keep the video-VQA in the test set, which translates to 896 questions, and write memory for the video at 2.5Hz (same frequency as that in our dataset). The predictions are verified by rule-based scripts that extracts answers.

B.3 Hyperparameters

Tab. 4 lists the hyperparameter we use for our experiments. Notably, to align the training dynamics, for our no memory and CoT-as-memory baseline on the general driving dataset, we also train them on the same memory writing data as our method.

C Dataset Curation and Statistics

C.1 All-Way Stop Dataset

C.1.1 Dataset Curation

We curate our all-way stop dataset by first choosing the clips with all-way stop sign with the following standards: 1) ego has stopped in the clip, where "stop" is defined as speed ≤ 0.3 m/s for ≥ 0.2 seconds; 2) besides ego car, there must be at least one more car that has stopped, and the period of stopping must be no more than 1 second apart from ego's period of stopping (such that memory is needed); 3) At most one car has stopped before the clip starts, such that the stopping order can be recovered from the video. With such video, we then identify the moments of "stop" and "go" event of each car in the scene with trajectory information for ego and bounding box detection for other cars. In total, we curate 7436 clips for SFT training, 2479 clips for RL training and 2479 clips for evaluation with no joint clip between the splits.

Memory curation. As we care about "stop" and "go" in the all-way stop scenario, our memory is closely related to the stop/go status of each car and generated in a rule-based manner. We define six sequential states for the car: "moving towards the crossing", "almost arrives at the crossing", "arrived at the crossing", "stopping", "start moving", "passing the crossing", and assign them to our key frames according to the time of stop/go events. Memory at each frame is a concatenation of state description for each car (e.g. front car, left car, right car and ego); see Sec. C.1.3 for examples.

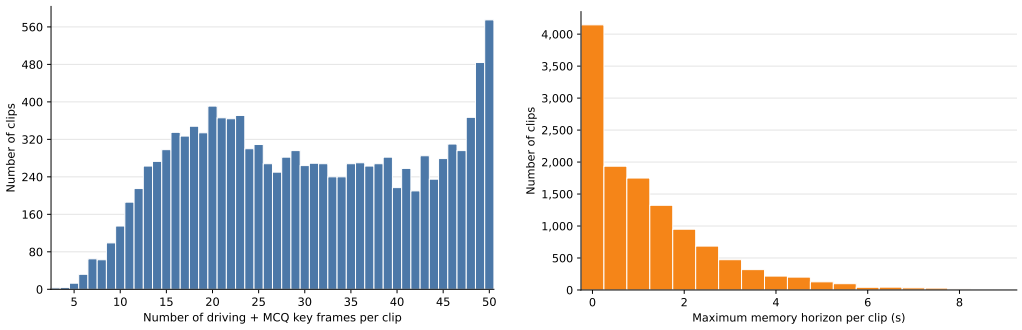
VQA curation. In this dataset, we focus on the following question: "What is the stopping order between the cars?". If the scene only involves two cars, we ask whether one of the car comes earlier; if three or more cars are involved, we ask that in which permutation did the car arrive. The reason for focusing on this question is because we find the other questions, such as going order, are severely biased (e.g. towards ego car moving last; see Sec. C.1.2 for details).

Table 4: List of hyperparameters for our algorithm.

Supervised Fine-Tuning	
global batch size	128
learning rates	1×10^{-5} (vision encoder); 1×10^{-4} (language model and LM head)
optimizer	fused AdamW
Adam betas / epsilon	$(0.9, 0.95)$ / 1×10^{-8}
weight decay	0.1
lr scheduler	cosine decay; 100-step linear warm-up; minimum lr factor 0.01
gradient clipping	1.0
precision	bfloat16 parameters; float32 reduction
maximum sequence length	8192 tokens
input frames per sample	4 (0.1s per frame)
weight for memory-writing in SFT c_1	1 for general driving, 0 for all-way stop
VQA weight in SFT c_2	10
Semi-Closed-Loop RL	
number of steps	420 (all-way stop, 2 epochs); 760 (general driving, 1 epoch)
global rollout batch	144 trajectories per training step (12 clips $\times$ 12 trajectories)
rollouts per clip / group size	12
optimizer	fused AdamW
learning rate	3×10^{-6}
lr scheduler	constant
Adam betas / epsilon	$(0.9, 0.999)$ / 1×10^{-6}
weight decay	0.01
sampling temperature / top-p	0.6 / 0.98
gradient accumulation interval	60 samples per GPU, with the final partial interval flushed
optimizer steps per epoch	vary with trajectory length; typically 3-4 (144 trajectories $\times$ 20-25 steps / 16 GPUs / 60 samples)
epochs of update per batch	1
gradient clipping	1.0
PPO clipping range	$\epsilon_{\text{low}} = 0.20$, $\epsilon_{\text{high}} = 0.28$
KL coefficient	0.001
importance sampling weight clipping	2.0
MCQ reward weight	0.5
memory reward	$\frac{1}{n}\sum_{t=i}^{n} r_t^{\text{drive}} + 0.5\, r^{\text{MCQ}}$ for step i
driving reward	$-0.2\,\text{ADE}$ for $\text{ADE} < 5\,\text{m}$; -1 otherwise
maximum response / context length	512 / 4096 tokens
training precision	bfloat16 parameters; float32 reduction
Evaluation	
sampling temperature	0.6
sampling top-p	0.98
parallel rollouts K	6
trajectory horizon	6.4 s
trajectory discretization	64 future waypoints at 10 Hz
memory horizon T	50

C.1.2 STATISTICS

Our all-way stop dataset contains 12394 clips, randomly divided into SFT training, RL training, and test sets in proportions of 60%, 20%, and 20%, respectively, as described in the main paper. Of these, 9877 clips concern the arrival order of two cars, while 2517 concern the arrival order of three or more cars. As co-waiting and situations in which vehicles' stopping periods overlap or are separated by at most 1 second are important but relatively rare in driving, we curated this dataset by filtering over one million clips. Each memory entry contains 14.827 tokens on average.



(a) Number of keyframes (b) Co-Wait Time

Figure 7: An illustration of the two key statistics of our dataset. Panel (a) is the number of keyframes, which is the length of our clip; panel (b) is the co-wait time between ego and other car, which is a surrogate for the memory horizon required.



Figure 8: An example of the visual input for all-way stop dataset at the VQA frame.

Fig. 7 shows the distribution of the number of keyframes per clip in panel (a) and the maximum co-waiting duration between the ego vehicle and any other vehicle in each clip in panel (b). Panel (a) reflects the distribution of clip lengths, as consecutive keyframes are spaced 0.4 seconds apart. Panel (b) serves as a proxy for the required memory horizon: when two vehicles are waiting simultaneously, the current frame alone may not reveal which arrived first. Notably, only 2.72% of clips have a co-waiting duration of at least 5 seconds, and none exceeds 10 seconds. **Using this proxy, a memory horizon of 5 seconds covers the estimated memory requirement for over 97% of clips, while 10 seconds covers all clips.** These observations guide our choice of memory horizon.

Bias in departure order. We examine the distributions of stop and go event orderings between the ego vehicle and other vehicles in our dataset. Across all pairs of stop events involving the ego vehicle and another vehicle, the ego vehicle stops earlier in 31.7% of cases, later in 42.6%, and simultaneously in 25.6%. For pairs of go events, the ego vehicle starts moving earlier in 18.8% of cases, later in 66.5%, and simultaneously in 14.8%. Given this strong bias toward the ego vehicle starting later, we exclude questions about departure order from our VQA task.

C.1.3 Examples

Below we show an example of question, corresponding memory and answer in an all-way stop dataset, with input visualized in Fig. 8. The rationale of the answer are generated by GPT-5.6 Luna by taking the memory, the question and the answer option as input.

Example Question:

Determine the chronological arrival order of the following vehicles at the crossing. Here, "$>$" means that the event on the left occurred more than 0.5 seconds earlier, while "$=$" means that the two events occurred within 0.5 seconds of each other; "$\geq$" means "no later than" and allows either relation. If multiple answers apply, choose any correct one.
1. car ahead arrives at the crossing 2. ego arrives at the crossing 3. left car arrives at the crossing

A. $3 \geq 2 \geq 1$ **B.** $2 \geq 1 \geq 3$ **C.** $2 \geq 3 \geq 1$
D. $1 \geq 3 \geq 2$ **E.** $1 \geq 2 \geq 3$ **F.** $3 \geq 1 \geq 2$

Example Memory:

`[-0.4s]` The car ahead seems to start moving. The left car is passing the crossing. The ego vehicle seems to start moving.
`[-0.8s]` The car ahead arrives at the crossing. The left car is passing the crossing. The ego vehicle is stopping.
...
`[-4.8s]` The car ahead is moving towards the crossing. The left car stays still. The ego vehicle arrives at the crossing.
`[-5.2s]` The car ahead is moving towards the crossing. The left car arrives at the crossing. The ego vehicle almost arrives at the crossing.
...
`[-6.8s]` The car ahead, the left car, and the ego vehicle are moving towards the crossing.

Example CoT as memory:

`[-0.4s]` the crosswalk due to the pedestrian crossing ahead.
`[-0.8s]` the intersection because the crossing vehicle has cleared.
...
`[-4.8s]` to yield to the vehicle crossing the intersection.
`[-5.6s]` to stop at the stop sign controlling the intersection.
...
`[-6.8s]` to stop due to the stop sign at the intersection.

Example Answer:

Ours. According to the history, the car ahead arrives at the crossing (1) 0.8 seconds ago, the ego vehicle arrives at the crossing (2) 4.8 seconds ago, and the left car arrives at the crossing (3) 5.2 seconds ago. Since the last two events are within 0.5 seconds of each other, the order is $3 = 2 > 1$. Thus, the answer is **A**.

CoT as memory. The left car is already passing through and then clearing the intersection, so it arrived first. The ego vehicle remains behind the stop line during the final visual inputs, while the car ahead does not enter the crossing before the ego vehicle. Thus, the answer is **A**.

C.2 General Driving Dataset

C.2.1 Dataset Curation

We curate a dataset that tests the performance of our algorithm in more diverse and challenging driving scenes. To identify such scenes, we use Qwen3-VL-30B-A3B-Instruct (Bai et al., 2025) and collect a set of 45446 clips (27246 SFT training set, 9088 RL training set and 9112 test set) with challenging scenes; see Sec. C.3 for the prompt.

Decision graph. With such clips available, the key challenge of language-based memory is to balance between brevity and inclusion of the objects that indeed affect driving; describing each of the 20 pedestrians on the sidewalk who never enters the road will only make the language memory less efficient than image input. Thus, we design a novel agentic framework in which the agent first identify and track items of interests in the frame with tools according to the video context. The agent

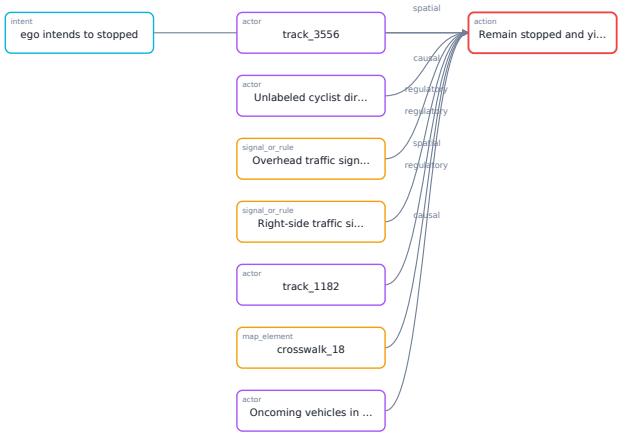


Figure 9: An example of a decision graph.

then builds a *decision graph*, which describes the objects of interest on the road and their relation with the ground truth action of ego car on this frame. By doing so, we ensure that all items described in the memory are relevant to the driving decision. Such graph is built with the following procedure:

- *Object Grounding:* Given the video clips, we first apply specialized object detection and map-element perception models to localize driving-relevant entities, including vehicles, Vulnerable Road Users (VRUs), crosswalks, lane areas, road signs, and traffic lights. Each detected entity is represented by a grounded visual prompt, such as an indexed bounding box, mask, or polygon overlaid on the corresponding frame. These prompts establish an explicit correspondence between image regions and unique object or map-element identifiers.
- *Semantic Association:* With the grounded object, a vision-language model is then queried with the visually prompted frames to associate higher-level semantics with the grounded entities, such as their type, state, motion, and relation to the ego vehicle or road layout. The resulting grounded entities and semantic attributes are converted into structured evidence records. Such evidences form a comprehensive set of candidate nodes in the decision graph, which will be filtered in the next step.
- *Evidence Gathering:* Given a specific key frame from the video and the related evidences, we call GPT-5.6 Luna to select the ones that affects driving with rationale, which is essentially a filter over the evidences. The model is also asked to review the frame and add any missing evidences in the image.
- *Retrieve Final Action:* We call GPT-5.6 Luna to generate future action of ego car in natural language, and calibrate with the ground truth future trajectory.
- *Build Graph:* With both the evidences and final action available, we prompt GPT-5.6 Luna to build a graph, where each evidence and final action is a node and the connection between them is an edge. The model is asked to give rationale and a category (e.g. causal, temporal, intentional, regulatory, etc.) for each edge connected.
- *Description-Based Linting:* With the decision graph generated, we use a rule-based script to conduct a sanity check on the graph, and regenerate those which have invalid node, empty rationale, or no connection to the final action.

Fig. 9 shows an example of decision graph. Notably, as the decision graph requires multiple VLM calls and can be expensive in practice, we only generate such graph for every 1.2s (i.e. every three keyframes), and then use GPT-5.6 Luna to fill in memory for the time between decision graphs.

Memory curation. With decision graphs for each key frame, we prompt GPT-5.6 Luna to curate the memory by first describing the nodes in the graph connected to the final action, then asking the model to review and merge memories from the same clip and "connect" them by keeping track of the same object throughout the text. We found that merging memory is vital, as repeated description

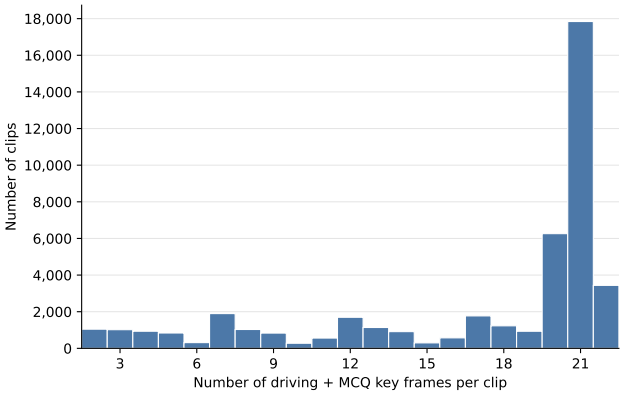


Figure 10: The distributions of the number of keyframes in the general driving dataset. Most clips are around 21 frames, which translates to 8 seconds of video.



Figure 11: An example of the visual input for the general driving dataset at the VQA frame.

of the same item is not only lengthy, but can also be misleading; for example, the stop caused by a pedestrian on the road, if described repeatedly, can mislead the model on the judgment of the traffic light. See Sec. 4.3 for an example. Notably, the memories are generated independently from VQA; this is because if the memory overfits toward a particular VQA question, it will only focus on a small set of object of interest and miss most items in the scene that affects driving.

VQA curation. We curate two types of VQA questions with GPT-5.6 Luna: the easy type and the hard type, where the former directly asks about an object of interest in the video, while the latter is multi-hop, i.e., asks the relative change between two time frames by checking the decision graph difference between the two frames. We use the former in the training set and the latter in the test set. This is because we found that training and testing on the same type of VQA questions will overfit and only illustrates the memorization of the dataset instead of genuine scene understanding; see Sec. C.2.3 for examples of the dataset, Sec. C.3 for prompts on curating the questions, and Sec. D.4 / Sec. D.6 for ablations.

C.2.2 Statistics

Our general driving dataset has a total of 27246 SFT training clips, 9088 RL training clips and 9112 test clips; Fig. 10 shows the distributions of the number of keyframes. On average, the length of each entry of memory is 29.054 tokens.

C.2.3 Examples

Below we show an example of questions and corresponding memories in our general driving dataset, with the image input illustrated in Fig. 11:

Example Easy Question:

What does the ego vehicle do throughout the sequence as it approaches and traverses the intersection?
A. It stops before the intersection, waiting for cross traffic to clear.
B. It turns left across opposing traffic after the signal changes red.
C. It pulls toward the curb and parks beside the tour storefront.
D. It continues straight across the intersection without changing its lane position.
E. It reverses briefly, then waits behind the stopped red car ahead.
F. It proceeds through the green-lit intersection, following the roadway's rightward curve.

Example Hard Question #1:

What sequence links the scene at 5.6 seconds ago with the later situation at 2.0 seconds ago?
A. Left-side traffic and a right-side parked vehicle constrained passage; later, a red vehicle followed ahead right-to-left.
B. Left-side traffic and a right-side parked vehicle constrained passage; later, a red vehicle crossed ahead left-to-right.
C. Left-side traffic and a right-side parked vehicle allowed passage; later, a red vehicle crossed ahead right-to-left.
D. Right-side traffic and a left-side parked vehicle constrained passage; later, a red vehicle crossed ahead right-to-left.
E. Left-side traffic and a right-side parked vehicle constrained passage; later, a white vehicle crossed ahead right-to-left.
F. Left-side traffic and a right-side parked vehicle constrained passage; later, a red vehicle crossed ahead right-to-left.

Example Hard Question #2:

Which sequence best describes the ego vehicle's response as it approached and then traversed the intersection?
A. It began stopped behind traffic, then merged right while continuing through the intersection.
B. It began stopped near the crosswalk, then turned left after nearby pedestrians cleared its path.
C. It began stopped amid tight lateral clearances, then proceeded straight while preserving crosswalk clearance.
D. It began moving beside parked vehicles, then stopped only after clearing the crosswalk.
E. It began moving through the intersection, then continued straight after stopping beside roadside vehicles.
F. It began moving behind traffic, then remained stopped as oncoming vehicles cleared the intersection.

Example CoT as Memory:

`[-0.4s]` the right lane change due to slower traffic ahead in the current lane.
. . .
`[-2.0s]` the right turn into the rightmost lane because the signal permits the turn, the lead car ahead is already turning, and the crosswalk and bike lane to the right are clear of conflicts.
. . .
`[-4.8s]` to merge right behind the lead car because the intersection and crosswalk ahead require a rightward merge.
`[-5.6s]` to turn right at the intersection to follow the lead vehicle.
`[-6.4s]` to stop due to the red traffic light.
. . .
`[-8.4s]` to merge left due to the lead vehicle moving left ahead.

Example Memory:

[-0.4s] The ego vehicle continues straight past the intersection crosswalk behind the red vehicle, with additional traffic ahead and a marked bicycle lane on the right.

...

[-2.0s] The ego vehicle continues through the intersection while the red crossing vehicle moves farther left and vulnerable road users remain near the crosswalk.

...

[-5.6s] The ego vehicle remains stopped as the oncoming vehicle continues past on the left and vulnerable road users remain near the crosswalk and right curb.
[-6.0s] The ego vehicle remains stopped with the vehicle ahead waiting, a large black vehicle at the right edge, an oncoming vehicle alongside on the left, and vulnerable road users near both sides of the crosswalk.

...

[-8.4s] The ego vehicle is stopped at a nighttime urban intersection approach with an oncoming vehicle on the left, a vehicle directly ahead, a vehicle close along the right edge, crosswalks ahead and left, and pedestrians near both crossings.

Example Answer:

Ours.
Easy: F. The ego vehicle proceeds through the green-lit intersection, following the roadway's rightward curve.
Hard #1: F. Traffic occupied the left side while a parked vehicle narrowed the right side, constraining straight passage. A red vehicle later crossed ahead from right to left.
Hard #2: C. The ego vehicle was initially stopped while nearby traffic and close roadside vehicles constrained the approach. It then began moving straight through the intersection while preserving crosswalk clearance.

CoT as memory.
Easy: F. The ego vehicle proceeds through the green-lit intersection, following the roadway's rightward curve.
Hard #1: F. The constrained passage was followed by the red vehicle crossing ahead from right to left.
Hard #2: C. The ego vehicle changes from being stopped amid tight lateral clearances to proceeding through the intersection.

C.3 Prompts for Data Generation

Below we show the core prompt for data generation. We first show the prompt for filtering the challenging clips for the general driving dataset. We only disclose excerpts of the prompt for both brevity and protection of proprietary information:

Find challenging scenario that requires deep human reasoning to drive through safely. Key criteria include:
- Complex interactions between multiple road users (e.g., pedestrians, cyclists, vehicles) that require careful negotiation and anticipation.
- Unusual or unexpected events (e.g., sudden pedestrian crossing, erratic behavior from other drivers) that demand quick and accurate decision-making.

and exclude:
- only concern is low visibility or adverse weather, without complex road user interactions or unexpected events.
...

Analyze the video and motion to deduce ego's driving decision at the keyframe according to the pre-defined decision list below. Here's a list of pre-defined LONGITUDINAL and LATERAL driving decisions:
- LONGITUDINAL decisions
– Stop
Decelerate to, hold at stop/yield lines or other control points (traffic light, stop sign, school bus/railroad rules, blocked path by lead vehicle).
– Yield
Yield to the [pedestrian/cross-traffic/cyclist/emergency vehicle] / give way for / wait for priority traffic to clear.
...
Priority policy (must follow):
1. First check if ego is conducting 'interesting' decisions:
- LONGITUDINAL: 'Yield', 'Stop', 'Pass/Overtake', 'Gap-search', 'Adapt Speed'.
- LATERAL: 'Lane Change', 'Nudge', 'Turn', 'Merge', 'Split'.
- If any interesting decision is clearly present, output that decision and do not downgrade to generic 'Keep Distance', 'Keep Lane' or 'Resume Speed'.
2. Lane change and nudge are top-priority lateral decisions:
...
Critical definitions:
1. Lead vehicle means a vehicle ahead in the same lane and moving in the same direction.
...
Best practices:
1. Use "crosswalk" and only mention it when a pedestrian is walking across the road.
...
Format:
...

Next, we show the prompt for VQA curation of the general driving dataset:

 # Easy questions
You are an expert autonomous-driving scene analyst.
You are given front-wide-camera frames sampled at 2 frames per second from exactly the interval beginning at this clip's first retained driving frame and ending at its final retained driving frame. Frames are ordered oldest to newest. A prior caption is supplied only to help locate potentially relevant actors; it may be wrong, so determine behavior from the frames.
Prior caption: {prior_description}
Create one six-option multiple-choice question about a temporal behavior or ego-object interaction that requires the full interval, not only the final frame. If no other actor is salient, ask about ego motion or traffic-control response. Audio is unavailable.
Requirements:
- Exactly six mutually exclusive options A-F.
- correct_answer must be A before downstream deterministic shuffling.
- Options must be plausible and scene-compatible, with matched grammar, specificity, certainty, and action intensity.
- Each option must have 8-12 words; longest-shortest difference at most 3 words.
- Do not use memory notes or information outside the supplied frames.
- Return only the requested JSON object.

 # Hard question 1
You are an expert autonomous-driving scene analyst and a rigorous temporal multiple-choice-question author. You receive two synchronized sources for one driving clip:
1. Front-wide-camera images ordered oldest to newest. Immediately before every image is a TEXT label of the form "Frame time: N.N seconds ago" or "Frame time: now". The label is not drawn on the image. Images are sampled at approximately 2 frames per second, so adjacent labels are usually about 0.5 seconds apart. "Now" is the final question image.
2. Raw structured decision-graph snapshots sampled on a 1.2-second source grid. Every snapshot is explicitly marked with its time relative to "now" and contains compact 'evidence', 'nodes', and 'edges'. Node fields include id, type, label, and confidence. Edge fields include source, target, relation, rationale, and confidence. Use this structure to follow entities and decision relations over time, but verify every question-critical claim against the images.
Important graph interpretation rules:
- An 'action' node is a hypothetical safe decision, not necessarily observed ego behavior. Never present it as something ego actually did unless images show it.
- Internal node/track IDs are for linking snapshots only and must never appear in a question or option.
- A snapshot marked slightly "after now" exists only because source and camera timebases can differ within the stated alignment tolerance. Never use a fact visible only after now.
- Missing snapshots must not be interpolated or invented.
Raw 1.2-second decision-graph context:{decision_graph_context}
Produce exactly ONE 'past_anchor' six-option MCQ. Analyze the complete clip before writing:
1. Track every salient actor, traffic control, road feature, hazard, and the ego vehicle across time. Resolve which references denote the same entity.
2. Identify state changes, temporal order, interactions, and causal or regulatory links. Images are authoritative when they conflict with graph snapshots.
3. Choose one exact supplied historical time and write it naturally as "N.N seconds ago". Never use notation such as "t=-2.4s".
4. The answer must require both the anchored historical evidence and evidence from a second supplied time. Do not ask for a static fact at the anchor alone.
5. Apply the final-image-only counterfactual: hide every historical image and all graph snapshots. If a careful observer could answer from "now" alone, reject and rewrite the question.
Time and spatial ambiguity rules:
- Do not ask "when did X happen?" when X persists across multiple images. A state covering an interval cannot be assigned a unique point timestamp.
- Prefer questions about a change after the anchor, an interaction spanning the anchor and a later time, or the complete anchored state needed to explain a later response.
- Exactly one option must be completely correct under every reasonable reading.
- Do not mix overlapping spatial axes. If an actor is ahead-left, "ahead" and "left" are both partially true. Ask about one explicitly named axis: lateral = left/center/right or longitudinal = ahead/alongside/behind. Otherwise use matched composite relations such as ahead-left/ahead-center/ahead-right in every option.
- Avoid nested categories such as moving versus accelerating, stopped versus yielding, vehicle versus truck, or crossing versus ahead-left.
- Distinguish actor identity whenever two similar actors could match.
- State in 'ambiguity_check' why A is uniquely complete, why the strongest distractor is false, and why the anchor does not represent a persistent-state timing ambiguity.
Distractor quality and wording:
- Distractors must be realistic near-misses for this exact scene. Prefer changing one subtle attribute: temporal order, matched spatial relation, speed trend, signal state, actor identity, or ego response.
- Never use obviously absurd, dangerous, or extreme actions merely to fill an option. In particular, no option may use reverse/reversed/reversing, swerve/swerved/swerving, crash/collide/collision, U-turn, driving onto a sidewalk or curb, entering oncoming traffic, spinning, or rolling over.
- Do not invent an actor type or maneuver unsupported by the clip.
- Avoid giveaway modifiers such as "suddenly", "recklessly", "violently", "immediately", "without reason", "at full speed", or "completely ignored".
- Match grammar, semantic axis, specificity, certainty, and action intensity across all six options.
Output requirements:
- Exactly six mutually exclusive options A-F.
- 'correct_answer' must be A before downstream deterministic shuffling.
- Each option must contain 8-16 words; longest-shortest difference at most 5 words.
- Do not ask about a static fact, count, color, or location visible at "now".
- Do not ask about audio, hidden intent, or information outside supplied inputs.
- Do not expose internal graph IDs or refer to graphs, memory, notes, or frames in the question or options.
- Return only the requested JSON object.

 # Hard question 2
... (same as hard question 1) ...
Produce exactly ONE 'multi_hop' six-option MCQ.
Analyze the complete clip before writing:
1. Track every salient actor, traffic control, road feature, hazard, and the ego vehicle across time. Resolve which references denote the same entity.
2. Identify state changes, temporal order, interactions, and causal or regulatory links. Images are authoritative when they conflict with graph snapshots.
3. Build an evidence chain from at least two distinct supplied times. At least one indispensable fact must be historical.
4. The answer must require combining at least two different decision-relevant facts, such as an actor state change plus ego response, a traffic-control transition plus actor interaction, or temporal order plus the resulting maneuver. Comparing two locations or restating one change is not multi-hop.
5. Apply the final-image-only counterfactual: hide every historical image and all graph snapshots. If a careful observer could answer from "now" alone, reject and rewrite the question.
6. Do not put numerical timestamps in the question or options.
Mandatory ambiguity audit:
... (same as hard question 1) ...

Then, we show the prompt for memory merging of the general driving dataset below:

Fill and rewrite the complete 0.4-second memory sequence for this driving clip. Every labeled image is one required output timestamp. Some images include an independently generated anchor memory; the other images explicitly say that no independent memory exists. Return exactly one memory for every image timestamp, in the same order.
Rules:
1. Use every image, the anchor memories, and temporal continuity together. Preserve all factual anchor information, but do not invent details unsupported by either an image or an anchor.
2. Locate changes at the earliest 0.4-second image where they are visibly supported. For example, if a signal changes between two 1.2-second anchors, inspect all intermediate images and mention the new color only at the frame where it first appears. Apply the same frame-accurate treatment to ego acceleration/deceleration, stopping, starting, turning, lane changes, actor motion, and visibility changes.
3. The first output establishes the scene. Later outputs are concise temporal deltas. Omit unchanged static background such as an already established signal color, sign, crosswalk, lane, snowbank, or road surface until it changes. Every frame must still contain useful supervision: use concise continuity wording for reliably visible dynamic subjects when no new event occurs.
4. Track the same physical subject across frames. Introduce it once with enough visible appearance, location, and behavior to identify it uniquely, then use a stable natural reference while unambiguous.
5. When reliable observed ego behavior is visible, put it first and begin with "The ego vehicle". Describe ego motion only qualitatively, such as stopped, starting, accelerating, decelerating, moving straight, turning, or changing lanes. NEVER output a numeric ego speed or units such as m/s, mph, or km/h in any output frame. If ego behavior is uncertain, omit the ego vehicle completely. Never write "The ego motion remains unspecified", "The ego vehicle's observed motion is unspecified", "No ego maneuver is evidenced", or any equivalent placeholder.
6. An anchor memory equal to "N/A" is a failed generation, not a fact. Recover that frame from its image and temporal context; never copy "N/A" into the output.
7. Never emit track IDs, lane ordinals, node/evidence IDs, or raw timestamps inside memory_text. Do not report hypothetical safe actions or future plans as actual behavior.
8. Return one concise present-tense sentence per timestamp. Do not leave any output empty.
The response must match the requested JSON shape. Copy each input timestamp exactly into the corresponding output object; put only the completed sentence in memory_text.

Finally, we report the prompt for generating answers for VQA:



Figure 12: An example where memory benefits general driving; a pedestrian in the red box is occluded behind the crosswalk sign, which is hard to discover; the intention of the pedestrian is even harder to tell.



Figure 13: The previous frames of the same clip as that in Fig. 12. With these frames, it is clearly shown that the pedestrian is walking towards the road and the ego car should yield.

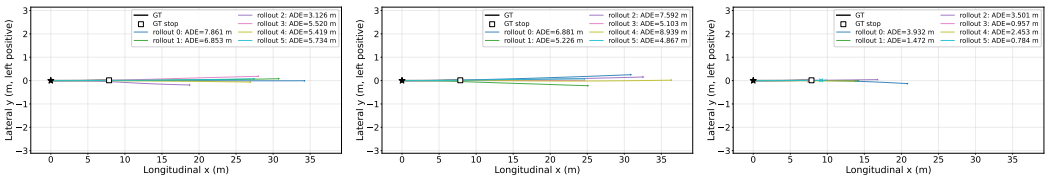
You are verifying a fixed-ground-truth multiple-choice question about an autonomous-driving scene.
You receive: 1. one current ego front-wide-camera image; 2. timestamped scene memory from earlier frames; 3. a fixed question with six fixed options; and 4. the fixed correct answer label and text.
Write one concise rationale that explains why the fixed correct answer follows from the current image and the supplied earlier scene memory.
Requirements: - Do not change or challenge the supplied answer. - Use only evidence visible in the current image or stated in the supplied memory. - Describe the relevant temporal change or ego-object interaction directly. - Do not mention "image", "frame", "memory", "notes", "prompt", or the answer label. - Do not introduce precise numbers, identities, directions, or events that are absent from the supplied evidence. - Use one or two sentences and no answer tag. - If the supplied evidence genuinely contradicts the fixed answer or cannot support any concise rationale, set supported to false instead of inventing evidence. - Return only the requested JSON object.
Earlier scene memory: {memory_block}
Fixed question: {question}
Fixed options: {options}
Fixed correct answer: {correct_label}. {correct_text}

D Additional Experiment Results

D.1 More Qualitative Results

In this section, we illustrate another qualitative example in our general driving dataset (Sec. 4.2) where language memory helps. Fig. 12 shows that a pedestrian is occluded by the crosswalk sign, which is hard to discover in current input; however, once the agent has access to memory as shown in Fig. 13, it is much easier to see that the pedestrian has an intention of walking onto the road. Below shows an example of memory of our agent:

[-0.4 s] The ego vehicle continues straight toward the crosswalk behind the vehicle ahead while pedestrians remain on the right and center-left portions of the crossing.
[-0.8 s] The ego vehicle continues straight behind the vehicle ahead; the pedestrian is farther into the right side of the crosswalk, with another pedestrian near the center-left refuge area.
[-1.2 s] The ego vehicle continues straight behind the vehicle ahead as the **pedestrian moves leftward across the right side of the crosswalk** and the construction barriers remain along the right edge.
[-1.6 s] The ego vehicle is moving straight toward a marked crosswalk, following a vehicle directly ahead, with another vehicle ahead-left, **a pedestrian crossing from the right**, and red-and-white construction barriers narrowing the right edge.
...



(a) No memory (b) CoT-as-memory (c) Ours

Figure 14: The trajectories predicted by different methods in the example of Fig. 12 and Fig. 13. It is clearly shown that our method outperforms baselines.

Table 5: The ablation on *Da Capo*, which proves that a better credit assignment significantly improves final performance over trajectory-level GRPO. The standard deviation in advantage also helps by introducing a natural scaling factor that balances rewards of different magnitudes.

	minADE ↓	Avg. ADE ↓	ML. ADE ↓	Stop SR ↑	Go SR ↑	Δ_{pos} ↓	Δ_{dur} ↓	Roll ↓	MCQ Acc. ↑
Da Capo (Ours)	0.951	**1.866**	**1.915**	**89.26%**	**45.01%**	**1.070**	**1.183**	5.26%	**51.66%**
Capo (no std)	**0.942**	1.952	2.047	89.22%	44.27%	1.130	1.277	**5.01%**	50.91%
Capo + const. scaling	0.943	1.911	1.979	89.21%	44.74%	1.122	1.227	5.03%	51.57%
GRPO (traj-level reward)	1.007	2.113	2.194	87.11%	40.94%	1.312	1.392	6.28%	48.16%
AD-Memo (SFT only)	1.049	2.121	2.256	87.48%	40.66%	1.257	1.469	6.05%	45.99%

The memory clearly records the existence and intention of pedestrian. The predicted trajectories are illustrated in Fig. 14, where our method outperforms no memory and CoT-as-memory.

D.2 Deviation Adjustment (DA) for *Da Capo*

To test the necessity of introducing standard deviation in our RL algorithm (see Appendix A for detailed rationale), we test our RL algorithm based on the same AD-Memo (SFT-only) checkpoint against the following baselines: *Capo* (no standard deviation), using a constant scaling factor on advantage ($= 0.4$), and trajectory-level GRPO. The results summarized in Tab. 5 show that adding standard deviation to our algorithm improves the empirical performance. The result also shows that all variants of *Capo* far outperform GRPO with trajectory-level reward, which proves the benefit of removing causally unrelated reward terms.

D.3 Scaling Up Model Size

In order to see how our model works with models of larger scale, we further conduct SFT on another 8B checkpoint of Alpamayo 2 on our all-way stop dataset (Sec. C.1.1). Tab. 6 shows the result, where 8B models generally outperform 2B models; our finetuned model performs the best among 8B models, and also far outperforms Alpamayo 2 Super 34B.

D.4 Overfitting Check on VQA of the General Driving Dataset

The biggest challenge of curating VQA questions on the general driving dataset is that models will easily overfit to questions from the same distribution by memorizing patterns in the dataset without understanding, sometimes even without assigning any attention to the visual input. Many prior autonomous driving benchmarks are haunted by such issue. For example, models on TUMTraffic-VideoQA (Zhou et al., 2025a) can achieve 71% accuracy with no visual input, while only achieving 81% accuracy with 11 frames; NuScenes-QA (Qian et al., 2024) reports an accuracy of 53.4% by only looking at the question alone; on DriveBench (Xie et al., 2025), models with no image input can even achieve 95% GPT-score performance compared to normal input.

Therefore, to check whether our model overfits on the curated questions after training, and also as a sanity check to the correctness of our curated questions, we test the performance of GPT-5.6 Luna on our "easy" and "hard" split of MCQ questions, as well as the performance of our checkpoint when trained on the respective split of these questions. The result is illustrated in Tab. 7, which gives us the following two insights:

Table 6: Results of 8B models on All-Way-Stop v4. The result clearly shows that our 8B model works the best.

	minADE ↓	Avg. ADE ↓	ML. ADE ↓	Stop SR ↑	Go SR ↑	Δ_{pos} ↓	Δ_{dur} ↓	Roll ↓	MCQ Acc. ↑
Alpamayo 2 Super 34B	0.993	2.293	N/A	78.75%	34.30%	2.182	1.488	15.94%	33.19%
Base model (2B)	1.386	2.351	2.393	82.06%	33.74%	1.725	1.871	11.79%	0%
No memory (2B SFT)	1.102	2.185	2.318	87.04%	38.59%	1.358	1.564	6.62%	33.77%
CoT-as-memory (2B SFT)	1.096	2.171	2.296	86.89%	39.37%	1.377	1.531	6.69%	45.41%
Ours (2B SFT)	**1.049**	**2.121**	**2.256**	**87.48%**	**40.66%**	**1.257**	**1.469**	**6.05%**	**45.99%**
Base model (8B)	1.315	2.254	2.332	83.72%	35.65%	1.627	1.776	10.90%	0%
No memory (8B SFT)	1.011	2.092	2.255	88.11%	42.05%	1.252	1.415	6.04%	39.31%
CoT-as-memory (8B SFT)	1.000	2.025	2.130	88.18%	42.48%	1.230	1.374	5.92%	44.92%
Ours (8B SFT)	**0.980**	**1.986**	**2.071**	**88.99%**	**43.24%**	**1.130**	**1.352**	**5.40%**	**52.46%**

Table 7: The MCQ accuracy on our "easy" VQA questions (i.e. the one in the training set) and "hard" VQA questions (i.e. the one in the test set) for the general driving dataset. GPT-5.6 Luna with full video serves as a sanity check and verifier for the correctness of our generated VQA questions. Avg. test set is the average of hard #1 and hard #2; the example for each type is listed in Sec. C.2. It is clearly shown that test on the same distribution of VQA questions as training causes overfitting. Thus, we deliberately use different questions for the training and test set.

	Easy	Hard (#1, past-anchor)	Hard (#2, multi-hop)	Avg. test set
Random	16.67%	16.67%	16.67%	16.67%
Alpamayo 2 Super 34B	48.45%	25.77%	25.65%	25.71%
No memory	**98.14%**	54.93%	61.10%	58.02%
CoT-as-Memory	98.02%	51.46%	63.56%	57.51%
Ours	97.92%	**60.92%**	**70.34%**	**65.63%**
CoT-as-Memory (GT mem)	97.99%	51.25%	63.48%	57.36%
Ours (GT mem)	98.22%	63.00%	72.29%	67.65%
GPT-5.6 Luna + full video	94.34%	89.14%	91.57%	90.35%

- VQAs training on the same distribution will very likely overfit on the test set, even surpassing GPT-5.6 Luna performance with full video. We also tried to use the "hard" split as the training set, but the accuracy on test data generated from the same distribution will still reach over 96%.
- Our designed memory indeed achieves the best result on the harder test set with different distributions, and can do even better with ground truth memory (as opposed to CoT-as-memory where ground truth memory does not help).

D.5 Which Scene Does AD-Memo Benefit the Most?

In this section, we will investigate our test set and to locate the scenes where AD-Memo benefits the most. Following prior work (Tan et al., 2026), we ask GPT-5.6 Luna to classify the clips in our test set into 15 categories, including turning maneuver, speed control, merging, lane change, cut-in, nudge maneuver, nudge static obstacle maneuver, Vulnerable Road Users (VRU), intersection navigation, lane keeping curve, lead vehicle following, traffic control compliance, lane keeping, stop for vehicle, and others. Fig. 15 shows the ratio of each type of data, and Tab. 8 shows the performance difference between our method and the baseline without memory. The result shows that our method improves more on the performance of average and the most likely rollout, and works better on the more common challenging scenes, especially traffic control compliance, lead vehicle following, and stop for vehicles.

D.6 The Necessity of Decision Graph

As we mentioned in Sec. C.2.1, the purpose of building decision graphs is to identify the objects on the road that affects driving. In this section, we illustrate the necessity of decision graph by providing an example in Fig. 16, where we compare the memory directly generated by simply feeding the

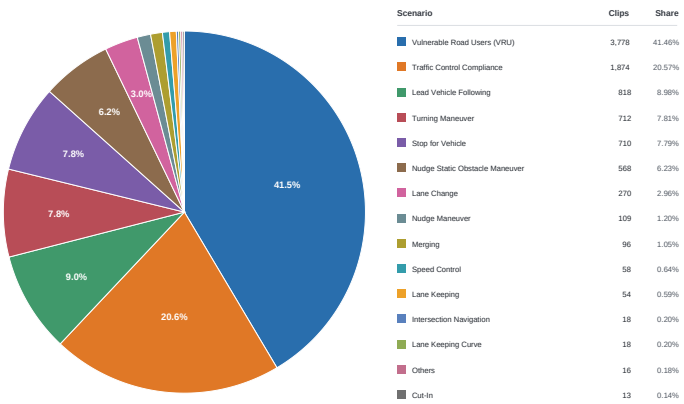


Figure 15: The distribution of different scenes in the test set of our general driving dataset.

Table 8: Per-scenario open-loop trajectory error gains of our model over the no-memory model (after RL); as it is the *gain*, the higher the better, i.e., positive values indicate improvement, and vice versa. The categories are sorted in descending order of their shares. All values are in meters. It is shown that AD-Memo outperforms the no-memory model on the more common scenes, especially traffic control compliance, lead vehicle following and stop for vehicles.

	ADE gain (min/Avg./ML)	FDE gain (min/Avg./ML)	Shares (%)
Vulnerable Road Users	-0.011/+0.010/+0.018	-0.011/+0.054/+0.072	41.46%
Traffic Control Compliance	+0.013/+0.023/+0.030	+0.089/+0.101/+0.105	20.57%
Lead Vehicle Following	+0.003/+0.017/+0.012	+0.021/+0.058/+0.031	8.98%
Turning Maneuver	-0.002/+0.024/+0.031	-0.019/+0.061/+0.092	7.81%
Stop for Vehicles	+0.008/+0.011/+0.017	+0.057/+0.072/+0.067	7.79%
Nudge Static Obstacle Maneuver	-0.006/+0.030/-0.001	-0.040/+0.063/-0.035	6.23%
Lane Changes	+0.020/+0.060/+0.060	+0.021/+0.143/+0.124	2.96%
Nudge Maneuver	+0.002/+0.042/+0.067	+0.051/+0.081/+0.154	1.20%
Merging	-0.010/+0.031/-0.017	-0.013/+0.092/-0.068	1.05%
Speed Control	-0.043/+0.033/+0.030	-0.089/+0.159/+0.136	0.64%
Lane Keeping	-0.036/-0.031/-0.012	-0.155/-0.132/-0.049	0.59%
Intersection Navigation	+0.022/+0.051/-0.128	-0.238/-0.035/-0.503	0.20%
Lane Keeping Curve	-0.089/-0.061/+0.024	-0.455/-0.258/-0.045	0.20%
Others	-0.270/-0.337/-0.332	-1.041/-1.220/-1.398	0.18%
Cut-in	-0.128/-0.068/+0.003	-0.396/-0.211/-0.190	0.14%

whole video into GPT-5.6 Luna and prompting the model to write memory. The generated memories are as follows:

> `Direct generation`: The pedestrian is at or very near the right curb/sidewalk and is nearly clear of the crosswalk; ego continues straight.
> `Decision graph`: The ego vehicle remains stopped while the pedestrian with the stroller is at the right side of the rainbow crosswalk, with a yield sign on the right and large vehicles, the pickup, and box truck obstructing the left approach.

The result clearly shows that without decision graph, the generated memory will lose track of many vital items in the image, such as the stroller, crosswalk, yield sign and obstruction on the left.

D.7 Decode Success Rate

As the output of our model contains CoT, memory and trajectory tokens, it is possible that the output of the model does not follow the corresponding format, thus leading to failures of decoding output. In Tab. 9, we report the decode rate of our model and baselines, which shows that our model maintains a high decode success rate. Notably, during evaluation, if we fail to decode memory from one step, we will set the corresponding slot of memory to null moving forward.



(a) Visual input (b) Object detection (c) Decision graph

Figure 16: An example of how our pipeline based on decision graph identify all objects related to driving. All objects that affect driving decision are grounded in panel (b) from the input frame shown in panel (a); these objects end up as nodes in the corresponding decision graph in panel (c).

Table 9: The decode success rate for our model and baselines, which are all reasonably close to 100%.

Model	Decode success rate
All-way stop dataset	
Base model	100.00%
Alpamayo 2 Super 34B	100.00%
No memory (SFT)	99.44%
CoT-as-Memory (SFT)	99.41%
Ours (SFT)	99.51%
No memory (RL)	99.85%
CoT-as-Memory (RL)	99.87%
Ours (RL)	99.86%
General driving dataset	
Base model	99.87%
Alpamayo 2 Super 34B	100.00%
No memory (SFT)	99.78%
CoT-as-Memory (SFT)	99.69%
Ours (SFT)	99.62%
No memory (RL)	99.91%
CoT-as-Memory (RL)	99.85%
Ours (RL)	99.83%

D.8 Semi-Closed Loop RL Training Curve

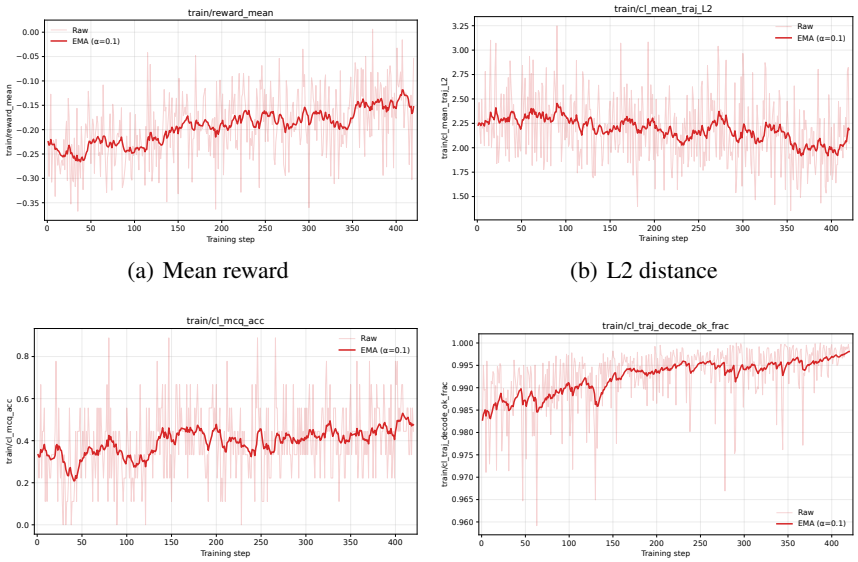
Fig. 17 illustrates the training dynamics of our semi-closed loop RL on the all-way stop dataset, while Fig. 18 illustrates the training dynamics of our semi-closed loop RL on the general driving dataset.

D.9 Ablation on λ^{VQA}

Tab. 10 shows the ablation result of using different λ^{VQA} on our test set. The result shows that our method is generally robust to the choice of λ, with the optimal λ^{VQA} for each metric spanning over the table. Notably, larger λ^{VQA} does slightly better than smaller λ^{VQA}. Such performance difference comes from the fact that the cumulative future driving reward is a highly noisy signal. Thus, even if smaller λ^{VQA} encourages the model to focus on driving, its performance will not necessarily improve. Such noisiness is also one of the motivation for *Da Capo* that gives better credit assignment and reduces variance.

D.10 Ablation on SFT Training Length

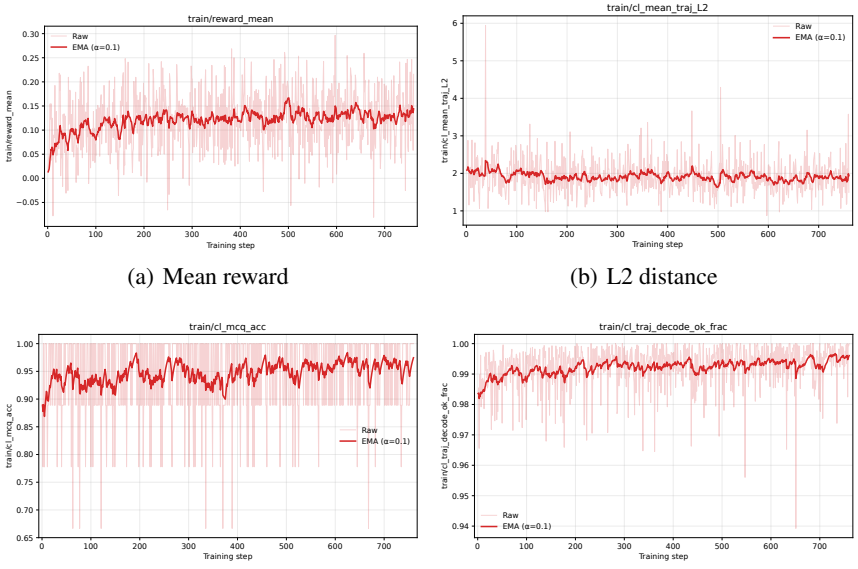
To test the optimal training length of our model on SFT, we train our 2B checkpoint for up to 4 epochs on an earlier split of our general driving SFT dataset, and test its performance on a separate



(a) Mean reward (b) L2 distance

(c) VQA accuracy (d) Decode rate

Figure 17: The training dynamics of our semi-closed loop RL on the all-way stop dataset. We report the original curve and Exponential Moving Average (EMA).



(a) Mean reward (b) L2 distance

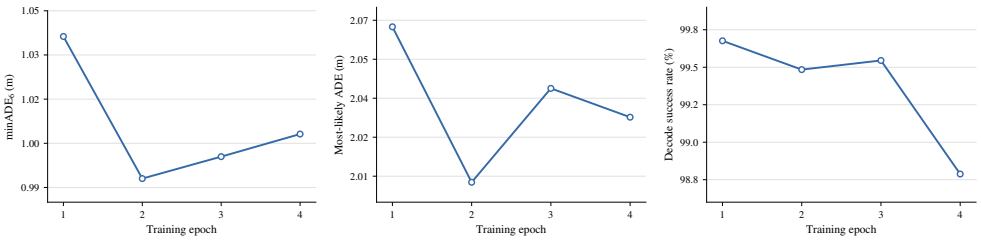
(c) VQA accuracy (d) Decode rate

Figure 18: The training dynamics of our semi-closed loop RL on the general driving dataset. We report the original curve and Exponential Moving Average (EMA). The VQA accuracy is close to 1 as the problems are relatively easy to overfit, and thus we test different problems in the test set; see Sec. D.4 for details.

2000-clip validation set on that split. The result is illustrated in Fig. 19, which shows that training for 2 epochs is optimal.

Table 10: The result of our method on the all-way stop dataset with different λ^{VQA}. The performance is generally consistent, and no hyperparameter choice prevail on all metrics.

λ^{VQA}	minADE ↓	Avg. ADE ↓	ML. ADE ↓	Stop SR ↑	Go SR ↑	Δ_{pos} ↓	Δ_{dur} ↓	Roll ↓	MCQ Acc. ↑
0.02	0.945	1.886	1.927	88.73%	44.80%	1.111	1.184	5.54%	50.43%
0.05	0.943	1.877	1.913	88.67%	45.01%	1.092	1.178	5.38%	50.65%
0.1	**0.932**	1.886	1.898	88.71%	45.01%	1.129	1.179	5.28%	51.61%
0.2	0.960	1.876	1.931	89.03%	44.94%	1.099	1.194	5.34%	51.12%
0.5	0.951	**1.866**	1.915	**89.26%**	45.01%	**1.070**	1.183	5.26%	**51.66%**
1	0.941	1.900	**1.891**	88.52%	45.04%	1.119	**1.174**	5.44%	50.38%
2.5	0.942	1.886	1.909	88.10%	**45.33%**	1.130	1.175	5.74%	50.80%
5	0.948	1.874	1.908	88.87%	45.07%	**1.070**	1.175	**5.25%**	50.47%



(a) minADE (b) Most likely ADE (c) Decode rate

Figure 19: The performance change with the training length of SFT on the validation set. It is clearly shown that, after 2 epochs, not only does the trajectory move further from ground truth, but the format following ability of the model is also degrading due to exposure bias (Bengio et al., 2015), i.e., more invalid rollouts appear in sampling.

E Computational Resources

Our computational resource consumption is as follows:

Data curation: Generating a decision graph takes on average 10.43 model calls, which sums to roughly 15K input tokens and 5K output tokens.

SFT experiments: All SFT experiments are conducted on a single compute node with 8 NVIDIA H100 GPUs (80GB memory), 96 AMD EPYC 9R14 CPUs, and 512GB memory. A 8900-step SFT training on the general driving dataset takes approximately 16 hours.

RL experiments: All RL experiments are conducted with four compute nodes (two for rollout and two for training), each of which is identical to a SFT node. RL experiments typically require 2-3 days to complete.

Evaluation: All evaluations are conducted with 16 single-GPU node with each node identical to the SFT node (except for the number of GPU). Evaluation usually takes 4-6 hours.

F Limitation and Future Works

Horizon of memories. Our work mainly studies memories that lasts around 10 seconds. While we empirically verified this to be sufficient for our task of interest (Sec. C.1), long-term regulations such as speed signs may exert influence on driving for minutes to hours. While AD-Memo can straightforwardly address such problem by adding a special buffer for "long-term memory" heuristically, an in-depth investigation into such area is still interesting.

Extending our finding to action expert. Our experiment is mainly done with a VLM backbone, which means the action expert in Alpamayo is not involved in this work. This is because in the Alpamayo training pipeline (NVIDIA, 2026), the action expert requires extensive training after the VLA backbone is finalized, which is too expensive to run. To see whether the advantage of language memory persists with action expert is an interesting avenue to explore in the near future.

Comparison with vector-based baselines. We choose the same training recipe without memory and with CoT-as-memory as our baseline because direct comparison to vector-based baselines is difficult. Most related papers have no training code or checkpoint released, such as MindVLA-U1 (Huang et al., 2026c), enhanced HOPE (Kim & Park, 2026), VLN-AVP (Li et al., 2026b) and AdaDrive (Zhang et al., 2025).

For those with code available, COMPACT-VA (Liang et al., 2026) considers the closest scene to us, and we adopt some of its metrics. However, the memory design is based on a different VLA architecture with the assumption of 20 or more frames, which is not supported by our Alpamayo base model (pretrained to only receive 4 frames). Also, it requires a fixed number of frame input with no padding (e.g. 20), for which many clips in our test set cannot be tested for their limited length (and it is also unfair as we report the average of many frames in a clip, including early ones in the video that must be padded for COMPACT-VA).

For the rest of the baselines, MoMAD (Song et al., 2025) uses ResNet (He et al., 2016) instead of VLA as backbone; to adapt such a method to VLA is beyond our scope (and we already have a better model, Alpamayo 2 Super 34B, as the off-the-shelf baseline compared to MoMAD's original checkpoint). ST-Occ (Leng et al., 2025) is not a driving model and deals with voxels; Orion (Fu et al., 2025) is based on CARLA dataset, while our work uses real-life clips with a different number of cameras. The dataset difference means that we cannot reliably compute "collision loss" and "boundary loss" in the paper.